Predicting Airbnb Listing Price Across Different Cities

Yuanhang Luo (royluo), Xuanyu Zhou (xuanyu98), Yulian Zhou (zhouyl)

December 14, 2019

1 Abstract

Airbnb has become increasingly popular among travelers for accommodation across the world. Accordingly, there are large datasets being collected from the Airbnb listings with rich features. In this project, we aim to predict Airbnb listing price in three cities – New York City (NYC), Paris and Berlin with various machine learning approaches. With one of our best approach – neural network, we have achieved r-squared values of 0.769 in train and 0.741 in test on the NYC dataset, values of 0.762 in train and 0.716 in test on the Paris dataset, and values of 0.816 in train and 0.773 in test on the combined dataset. We showed that neural network that is trained on the combined dataset of NYC and Paris, rather than the individual datasets, is more generalizable to predict the price in a different city – Berlin.

2 Introduction

Unlike hotels, which have their own pricing system, Aribnb prices are usually determined by the hosts empirically. It poses challenges for the new hosts, as well as for existing hosts with new listings, to determine the prices reasonably high yet without losing popularity. On the consumers' side, though they can compare the price across other similar listings, it is still valuable for them to know whether the current price is worthy and if it is a good time to book the rooms.

The nightly price for Airbnb renting depends on multiple factors, and we divide the input type into 4 categories, including continuous, categorical, text, and date features. We have extracted more than 60 features from the dataset. Here we only list a few of them that are both representative and important for the task, such as room size {accommodates, bathrooms, bedrooms, beds, ...}, extra fees {security deposite, cleaning fee, extra people, ...}, reviews scores {review scores rating, review scores accuracy, review scores cleanliness, ...}, location {neighbourhood, latitude, longitude, ...}, facilities {transit, amenities, property type, ...}, and booking related {availability, cancellation policy, host verification, ...}. The ground-truth label is the actual listing price, and we use a variety of regression approaches including linear regression, k nearest neighbor regression, random forest regression, XGBoost, as well as neural network, to predict the value.

3 Related Work

Airbnb price prediction becomes popular due to the availability of large datasets in many cities. In 2015, Li et al.[1] use Multi-Scale Affinity Propagation for price recommendation, and show that it largely improves the precision of the reasonable price prediction. In 2017, Wang et al.[2] worked on Airbnb datasets from 33 cities, and identified the 25 price determinants from a sample of 180,533 accommodation rental offers using ordinary least squares and quantile regression analysis. A similar work by Teubner et al.[3] extracts reputation-related features, and investigate its effect on pricing with linear regression. In 2019, Kalehbasti et al.[4] used multiple machine learning approaches and sentiment analysis on predicting Airbnb price in NYC dataset, and they achieved 0.6901 R^2 value on the test dataset. Recently, Lewis[5] predicted Airbnb price for properties in London by using machine learning and deep learning, and shows that XGBoost provides the best accuracy ($R^2 = 0.7274$), superior to other machine learning models in Kaggle competitions[6].

Despite that multiple projects were carried out on predicting the listing prices, none of them has been performed across different cities. In this work, we focus on the following three tasks. First, we would like to explore different features via feature extraction and engineering. Second, we would like to experiment and compare different machine learning techniques in price prediction. Finally, we want to train a more generalized model and perform transfer learning.

4 Dataset and Features

4.1 Dataset

We use Kaggle datasets [7, 8, 9] for Airbnb listings in NYC, Paris and Berlin respectively. All three datasets contain a detailed listing table with 96 raw input columns/features. The NYC dataset contains a total of 44317 listings,

the Paris dataset contains 59881 listings, and the Berline dataset contains 22552 listings. We split the dataset into train:validation:test with a ratio of 7:2:1 for each dataset.

4.2 Features

Features are chosen only if they are informative and are likely to correlated with the price label. Therefore, we eliminated features like id and host_id, which appear to be noise, features like market and country_code, where the majority of the data take the same value, and features like license and jurisdiction_names, where the majority of the data are null.

- **Label**: The ground-truth label is the listing price. As there exists abnormally high prices in the datasets, we have used two approaches – data thresholding and label transformation – to alleviate this problem. For data thresholding, we cut off data with price over 500 dollars per night, which eliminates approximate 1% of the total listings. For label transformation, we have tested different power transformations, as well as logarithmic transformation, and we found that square root transformation and logarithmic transformation work well for the price prediction. Figure 1 compares a regression model performances without/with label transformation.
- **Continuous features**: We identified highly correlated continuous features and eliminated them from our input, and we obtained 28 continuous features in total (Figure 2 and 3). We filled the null with 0 for price related features (security_deposit and cleaning_fee), and mean value for the rest of the features. We then performed standardization transformation with normalized feature $x \sim \mathcal{N}(0, 1)$. For some of the following experiments, we also included 2-degree interaction terms of the continuous features.
- **Categorical features**: For most of the categorical features, we directly performed one-hot encoding, while for a small fraction of list features, like amenities and host_verifications, we encoded them into vectors via dictionary building and mapping. Altogether, we obtained 20 encoded features.
- **Text features**: To utilize the text features, such as summary, transit, neighborhood_overview, we counted tf-idf on unigrams and bigrams. We then performed truncated singular value decomposition (SVD) to reduce the dimension of each text feature to 50, which makes the dimension of the 12 text features into a 600 dimension vector.
- **Date features**: We have 3 date features (host_since, first_review, last_review), and we converted them into continuous values by filling the null value with the mean date, and subtracting the earliest date value from all date values.

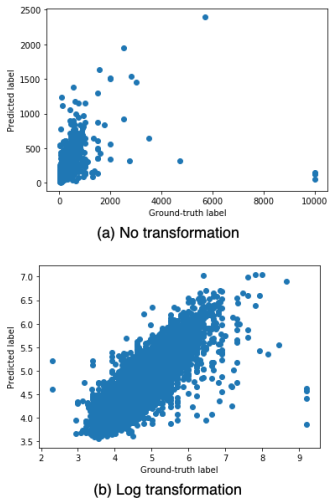


Figure 1: Label transformation with XGBoost on continuous features

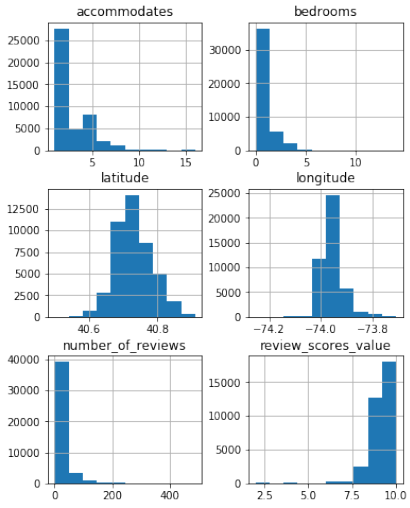


Figure 2: Histogram of continuous feature examples

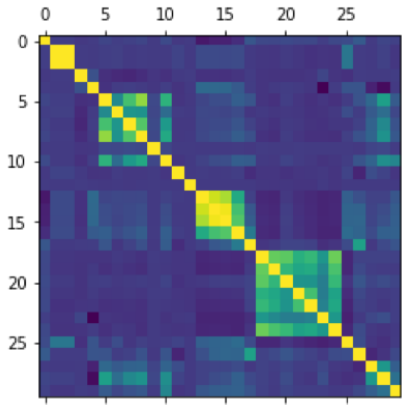


Figure 3: Continuous feature correlation (yellow–high, blue–low)

5 Methods

5.1 Baseline

We used linear regression with L_2 regularization and unweighted K-nearest neighbor (KNN) regression as the baseline models. For linear regression, it approximates label y as a linear function of features x as $h_\theta(x) = \theta^T x$. It aims to

minimize the cost function $J(\theta) = (y - h_\theta(x))_2^2 + \alpha||\theta||_2^2$. For KNN, we identify k listings with features that are most similar to the target listing measured by Euclidean distance $d_{ij} = \sqrt{\sum_{k=1}^{K}(x_k^{(i)} - x_k^{(j)})^2}$, and average their prices as the prediction $p_i = \frac{1}{n}\sum_{i=1}^{n} y_i$.

5.2 Random forest

A Decision Tree is a tree-structured plan of a set of attributes to test in order to predict the output, while random forest is improved from bagged decision tree, and uses modified tree learning algorithm with feature bagging. The model output an average of all k trees' prediction: $\hat{y}_i = \frac{1}{K}\sum_{k=1}^{K} f_k(x_i)$, $f_k \in \mathcal{F}$ (Figure 4).

5.3 XGBoost

XGBoost is a tree boosting algorithm with decision tree ensembles. It is different from random forest in two aspects. First, random forests builds tree independently while XGBoost builds one tree at a time. Second, random forests combines results of all trees at the end, while XGBoost combines results along the way[10]. It minimizes the objective $\text{Obj} = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$ for both train loss and tree complexity.

5.4 Neural network

We adopted 3 fully-connected layers in the multilayer perceptron (MLP) with ReLU activation as follows (Figure 5):

$$\hat{y}_i = W_3(\text{ReLU}(W_2(\text{ReLU}(W_1 x_i + b_1)) + b_2)) + b_3$$

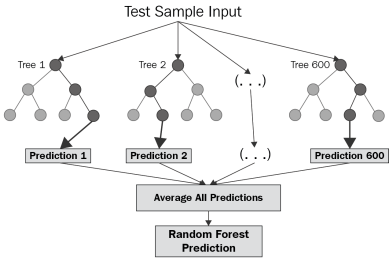


Figure 4: Illustration of random forest regression

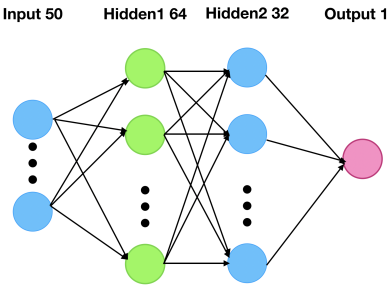


Figure 5: Illustration of the neural network

5.5 Evaluation metrics

The primary metrics is r-squared score $R^2 = 1 - \frac{\sum_i (y_i - f_i)^2}{\sum_i (y_i - \bar{y})^2}$, where y is the true label value, $\bar{y}$ is the average across true label values, and f is the predicted label value. Large r-squared value indicates good fit. We also use mean squared error (MSE) as a reference for performance as $\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - f_i)^2$.

6 Results

6.1 Machine learning

For the results in this section, we used the default settings for all machine learning approaches imported from the sklearn package. Note that we have experimented with different parameters, like regularization strength, learning rate, k in the KNN and etc., and the output metrics stay largely unchanged. For neural network, we used 4 fully connected layers (hidden_size 128, 512, 64) with ReLU activation in each of the hidden layer, Adam optimizer ($lr = 10^{-4}$ and weight_decay $= 10^{-5}$), and the batch size is 64.

Due to the existence of abnormally high price listings, direct prediction on price showed compromised performance (Data not shown and Table 2). Thus, we performed price thresholding with \$500 cut-off, and tested different models on the NYC dataset only with the continuous features. We showed that XGBoost and neural network achieved the best performance.

We further examined the XGBoost model performance without and with different label transformations, and we found that logarithmic transformation largely improves the model performance even without thresholding (Table 2 and Figure 1). After adding the interaction terms to the continuous features, the r-squared value further increases from 0.669 to 0.676.

Table 1: Continuous features with different models

			Train		Test	
Model	**Feature**	**Label**	**MSE**	**r-squared**	**MSE**	**r-squared**
Linear regression	C[a]	TH[b]	3514.64	0.489	3541.99	0.497
K-nearest neighbor	C	TH	4466.30	0.351	4760.61	0.323
Random forest	C	TH	1706.86	0.752	2427.05	0.655
Neural network	C	TH	2059.43	0.702	2367.44	**0.665**
XGBoost	C	TH	2273.98	0.669	2357.83	**0.665**

[a]Continuous feature with normalization
[b]Threshold $y <= 500$

Table 2: XGBoost model with different label transformations

			Train		Test	
Model	**Feature**	**Label**	**MSE**	**r-squared**	**MSE**	**r-squared**
XGBoost	C	N/A	16748.6	0.558	58003.1	0.195
XGBoost	C	Sqrt[a]	7.17806	0.652	10.4435	0.537
XGBoost	C	Log[b]	0.13905	0.695	0.14765	0.669
XGBoost	C+I[c]	Log	0.13061	0.708	0.14704	**0.676**

[a]$y' = \sqrt{y}$
[b]$y' = \log y$
[c]Add 2-degree interaction

In addition to the continuous features, we also extracted text, categorical as well as date features, and performed feature engineering. For text features, we found that unigram and bigram tf-idf represented features shows better performance compared to unigram count or unigram tf-idf (Table 3). While both the text features and categorical features are helpful for price prediction, date feature appears to be uncorrelated to the listing price (Table 3 and 4). Therefore, we concatenated features from the continuous, categorical and text features, and it achieves a r-square value of 0.706 with label thresholding and 0.740 with label log transformation respectively.

Table 3: XGBoost model with different text features

			Train		Test	
Model	**Feature**	**Label**	**MSE**	**r-squared**	**MSE**	**r-squared**
XGBoost	Text - unigram count	TH	4127.11	0.400	4632.63	0.342
XGBoost	Text - unigram tf-idf	TH	3868.55	0.438	4337.49	0.384
XGBoost	Text - unigram & bigram tf-idf	TH	3771.55	0.452	4231.74	**0.399**

Table 4: XGBoost model with date/categorical/text features

			Train		Test	
Model	**Feature**	**Label**	**MSE**	**r-squared**	**MSE**	**r-squared**
XGBoost	D[a]	TH	6654.41	0.033	6927.05	0.016
XGBoost	O[b]	TH	3134.42	0.546	3219.57	0.532
XGBoost	C+O+T[c]	TH	1885.25	0.724	2078.78	0.706
XGBoost	C+O+T	Log	0.11383	0.749	0.11731	**0.740**

[a]Date feature transformed to continuous value
[b]Categorical feature with one-hot encoding
[c]Text feature with unigram & bigram tf-idf

Finally, we tested the same settings with XGBoost on another dataset – the Paris dataset, and observed similar model performance (Table 5).

Table 5: XGBoost model with all features on Paris dataset

				Train		Test	
Model	**Data**	**Feature**	**Label**	**MSE**	**r-squared**	**MSE**	**r-squared**
XGBoost	Paris	C+O+T	TH	1308.47	0.697	1413.89	0.691
XGBoost	Paris	C+O+T	Log	0.11114	0.722	0.11667	**0.704**

6.2 Neural network for transfer learning

We used mean squared loss as the loss function, and SGD + Nesterov's momentum as the optimizer ($\mu = 0.9$, lr $= 0.005$), with a L2 regularization $\lambda = 0.005$.

While training the neural network, we noticed that it is subjected to overfitting issue especially when the feature number is large, and we used two major techniques to alleviate the problem. First, we chose a relatively small number and size of hidden layers (2 hidden layers of size 64 and 32), in contrasting to 3 hidden layers of size 128, 512 and 64 in our pioneer experiments. More importantly, we performed early stopping, which uses 10% of the training data as cross validation, and terminates the training if the change in validation r-squared value is less than a threshold of 0.0001 for 10 iterations.

In the first set of experiments, we compared the training of neural network in individual versus combined dataset (Table 6). Although the two cities do not have all features in common, for example the categorical feature 'neighborhood', we observed improved model performance when training on the combined dataset with a r-squared value of 0.773.

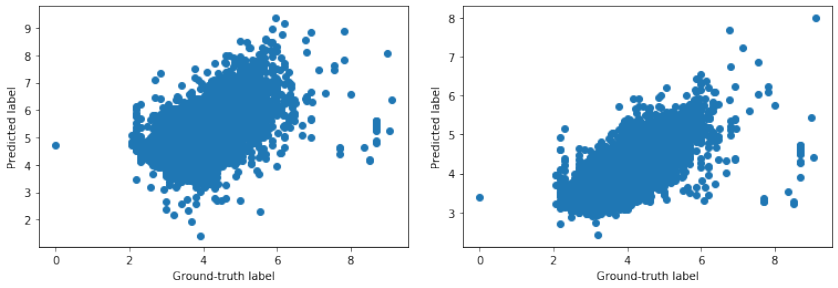
Table 6: Result of cross-dataset learning

			Train		Test	
Data	**Feature**	**Label**	**MSE**	**r-squared**	**MSE**	**r-squared**
NYC	C+O	Log	0.10525	0.769	0.11723	0.741
Paris	C+O	Log	0.09448	0.762	0.11277	0.716
NYC and Paris	C+O	Log	0.08331	0.816	0.10357	**0.773**

In the second set of experiments, we investigated whether the neural network model is transferable to predict price in different cities. For this purpose, we trained a price prediction model on NYC dataset, Paris dataset, as well as on the NYC + Paris combined dataset. Afterwards, we test the trained model on an unseen dataset – the Berlin dataset. We found that the model trained on combined dataset is more generalizable to price prediction on Berlin dataset with a r-squared value of 0.514, than that trained on NYC or Paris dataset alone (Table 7 and Figure 6).

Table 7: Transfer learning tested on Berlin dataset

			Train		Test	
Data	**Feature**	**Label**	**MSE**	**r-squared**	**MSE**	**r-squared**
Train on NYC only	C+O	Log	0.10653	0.765	1.78007	-3.512
Train on Paris only	C+O	Log	0.09494	0.761	7.26637	-17.42
Train on NYC and Paris	C+O	Log	0.10461	0.763	0.19163	**0.514**



(a) Train on NYC dataset, test on Berlin dataset

(b) Train on NYC and Paris, test on Berlin dataset

Figure 6: Transfer learning

7 Conclusion

Overall, we have performed extensive feature extraction and engineering, and experimented with various machine learning approaches in predicting Airbnb listing price. We showed that XGBoost and neural network out-perform other approaches, and achieve r-squared value greater than 0.7. We also showed that by training the neural network on combined datasets from different cities, it augments the generic features and suppresses the city-specific features without explicit feature extraction. This result is of particular interest, as it indicates the possibility of training a generalized model on datasets from multiple cities. In future work, we would like to improve the neural network performance with extra feature extraction and hyper-parameter tuning. More importantly, we would like to investigate into the transfer learning with neural network in a greater depth.

8 Contributions

Yuanhang Luo: part of feature engineering, transformation on labels, run experiments, co-write reports.
Xuanyu Zhou: neural network experiments, transfer learning and cross-datasets learning, co-write reports.
Yulian Zhou: data clean up and feature extraction, coding up neural network and run experiments, co-write reports.

Code

- `https://colab.research.google.com/drive/1wLte7D7tySymBsbORAkGnl2ngYvd76JX`
- `https://colab.research.google.com/drive/1IV6QXPfpaKo3NSLbtcWMprhu1PhtTOCX`

References

[1] Yang Li, Quan Pan, Tao Yang, and Lantian Guo. Reasonable price recommendation on airbnb using multi-scale clustering. In *2016 35th Chinese Control Conference (CCC)*, pages 7038–7041. IEEE, 2016.

[2] Dan Wang and Juan L Nicolau. 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, 2017.

[3] Timm Teubner, Florian Hawlitschek, and David Dann. Price determinants on airbnb: How reputation pays off in the sharing economy. *Journal of Self-Governance & Management Economics*, 5(4), 2017.

[4] Pouya Rezazadeh Kalehbasti, Liubov Nikolenko, and Hoormazd Rezaei. Airbnb price prediction using machine learning and sentiment analysis. *arXiv preprint arXiv:1907.12665*, 2019.

[5] Laura Lewis. Predicting airbnb prices with machine learning and deep learning, 2019.

[6] Kaggle. Airbnb price prediction, 2018.

[7] Kaggle. Airbnb open data in nyc, 2019.

[8] Kaggle. Airbnb paris, 2019.

[9] Kaggle. Berlin airbnb data, 2019.

[10] Stephanie Glen. Decision tree vs random forest vs gradient boosting machines: Explained simply, 2019.