New York Airbnb Price Prediction

Hanjun Wei

Brown Data Science Initiative

GitHub Repository

Introduction
Correct pricing has always been one of the most important factors in for-profit organizations. An inappropriate price above or below the intrinsic product value, will result in a loss. Unlike traditional hotels, which have professional pricing strategies to adjust their price depending on factors such as seasonality, location, etc. Individual Airbnb hosts might be unable to set an appropriate price due to lacking market knowledge.

In this analysis, we will develop a pricing model by taking Airbnb market information in New York City into account and predicting an appropriate price to assist general local hosts in better setting their listing prices. This analysis would be a regression task since our target is to predict the Airbnb price per night in USD.

The data we used in this analysis is New York City Airbnb Open Data. This dataset contains information about all Airbnb Listings in New York City in 2019. The original data was sourced from publicly available information from the Airbnb site and collected by Inside Airbnb site. This dataset has been previously used for many Kaggle Data Challenge projects. Among them, the most common projects are Exploratory Data Analysis (EDA) and predict the availability of a given Airbnb listing. For the EDA-related projects, their potential goal is to have a descriptive analysis of the New York City Airbnb market in 2019. In their study, they found that the Airbnb price per night is highly skewed, and the downtown area tends to have a higher price. Besides, among all types of rooms, the entire room is the most popular. For the availability prediction-related projects, the author has rearranged the numerical variable availability into a binary categorical variable representing whether a given Airbnb has a whole year availability. The author eventually received an Accuracy of 0.83 and a ROC AUC of 0.77. However, I think this result is inappropriate since the target variable is highly imbalanced. I would recommend using precision and recall instead of ROC AUC.

This dataset contains 48,895 rows and 16 columns. Among them, 4 variables contain missing values; they are Last_review (20% are missing), reviews_per_month(20% are missing), host_name (0.04% are missing), and name (0.03% are missing).

Exploratory Data Analysis
In our dataset, there are initially 15 predictor variables and 1 target variable. The target variable y is the price per night in USD for a given Airbnb. The distribution of this variable is highly right-skewed. This makes sense because prices can be very expensive but not below zero. It is also a unimodal distribution with its mode at $100. By calculation, about 2.7% of data can be considered extreme outliers. Among them, the most extreme outlier or the maximum price is $10,000. The minimum price in this dataset is $0; we should drop those 11 observations at $0 since it is impossible to book an Airbnb for free. In another scenario where $0 might indicate a missing value, we should also drop them because missing values are not allowed in the target variable. Here, we should also drop observations considered outliers since our target user groups are general Airbnb hosts. After dropping those rows, we will have 47,511 remaining.

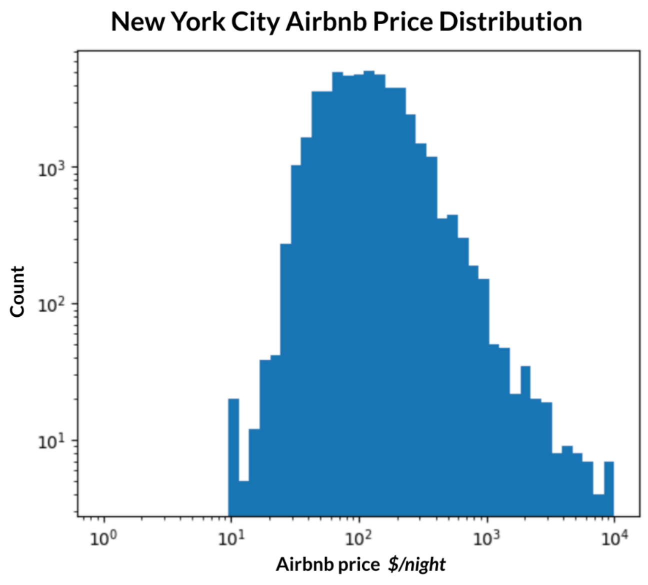


Fig. 1, This plot shows the distribution of Airbnb price per night in USD

For the predictor variables, we first dropped identifier variables id (Airbnb id) and host_id. The reason is that they just function as an identifier and will not give us any information related to the target price. We will also drop unstructured text variables such as name (Airbnb name) and host_name. They might influence the price in such a way that certain words may be more attractive to the customer. However, it requires more advanced NLP strategies which are not in the scope of this analysis. Next, we convert a date type variable last_review which shows the date of the last review Airbnb received into a numerical variable gap_between_last_review_and_end_of_2019 which shows the number of days between the last review date and 2019/12/31. After feature engineering, we have 11 predictor variables remaining.

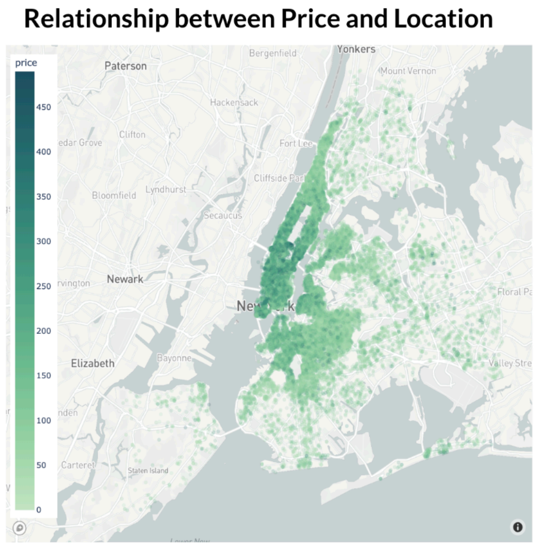


Fig. 2, This plot shows the relationship between Airbnbs price and their geological distribution.

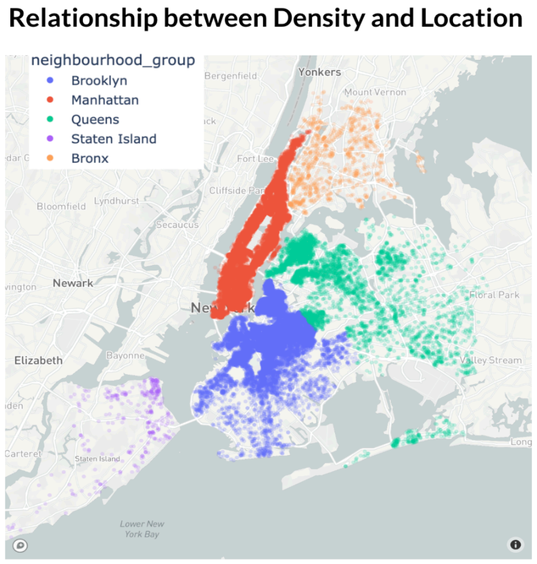


Fig. 3, This plot shows the relationship between Airbnbs neighbourhood and their geological distribution.

By setting aside extreme outliers, we can see there is a clear relationship between the pair of coordinate variables and the target variable price. In Fig. 2, the darker shade represents a more expensive price. As we can see, high price Airbnbs are primarily located in the downtown area (Manhattan and North Brooklyn region). And the price gradually decreased as we moved away from the downtown area. Similarly, from Fig. 3, you might notice that dots are closer to each other in downtown areas. This means that there are more Airbnb in North Brooklyn, West Queen, and Manhattan region. This makes sense because most tourist spots are located in the downtown area. Thus the high demand results in an increase in Airbnb prices. Meanwhile, there will be more supply of Airbnbs to meet the demand.

Another interesting relationship I found is between price and reviews received by Airbnb per month. From Fig. 4, we can see that most dots are gathered in the bottom left corner. And there is a clear negative correlation between price and the review rate. This is reasonable because as the price increases, fewer people can afford it, which results in lower review rates.

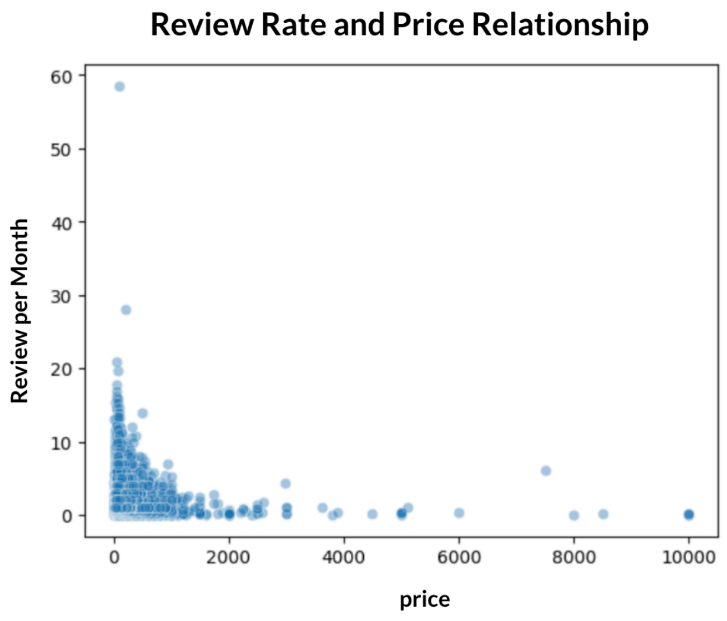


Fig. 4, This plot shows the correlation between monthly review rate and the daily price for Airbnb

Methods

Since our dataset does not have group structures or time-series patterns, we will keep the iid assumption. For the data splitting, the method I applied is the basic train-test split with 60% of data in train set, 20% of data in validation set, and the other 20% of data in test set. Here, I didn't apply any stratification method because our data have a considerably large size, and the overall distribution can be preserved in all sets. For the same reason, I didn't use K-fold validation since it is relatively time-consuming when performed on large datasets.

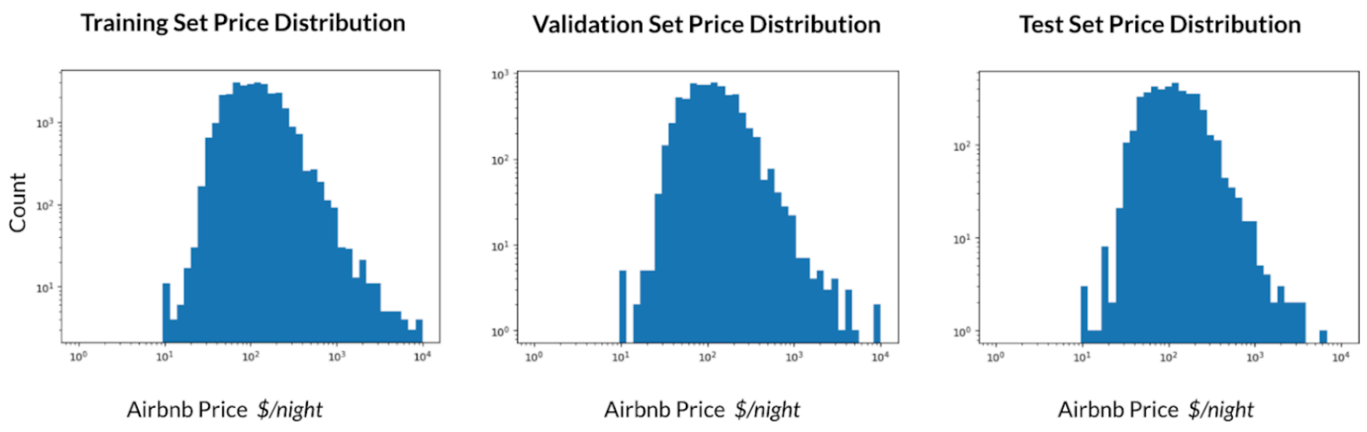


Fig. 5, Data distributions after basic split (60% train, 20% val, and 20% test).

In terms of preprocessing. I have used 4 types of preprocessors.

- For variable room_type, I used the ordinal encoder because there is an ordinal relationship among shared rooms, private rooms, and entire rooms.
- For variable neighborhood groups and neighborhoods, I used the one hot encoder because there is no ordinal relationship among neighborhoods.
- For variable availability_365, I used MinMaxScaler since the available day in a year is between 0 and 365.
- For the rest of the variables, I used StandardScaler because all of them are numerical variables and are not bounded.

After preprocessing, the number of predictor variables increased from 11 to 231. This is because the one hot encoder has converted each element in categorical variables into a new feature.

In the initial dataset, 4 out of 16 variables contain missing values. They are last_review (20% missing), review_per_month (20% missing), host_name (0.04% missing), and name (0.03% missing). We will ignore the last two variables since we have already dropped them. However, there are still 2 columns remaining. To deal with this missingness, the first strategy we will apply is XGBoost. XGBoost is a scalable, distributed, gradient-boosted decision tree model which can automatically handle missing values by deciding for each missing value which is the best way to impute them. However, for other models that cannot take in missing values, we will train them through the pattern submodel approach.

Fig. 6 shows that the missing values in the two columns appear together. In other words, there are only two patterns excited in this dataset. Thus, for models trained through the pattern submodel approach, the overall cross-validation pipeline is first started by data splitting and preprocessing. Then, splitting the test set again according to the test set

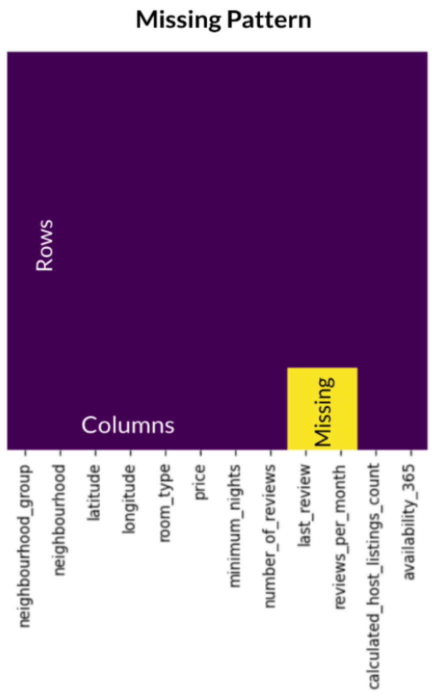


Fig. 6, Missing Data Pattern where yellow indicates missingness data pattern. For each pattern, we decided on the machine learning algorithm with its corresponding hyperparameter. And for each combination of hyperparameter, we will use the portion of train and validation data that match the given pattern to train and validates the model and pick the model with the best validation score and make a prediction on the test data. Then, do the same thing again for the other pattern and calculate the overall test score. Lastly, iterate this procedure 5 times to take the uncertainties due to data splitting and the non-deterministic ML model into account. For the XGBoost model, the cross-validation procedure is the same but just assumes only one pattern excited. Thus, we don't need to train the model for different patterns separately.

There are four ML algorithms that we will try in this project, and they are random forest, Ridge regression, KNN, and XGBoost. In random forest, the hyperparameters I tune are the maximum depth of a tree (max_depth) and the number of features in the model (max_features); In ridge regression, the hyperparameter I tune is the L2 regularization term (alpha); In KNN, the hyperparameter I tune is the number of neighbors in the model (n_neighbors); and in XGBoost, the hyperparameters I tune are the maximum depth of a tree (max_depth) and the L2 regularization term (reg_lambda). Among all hyperparameters, only "max_features" in Random Forest have bound values, so the possible values I tried are linearly spaced ([0.25, 0.50, 0.75, 1]). For the rest of them, I tried log-spaced values since they do not have an upper bound ([1, 3, 10, 30, 100]). The metric we will use is Root Mean Squared Error. Since it has the same unit as our target variable (USD/night), we could have more intuitive understanding when analyzing the results.

Results

After training each model. The final test RMSEs are illustrated in Fig. 7. As you can see, the XGBoost results in the lowest average RMSE score (56 USD/night) among all models. This test performance is 27 standard deviations away (below) from the baseline model. And by applying the XGBoost algorithm, we have reduced the prediction error from the baseline by 26 USD/night on average. Here, we will use the XGBoost as our final model since it has the best prediction power.

To understand the importance of each feature in our final model (XGBoost). We will compare and evaluate the results of three feature importance methods. As we can see in Fig. 8, the top three essential features in permutations and SHAP importance are "ord_room_type", "std_longitude", and "std_latitude".

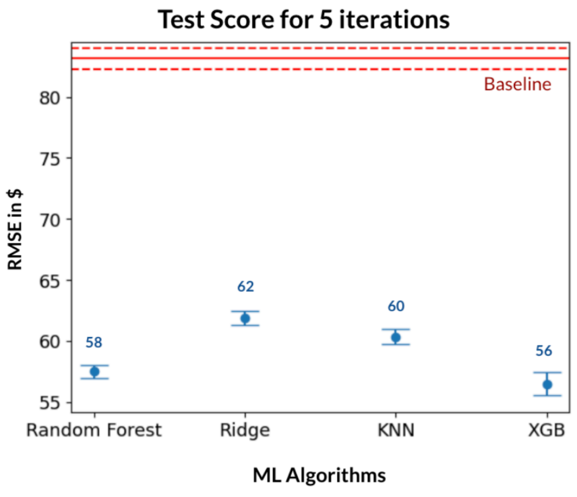


Fig. 7, The overall test RMSE in USD for each model

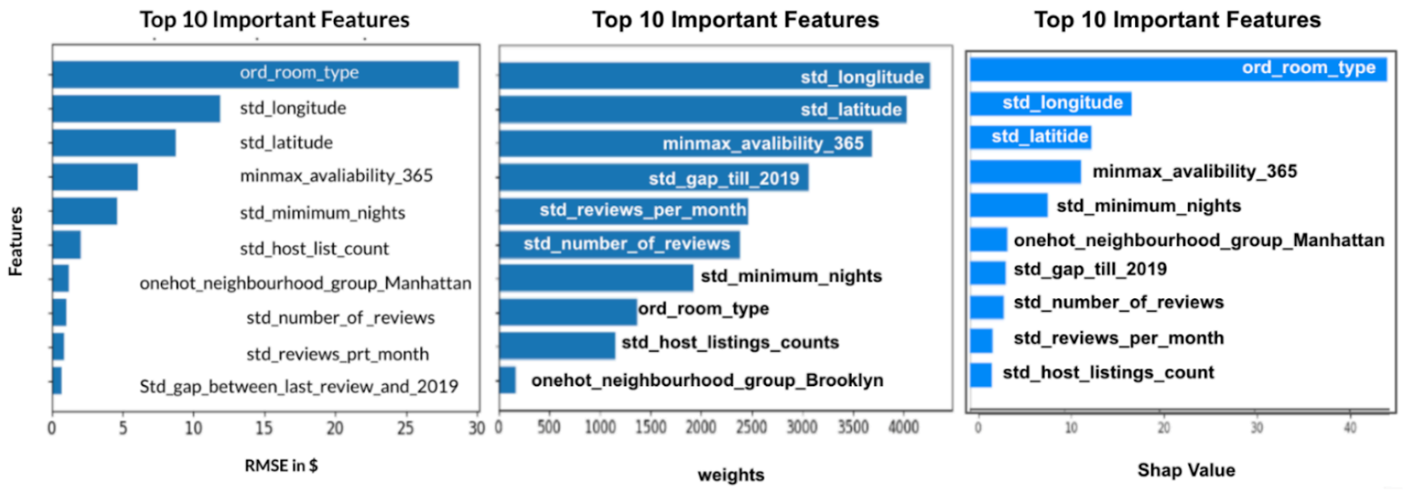
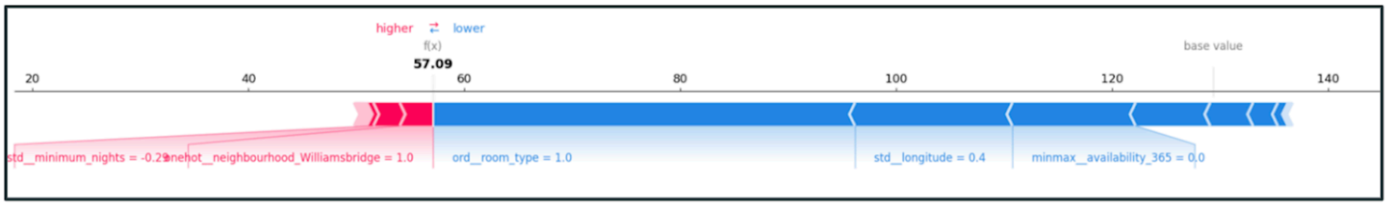


Fig. 8, Top 10 global feature importance with permutation (left), weight (middle), and Shap (right).

This makes sense because the room type and location of Airbnb could hugely influence its price as private-rooms tend to be more expensive than shared-rooms and prices in downtown are more expensive than it in other regions. The reason room type didn't being considered as the top three important features in weight global importance is because Weight favors numerical and high cardinality features.

To understand how does our model making individual predictions, we will have a look at two scenarios where the predicted price is low and high.

E.g. Low predicted price



E.g. High predicted price

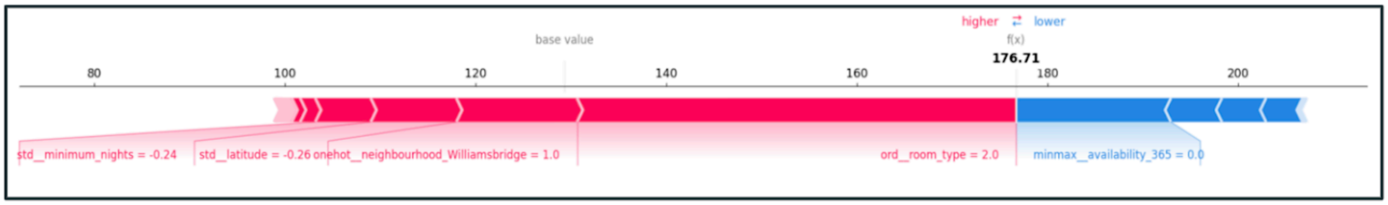


Fig. 9, The local importance of XGBoost model when predicted price is low (top) and high (bottom).

As Fig. 9 shows, "ord_room_type" is the most important feature in both cases, except it contributed negatively in the first but positively in the latter. This is because, in the low-price case, the room type is shared, and in the high-price case, the room type is private. The least essential feature in both cases is "minimum night" which is surprising since it's intuitive to think a room with a higher minimum night requirement may reduce its price to attract more guests.

Outlook

In this modeling process, we didn't focus too much on feature engineering. As we discussed, the target variable prices are heavily right-skewed. It might require log transformation to improve linear models' performance, such as ridge regression. Besides, we should tune more hyperparameters in XGBoost to further improve its prediction power since there are so many of them to try for. And we could also try more time-consuming algorithms such as SVM when time is available. Lastly, our data are collected in 2019, we might want to collect more recent data to reflect today's New York Airbnb market.

Reference

1. *Dgomonov. (2019, August 12). New York City airbnb open data. Kaggle. Retrieved October 18, 2022, from https://www.kaggle.com/datasets/dgomonov/new-york-city-airbnb-open-data*
2. *How is Airbnb really being used in and affecting the neighbourhoods of your city?* Inside Airbnb. (n.d.). Retrieved October 18, 2022, from http://insideairbnb.com/
3. Cao, J., Choi, T., & Khare, R. (n.d.). *NYC Airbnb Listings in 2019: Determining Factors that Affect a Listing's Price*. Retrieved from https://www.stat.cmu.edu/capstoneresearch/spring2021/315files/team5.html
4. Gcdatkin. (2020, October 30). *NYC Airbnb availability prediction*. Kaggle. Retrieved October 18, 2022, from https://www.kaggle.com/code/gcdatkin/nyc-airbnb-availability-prediction