Airbnb Pricing in New York City

—

Jixi Dai * Zicheng Hu † Ziyu Liu ‡

November 5, 2019

Abstract

Home-stay is a rising alternative of hotels for many people around the world and home-stay is the cheaper option for most of the time. Therefore, the pricing of the home-stay property plays an important role in selecting properties. This project employed big data techniques to establish models that examines what price a rational property owner would probably want to rent to others, where the price is neither too high to scare people off nor too low to dissatisfy him or herself. The main goal of this project is to provide a reasonable price range of the Airbnb room listings in New York City area, as a reference for both the hosts and the tenants. Although the price of a listing is typically set by the host, there is always a need for good price estimation. On the one hand, the hosts need this information for strategic pricing in order to make their listings both attractive to tenants and profitable to themselves. On the other hand, the travelers seeking for home-stays want to know the potential price range of listings near their destination so that they can better plan their expenditure and find good bargains.

*Cornell University, *jd999@cornell.edu*
†Cornell University, *zh343@cornell.eduu*
‡Cornell University, *zl722@cornell.edu*

Contents

1 Explanatory Data Analysis

1.1 Data Description

Our project examines the *New York City Airbnb Open Data* from Kaggle.com. This dataset provides detailed features of the listed properties on Airbnb, such as listing price, neighborhood, room type, availability and minimum number of nights required for booking. It also includes additional information such as property descriptions and reviews from previous bookings. Before preprocessing, the data contains 13 features and a total of 48895 observations.With this dataset, we would like to examine how various features influence the listing price of a property. We will try to capture such relationships by building different models (linear model, polynomial model, etc) after some appropriate data preprocessing.

1.2 Data Visualization

1.2.1 Variable Correlations

For numerical features, we plotted a correlation plot to visualize which aspects of the listed properties are relatively important in determining the price (unit: dollars/night).

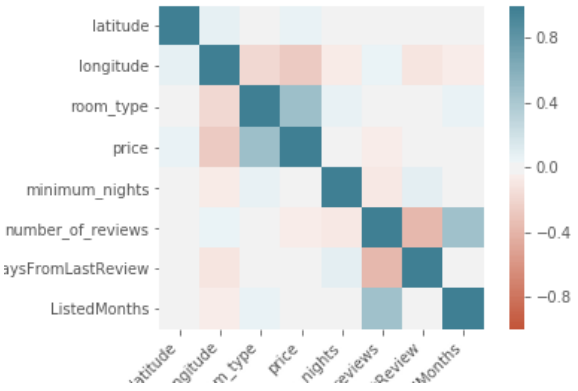

Figure 1: Correlation Plot

As we can see in Figure 1, price is highly correlated with latitude and room_type (property room type) and number_of_reviews (number of past reviews). This inference is in line with reality as the rent of properties located in Manhattan is generally more expensive than the rent of properties located in other counties, and an entire room normally costs more than a shared room or a private room. Also, some numerical features seem to have little correlation with the rent price so it's appropriate to use regularization or variable selection while training models.

1.2.2 Airbnb Prices

Figure 2 shows the distribution of listed Airbnb prices. Based on the plot, the listing price distribution is right-skewed and we fixed it by taking the log transformation (we will discuss the reasons for such transformation later in section 1.3.6).

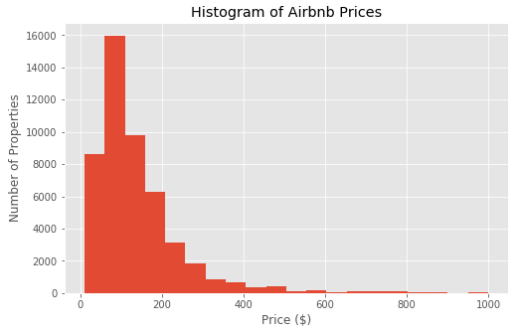

Figure 2: Airbnb Prices Histogram

As the rent price is correlated with latitude, we plotted the distributions of the rent prices in the five neighborhood groups around New York City (Manhattan, Bronx, Queens, Brooklyn, Staten Island). From Figure 3, the average price in Manhattan is the highest while the average price in Bronx is the lowest. Moreover, Manhattan also has the highest price variability compared to the other four neighborhood groups. These observations indicate that features containing property geographic information could be useful for predicting the listing prices.

Figure 3: Aribnb Prices in Different Counties

1.3 Data Preprocessing

1.3.1 Outliers

The conventional definition of an outlier is a data point that falls more than 1.5 times the interquartile range below the first quartile or above the third quartile. In this case, however, we do not stick with this definition due to the nature of our data. Instead, we only considered the price interval [10, 1000] based on common sense. (Prices below $10 are unlikely and barely profitable, and luxurious listings higher than $1000 are rare.) As a result, we dropped 250 outliers out of 48895 observations (0.5% of the data).

1.3.2 Ordinal Values

To facilitate our regression modeling, we transformed the "room_type" column into ordinal values. The average prices for the 3 room types are respectively $66 for shared rooms, $85 for private rooms, and $193 for entire homes/apts. This demonstrates an ordinal relationship among room types so we labeled 1, 2, and 3 for each type in ascending price order.

1.3.3 Nominal Values

The columns "neighborhood_group" and "neighborhood" contains nominal values. Since "neighborhood" specifies the district within each "neighborhood_group", the two columns contain overlapping information. To avoid correlation-related problems while keeping the geographic information intact, we concatenate these two columns into a single column of nominal values (named as "neighborhoods"). In addition, since there is no apparent ordinal relationship between different neighborhood values, we applied one-hot encoding on the new column to facilitate our model building and predictions.

1.3.4 Additional Features

The column "last_review" contains hidden significance indicating the freshness of a home listing, but its timestamp values are not intuitive enough for interpretation and modeling. Therefore, we reshape this column by calculating the difference between the current date and the original stamps, which is just the number of days since the last review. The new column ("days_since_last_review") gives a simple and reader-friendly integer value to measure how recent the home has been reviewed. Besides, we also replaced "reviews_per_month" with "month_listed" (i.e. "number_of_reviews" / "reviews_per_month"). This new column shows how long a listing has existed, which is more relevant to the price.

1.3.5 NA Values

Due to high data integrity, the NA values only appeared in new columns from 1.3.4, which were inherited from the original "last_review" and "reviews_per_month". For "days_since_last_review", it is hard to determine whether these missing values were caused by misrecording or simply no user reviews (we believe both causes are reasonable and likely), so we replaced these missing values by the overall median. The mean replacement is not suitable in this situation because our data has an apparent right-skewed pattern. For "month_listed", missing values from "reviews_per_month" generally indicate freshly listed properties, so we filled in 0 as replacement.

1.3.6 Response Transformation

According to the histogram of our price range (between $10 and $1000), the response variable is highly right-skewed with the majority of points lying between $10 and $250, and a long tail to the right. This is consistent with our intuition that most people are going for fair-priced listings rather than premium or luxurious listings. In order to reduce the effect of the minority expensive listings and stabilize the variance in our data, we can transform the distribution of prices by taking the logarithm of their values. This technique will help us improve the accuracy of regression models in the later phase.

2 Regression Modeling

2.1 Model Evaluation

We used Root Mean Squared Error (RMSE) between the logarithm of the predicted price and the logarithm of the actual sales price. As we mentioned in the previous section, by taking the logarithm on the prices we can make sure that the errors in predicting high prices and low prices will affect the result equally.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}} \tag{1}$$

2.2 Linear Model

We first fitted a simple linear model by solving the least squares problem and used it as a baseline for our project. In this case, there is no unique solution to the problem due to some linear dependence between features. Therefore, we need to include a regularization term to ensure the validity of our linear model.

2.2.1 Lasso Regression

Firstly, we added al1 regularizer to the least squared optimization problem to ensure a unique solution. Fitting a lasso model as our basic model could both give us a benchmark on how models are performing on our dataset and help us understand the importance of the variables in determining the prices.

$$minimize \ \sum_{i=1}^{n}(y_i - w^\intercal x_i)^2 + \lambda \sum_{i=1}^{n} w \tag{2}$$

Then we implemented a 5-fold cross validation to find the best λ that minimizes the RMSE. The optimal λ is 0.0000045 and the resulting test RMSE is 0.44915 while the training RMSE is 0.43924. We can see that both the training error and the test error are large, so the Lasso regression model is probably underfitting and hard variable selection should not be used on our dataset.

2.2.2 Ridge Regression

To prevent underfitting, we then replaced the l1 regularizer with a quadratic regularizer so we could both ensure a unique solution and keep all our features.

$$minimize \ \sum_{i=1}^{n}(y_i - w^\intercal x_i)^2 + \lambda \sum_{i=1}^{n} w^2 \tag{3}$$

Here we also implemented a 5-fold cross validation to find the best λ that minimizes the RMSE. As shown in Figure 4, the optimal λ is 0.0000045 and the resulting test RMSE is 0.44911 We can see that there is a minor improvement on test scores comparing to the Lasso regression model.

In general, linear models on property prices versus property features produce relatively high training errors and test errors, so we need to either find more property features or use more complex regression models on our dataset. Knowing that our purpose is to price Airbnb properties, it's natural to first seek for data on other property features, like amenities, nearby facilities and past reviews. However, as we only have property ids, we couldn't find the matching amenity data. As finding new data and features is harder in

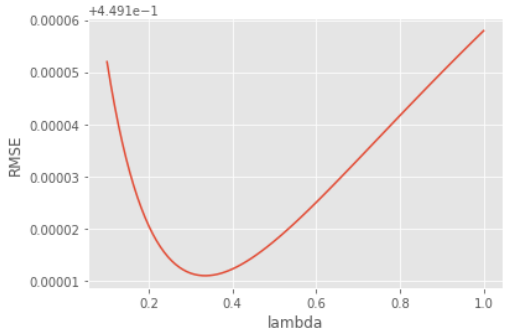

Figure 4: 5-fold Cross-Validation for Ridge Regression

our case, we took our next step to fit more complex models.

2.3 Tree-based Models

Based on previous results, our modeling still has space for improvement because there could be some nonlinear relationship between the input features and response that are not captured by the linear models. In order to adapt such possibilities and achieve higher prediction accuracy, we will attempt on more complex tree-based models. Since a single tree is most likely to underfit considering the size of our data, we decided to use bagging and boosting on these tree-based models to both increase flexibility and control variance.

2.3.1 Random Forest Regressor

The random forest algorithm introduces the bagging technique on decision trees. Essentially it collects a number of individual decision trees that operate as an ensemble and decorrelates these trees by splitting a random subset of features. The key parameter of this algorithm is the number of trees, for which we chose 10 different values to test on our regressor: 100, 200, 300, 500, 1000, 1500, 2000, 3000, 4000, 5000. According to Figure 5, the lowest RMSE we can achieve is 0.48710 by choosing 2000 different trees. Unfortunately in this case, the result did not show any improvement over our regularized linear models (in fact, worse performance than the linear models). This is partly due to the lack of features in our dataset. As a more flexible model, the random forest generally works well on large dataset with sufficient number of features, while the limited number of features of our dataset on hand may not be able to see the full power of this algorithm.

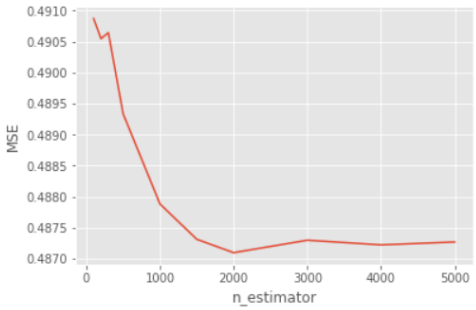

Figure 5: RMSEs for Random Forest with different number of trees

2.3.2 Gradient Boosting Regressor

While the random forest model averages large trees with low bias and high variance, boosting technique combines small trees with high bias and low variance. Boosting tree-based model is sequentially grown using information from previous grown trees and it fits each tree on a modified version of the original data to achieve a low overall bias. As a first attempt, we tried to fit a regular gradient boosting regressor. A potential problem with traditional gradient boosting is that it could learn too fast and thus ending up overfitting the data. To solve this problem, we tuned many different boosting parameters and kept the learning rate low (10 values strictly between 0.01 and 0.1). After fitting on test dataset, we found that the optimal learning rate is 0.07 and the corresponding cross-validation error is 0.40907. This is significantly lower than the previous linear models, so using a gradient boosting model on our dataset indeed helps with the underfitting problem. Also, comparing to random forest regressor with bagging technique, gradient boosting re-

gressor provided more flexibility and had significantly better performance on prediction accuracy.

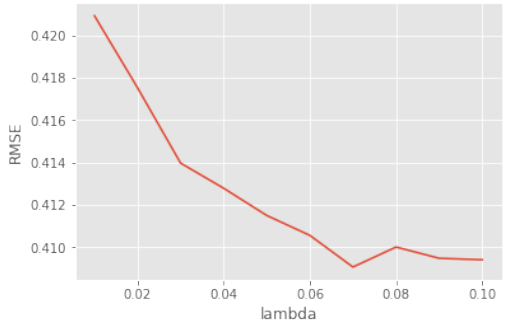

Figure 6: RMSEs for Gradient Boosting Regressors with different learning rates

2.3.3 XGBoost Regressor

XGBoost (Extreme Gradient Boosting) is an advanced implementation of the gradient boosting algorithm which uses more accurate approximations to find the best tree model. XGBoost has 3 major differences from traditional gradient boosting methods. First, XGBoost looks at the distribution of features across all data points in a leaf and reduces the search space of possible feature splits based on the previous information. This is more efficient than traditional gradient boosted trees which normally consider loss for all possible splits. Second, XGBoost includes a wide variety of hyperparameters for advanced regularizations so it can better ensure on overfitting prevention, whereas our previous gradient boosting can only address this concern by controlling the learning rate. Lastly, since its core algorithm is parallelizable, XGBoost has lower computational costs and can be applied to larger datasets. Although these advantages sometimes help XGBoost outperform many other algorithms, they do not guarantee better performance in every case, as the error rate on each model depend heavily on the specific dataset. To keep our models and metrics consistent, we apply the same tuned parameters for XGBoost and check its performance on the same range of learning rates (0.01 to 0.1). As a result, the optimal learning rate is also 0.07, yielding the best RMSE of 0.41529.

2.4 Model Selection

From the table below, the gradient boosting regressor gives the best test RMSE. If no additional features information can be found, we should use a gradient boosting regressor to make smart pricing predictions.

Cross-validation RMSEs for different regressors	
Regressors	RMSEs
Lasso	0.44915
Ridge	0.44911
Random Forest	0.48710
Gradient Boosting	0.40907
XG Boosting	0.41529

Table 1: Table for model selection

3 Conclusion

Smart pricing models are being used by online rental marketplace like Airbnb to give reference prices and we were trying to produce a smart pricing model based on Airbnb New York Open Data. Through our modeling processes, we noted that linear models tend to underfit our dadaset. To overcome the underfitting problem, we used several more flexible models: random forest, gradient boosting, and XGBoost. It turns out that the gradient boosting regressor produced lowest test error. Hence, with the current dataset, one should use the gradient boosting algorithm to get a better reference to the market prices.

4 Limitation and Fairness

A potential problem on the smart pricing model is that property brokers like Airbnb could force lower property rent prices based on our model to attract more customers to the online marketplace and improve their

overall profits. And this potential abuse of the smart pricing model may hurt the property owners because they will have to take prices that are lower than the intrinsic value of their properties.

As for fairness concern, since the properties of different property types or located in different geographical regional are being evaluated by the same model, there is no discrimination on property features and this fact guarantees the fairness of our method.

5 Potential Improvement

5.1 Additional Features

We made some improvement on regression error after we used more flexible models on our dataset. However, the most flexible model among all our models, random forest regressor, was not performing well on our dataset. A main reason for such under-performance, as we mentioned, is the insufficient number of features. This is also the most important concern for this project. Therefore, the first and foremost future work that we would take for improvement is to find more geographic-related data, for example crime data, living standard data and traffic data, so that there is more complete information on listed properties.

5.2 Word Embedding

Another potential improvement is to apply word embedding to evaluate the relevance of each keyword to the home listing price. Introducing these additional features is also the last exploration step we ccan take for our current dataset. In general if we want better prediction results, we will need additional data with more feature information relevant to the listing price.