Final Report

December 4, 2019

Title: New York City Airbnb Price Prediction
Name: Guanzhong Chen
Class: Data 1030
Brown ID: 140268394
GitHub Link: https://github.com/frank07080/Data-1030-Project

0.1 Introduction

Airbnb stands for "AirBed and Breakfast." More and more people nowadays choose to live in Airbnbs instead of hotels when they are out traveling. There are lots of reasons why Airbnbs become more popular than hotels when it comes to where to live during traveling. One of them is that you are renting houses from a person that give a total different feeling from hotels. Another one may be that Airbnbs are, in a sense, cheaper than hotels.

Price of Airbnb, then, becomes our target variable especially in city like New York. So why is price important? The reason is that most people would to take price as their first consideration when budgets are taken into accounts. It would be convenient if people can predict the price of an Airbnb if they are given enough information of that Airbnb. People usually care if the price of an Airbnb is more than 100 bucks. Therefore, we make it a binary classification problem. Given a set of features, we want to be able to predict whether the price is more than 100 bucks.

The dataset of New York City Airbnb we are using is from Kaggle. Here is the link https://www.kaggle.com/dgomonov/new-york-city-airbnb-open-data. This dataset has a total of 48895 samples and 16 features, such as neighbourhood, room type, longitude, latitude, and so on.

0.2 EDA

There are several EDA we can do with this dataset. We first want to see which neighbourhood is the most popular one. Therefore, we check the count of listings for each neighbourhood. And below is what I get:

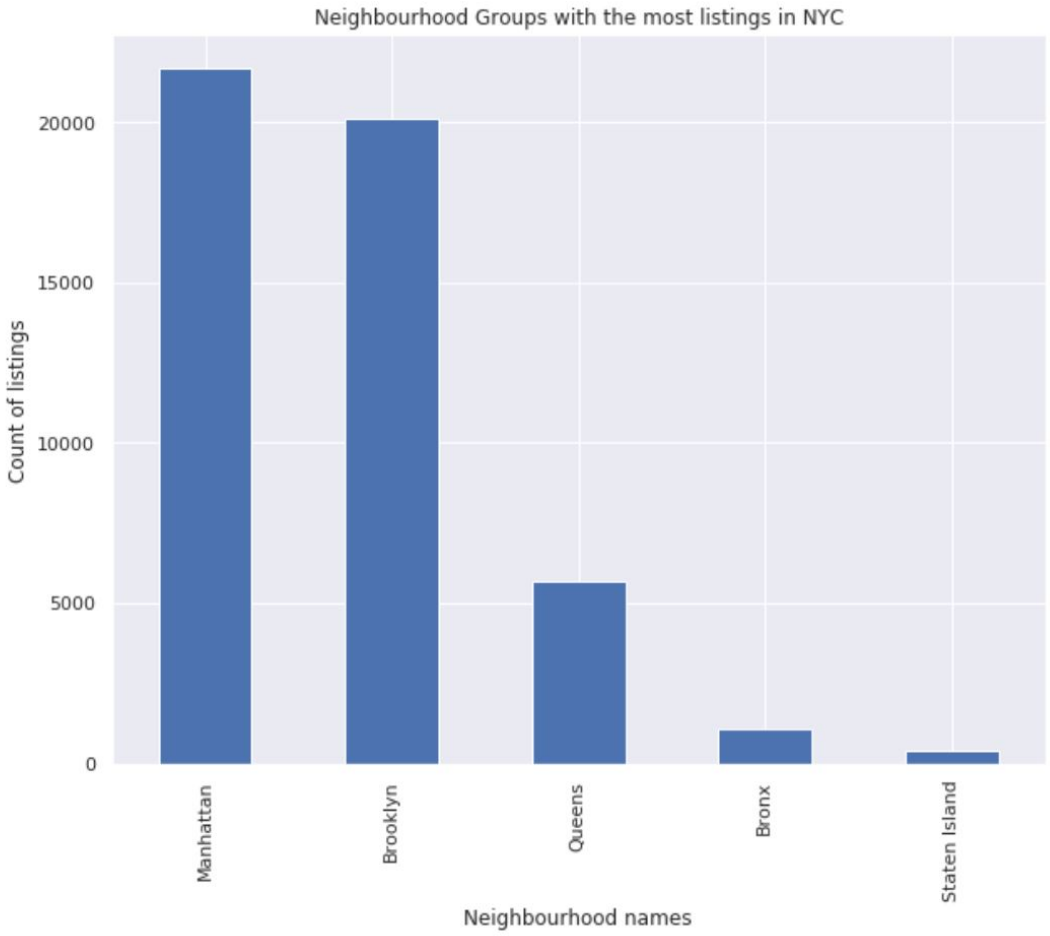

Figure 1: *It shows how many Airbnbs are listed in the data set with respect to each neighbourhood group. Note that the neighbourhood group Manhattan is the most popular one.*

As "Figure 1" shows, Manhattan is the most popular neighbourhood group. The second thing we want to see is for each room type, how many of them in proportion has a price bigger than 100 bucks. Here room type of 0.0, 1.0, 2.0 correspond to "Entire home/apt", "Private room", and "Shared room". Price of 0.0 and 1.0 correspond to "<=100 bucks" and ">100 bucks".

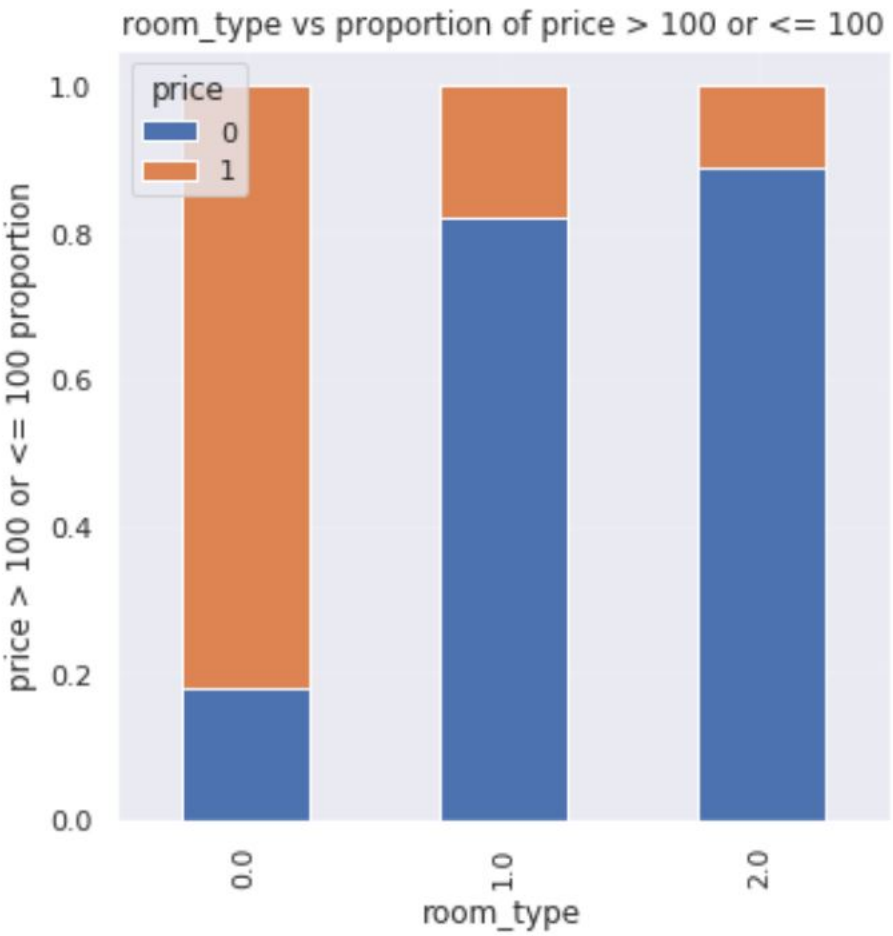

Figure 2: *It shows for each of the room type, what the proportions are for Airbnbs that have prices bigger than 100 bucks and Airbnbs that have prices less than or equal to 100 bucks. The proportions sum to 1. Here room type of 0.0, 1.0, 2.0 correspond to "Entire home/apt", "Private room", and "Shared room". Price of 0.0 and 1.0 correspond to "<=100 bucks" and ">100 bucks". Note that for room type of entire home/apt, expensive Airbnbs (more than 100 bucks) take in a big number of proportion.*

As "Figure 2" shows, for rooom type of "Entire home/apt" Airbnb, a large proportion of them has price bigger than 100 bucks. And for rooom type of "Shared room" Airbnb, a large proportion of them has price less than or equal to 100 bucks. The last EDA we do will be number of reviews versus the log of price. Here we treat the price as its original number (not binary types) and take the log of it.

Figure 3: *This figure shows the number of reviews for each log of prices of Airbnbs in NYC. Note that points tend to be centered in the middle of log price instead of both ends of log price.*

As "Figure 3" shows, people usually don't review either too expensive or too cheap Airbnbs. It may either because people can not afford the price of expansive Airbnb or because people are skeptical of the quality of cheap Airbnbs.

0.3 Methods

For this project, I have selected 4 original features to do this binary classification. After checking with original data, I only have missing values in feature called "reviews_per_month". And this is a continuous feature. Therefore I apply Iterative Imputer to this feature.

Then I start preprocessing data after working with missing values. I apply Ordinal Encoder/One Hot Encoder for categorical features. And I apply Min Max Scaler/Standard Scaler for continuous features. Total features I have after preprocessing is 8.

For ML pipeline, the very first step is to separate the data into three different sets: training set, cross validation set (CV set), and test set. By checking performance in CV set, we can get the best set of hyperparameters and the best model to apply to our test set. The way we do cross validation is called Stratified K-fold cross validation. In this way we are going to measure the average of K CV sets for a single set of hyperparameters. We then take the best average result over all sets of hyperparameters. That's how best set of hyperparameters and best model are chosen.

Next step in ML pipeline is to select the two classifier models: Random Forest Classifier and Logistic Regression. For random forest classifier, the parameters I have tuned are "max_depth" and "min_samples_split". For "max_depth", I have tried values of 10, 30, 100, and 300. For "min_samples_split", I have tried values of 16, 32, 64, and 128. For Logistic Regression, the parameters I have tuned are "penalty" and "C". For "penalty", I have tried values of "l1" and "l2". For "C", I have tried values of np.logspace(0, 4, 10).

After we get our best model, we apply it to test set. To see how it performs in the test set, the metric I use to evaluate my models' performance is accuracy score. That is the proportion of correct classification over all observations in the test set. This is because our final goal is to correctly classify as many new observations as possible. Therefore we take the accuracy score as our classificaiton performance metric.

For uncertainties due to data splitting, we try different values of random states of K-folds CV and take the average and standard deviation of its results. We note that the average of test set accuracy score is 82% with small standard deviation. Therefore, we have controlled the uncertainties due to data splitting. We also set random states for random forest classifier to control uncertainties brought from non-deterministic ML method.

0.4 Results

The accuracy score for baseline model is about 51%. That is if we randomly guess if an Airbnb has a price bigger than 100 bucks, we will have 51% of chance to get it correct. After machine learning pipeline, our random forest classifier and logistic regression classifier both achieve the accuracy score about 82%. That is much better than the accuracy score of a baseline model.

For this project, I applied the local feature importances method (shap value method) to do model inspection. That is, we explain the feature importance ranking for a specific observation in the test set. In the following force graph, we are going to explain how a specific observation in the test set gets such a predictive score for class 1 (price bigger than 100) from its baseline score. In other words, what feature brings the highest influence to make the observation its predictive score of class 1.

Figure 4: *This is a force graph shows the local feature importances for a specific observation. For this observation, we see that it gets a predictive score of 0.93 for class 1 (expensive Airbnb) mainly because its room type is Entire room/apt and it locates at Manhattan.*

From "Figure 4", we see that the most important feature to make the observation (Airbnb) expensive is its room type. It's expensive mostly because it is an entire house/apartment instead of a shared room. The second most important feature is it locates at Manhattan. It is expensive secondly because it locates at Manhattan instead of anywhere else. It's mostly not because of its reviews per

month or number of reviews (too small to see from the graph). In the context of current society, room type and location decides the Airbnb price mostly.

0.5 Outlook

There are several things I can do to improve my model. The first thing I can do is to try some different models, such as neural networks and SVC. I may try these out in the future to see if they give me a better accuracy score in the test set. The other thing I can do is to try a bigger number of hyperparameters. Say currently I have tried 16 sets of hyperparameter for a machine learning model. In the future I can try 64 sets of hyperparameters. Of course the drawback is that it would consume more time to do the hyperparameter tuning. The third thing I can think of to improve my model is to use a bigger number of features. More features may mean better classification accuracy. In the meantime, more features require more time of training.

0.6 References

https://www.kaggle.com/dgomonov/new-york-city-airbnb-open-data

[ ]: