Examination of The Factors That Drives The Airbnb Rentals to Success Across New York City

Manxueying Li,
Ziwei Zheng

05/04/2020

Introduction

In such a busy and crowded city as New York, more and more people are considering airbnb rentals. Evaluation is critical to the entire airbnb community, enabling tenants to choose their own accommodation plan wisely and enabling landlords to open their homes with confidence, attract tenants and provide intimate accommodation services. The fact that the airbnb business is based on positive reviews is not much of an exaggeration. Each comment helps the guest decide whether to book a certain room or not. The more positive reviews there are, the more landlords earn, according to airbnb.

As for how the mechanism works, after each stay, the landlord and the tenant have a chance to evaluate each other. Both parties have 14 days to write an evaluation after checking out. Unless both landlord and tenant have submitted an evaluation, the evaluation is hidden. The reviews are then posted on the tenant's profile and the landlord's listing and profile pages. As a result, the number of reviews shown on the profile could possibly reflect the popularity of the rent. This project, we take the number of reviews as an indicator of the popularity of given rent and we zoom into the factors that would affect its popularity.

There are a lot of factors that would affect the popularity of the rent. In this project, we collect a dataset that contains 15 factors that could potentially affect the popularity, including name of the airbnb, the location of the airbnb, its price and availability, etc. In this project we will delve into those important factors for hosting an Airbnb and how those factors would affect the popularity of Airbnb hosting. It would be potentially helpful for people who are hosting Airbnb in NYC to find out how they can be successful in attracting tenants.

There exist a lot of different correlations in this dataset, and we have divided the problem into two major parts: text processing and data processing. We have noticed that name is an important factor in people’s decision process for finding a rent. Different names can have positive or negative impact on people’s first impression for the host and the room, and therefore, it would be interesting to explore how it would affect the popularity for the airbnb. In order to resolve this correlation, in the first part of our project, which is text processing, we evaluate the relationship between the name of the airbnb and its number of reviews per month to see how its naming would affect people’s decision in choosing the rent. Name is not the single factor that would affect people’s preference toward a rent. In the section of data processing, we deal with those numerical data. For those that are not numerical, for example the neighbourhood group, we use one hot coding to convert the group to numerical data. In this part of the project, we relate the number of reviews to other quantitative measures (to be specific, 'neighbourhood group', 'latitude', 'longitude', 'room type', 'price', 'minimum nights', 'availability 365'), in our dataset aiming to find out how does each of them contribute to the popularity of airbnb.

In this report, we will present our approach to the problem, the difficulties we faced, as well as the evaluation of the results we get from our predictive model. In the end, we summarize the pros and cons of our models and present future steps for a more complicated model.

Our Approach and Methods

In this section, we will describe the methodology used in our project. Since there are two major parts of the project, we will discuss the approaches to the two parts separately. In both parts, we used data from kaggle (https://www.kaggle.com/dgomonov/new-york-city-airbnb-open-data) which describe the factors related to the number of reviews per month in New York City.

a. Descriptive Data

When dealing with descriptive data, the first step for us is to transform the descriptive words to numerical data that would make sense to the computer program. For the original data, we choose X to be the column of 'name' of the airbnb and y to be the column of 'review_per_month' in the original data and convert them to numpy arrays. For data preprocessing for y, we just cast them to a type of numpy float. And for X, we used CountVectorizer from sklearn to perform tokenization which separates the sentences into a set of tokens. And then created a bag-of-word matrix of all the unique words in all names. In the bag-of-word created from CountVectorizer, we have pairs of sequence number and feature name, in this case, the word that appears in the names. Then we convert it to our feature vector X, which is a matrix of shape of number of names vs. number of words, where each row represents the name of an airbnb. After creating the bag of word feature vector X, we perform a train test split and set the test set to be ⅓ of the total number of samples.

To have some basic idea of how our data looks like, we first tried the linear regression model. As expected, the linear regression model poorly fit our data set with the average loss of 3.8979136227124014e+23 on the test set. After we looked closer to the distribution of the number of reviews, we chose to use binary classification where we set y to 1 if the number of reviews per month is more than 10. The intuition of choosing 10 is that we have noticed that y has a maximum of 58.5, a minimum of 0.72 and 10 lines on the top 1% of the distribution, and therefore we chose to use 10 to classify whether the airbnb is popular in NYC. Since our feature vector X is big and mostly zeros, we calculated the baseline for our accuracy on the training and test set which is 0.998145 and 0.997450 respectively.

When doing binary classification, we have tried 'logistic regression', 'support vector machine' as well as 'neural network'. Before trying those methods, we normalized Xtr and Xts. The first method being used is 'logistic regression', with solver = 'liblinear', and the time used for training the data set is being recorded. The reason is that after doing research, we found that 'liblinear' is the winner of ICML 2008 large-scale learning challenge. It applies L1 Regularization and it's recommended for solving large-scale classification problems. Then we tried using SVM with 'rbf' kernel to solve this problem using a nonlinear kernel to see if it would give us a better result. The accuracy on training and test set is calculated and time for training the data set is recorded as well. We also used a Neural Network with one hidden layer. We used 'relu' as the activation function for the hidden layer and 'softmax' for the output layer. After performing nonlinear activation, we also have a regularization to prevent overfitting by ignoring randomly selected neurons during training at 50% chance. To do so, we added a dropout layer and set the probability to 0.5. To compile the model we chose 'sparse_categorical_crossentropy' as loss function, 'adam' as optimizer, and set the validation set to Xts and yts which are consistent with data

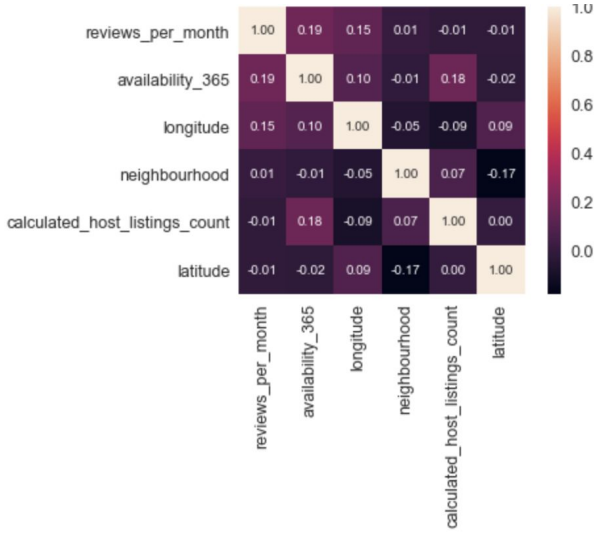
used previously for logistic regression and svm. At the end, we also evaluate the accuracy of the training and test set.

b. Numerical Data

Numerical data includes the input columns 'neighbourhood_group', 'neighborhood', 'latitude', 'longitude', 'room_type', 'price', 'minimum_nights', 'calculated_host_listings_count', 'availability_365', 'number_of_review'. We used multi-linear regression and neural networks to predict the number of reviews per month. The motivation behind two methods was that we wanted to use the simplest multi-linear model to do a general prediction first, and later to compare its performance with a complicated neural network model- Keras. Keras is known to be a powerful library backed by Google and other leading tech companies for developing deep learning models. It can be easily deployed across a wide range of platforms for deep learning.

First, we transformed some text data that are categorized as numerical data, to be fully numerical. For data in 'neighbourhood group' and 'neighbourhood' columns, we converted each string to its ascii value. For example, 'Brooklyn' is converted to a sequence of integers '66 114 111 111 107 108 121 110', where 'B' corresponds to 66, and rest characters follow the same logic. This ensures each region would be presented in a unique integer number. For data in the 'room_type' column, there are only three options. Therefore, we converted 'Private room' to 0, 'Entire home/apt' to 1, and 'Others' to 2. After converting all text to numbers, we seperated whole data to ⅔ training set and ⅓ test set.

A heatmap was used to investigate what are the top features that contribute the most to a rental place's popularity among all numerical features. We extracted 5 most correlated features as figure below:



The plot shows that 'availability_365' was the most important feature to 'reviews_per_month' as it has the highest correlation (0.19) among other numerical features. Longitude and latitude also contribute greatly to the popularity, and are considered together since they represent geographic location in New York City. Neighbourhood is somewhat important, and the rest features as less significant.

After grasping some insights on our data, we created a multi-linear regression model trying to find the optimal beta to minimize the squared loss. After training our model on the training set, we predicted the output *yhat* and compared *yhat* with the actual output *y* in our training set to compute the loss. This step is repeated for the test set.

Next, we standard-scaled the data before running Keras model. We created a sequential model with four layers, and compiled the model:

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 10)                100
_________________________________________________________________
dense_2 (Dense)              (None, 30)                330
_________________________________________________________________
dense_3 (Dense)              (None, 40)                1240
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 41
=================================================================
Total params: 1,711
Trainable params: 1,711
Non-trainable params: 0
```

Then, we feed training data to the model with epochs=150, batch_size=32. The model trained on 26010 samples, validated on 12811 samples. Finally, we evaluated the model's performance by calculating accuracy and loss.

Our Results

In this section we will discuss the performance of models we used against the baseline for both parts of our project.

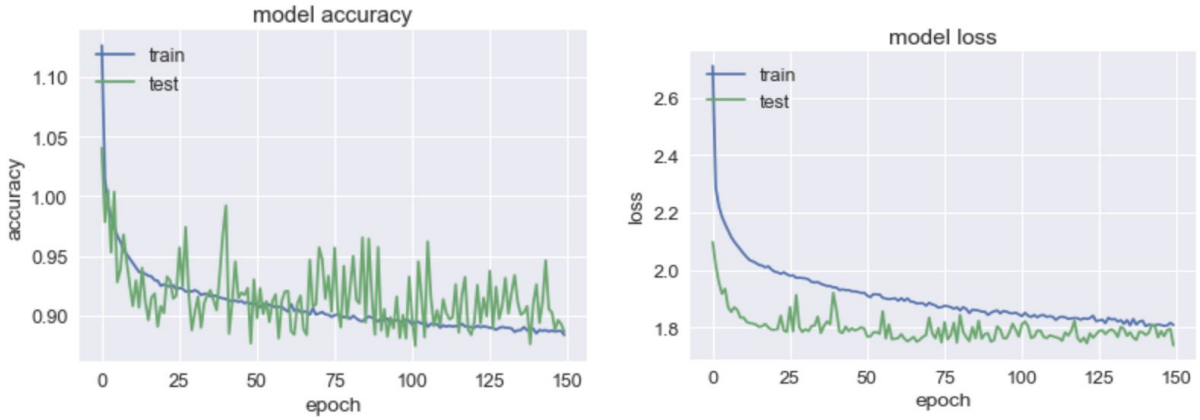
a. For word filtering, our final model performance is shown through the accuracy test, as the table below:

Method used	Accuracy on training set	Accuracy on test set	Time taken
baseline(all zeros)	0.998145	0.997450	/
Logistic Regression	0.998145	0.997450	0.9945271015167236
SVM	0.998261	0.997527	204.2654411792755
NN	0.999189	0.996214	606.9619624614716

From the chart above, we find that SVM gives the best prediction on the test set by training the data at a moderate amount of time. Since the neural network does not have a good performance as we expected, we tried to condense our feature vector to 500 most frequent words used for names using Tokenizer on Keras and setting the num_words to 500. We use the new X for the same neural network model and we got the accuracy of 0.997914 on the test set against the baseline of 0.997759 shown in the table below:

Method used	Accuracy on training set	Accuracy on test set
baseline(all zeros)	0.997991	0.997759
NN with max_words=500	0.999575	0.997914

b. For numerical data, multi-linear regression gives a loss of 4.03e+36 for the training set, and a loss of 6.72e+04 for the test set. Keras model performance is shown through the loss measure and accuracy test, as the figure below:



The left plot shows the model's accuracy, which is calculated through mean absolute error. The right plot shows loss value. This result could have some improvement, such as adding more layers or having a bigger dataset to feed the neural network.

Conclusion

The public Airbnb data has various attributes that have different degrees in affecting the rent popularity. The popularity is conveyed by the number of reviews per month from the dataset, where the higher number of reviews means more popularity of that rental property. Therefore, the number of reviews is y, which is the output our model tries to predict from input attributes. Dataset is splitted into ⅔ of training set and ⅓ of testing set. The input attributes are separated into descriptive ones, and numerical ones. Each type of attribute is computed through different models.

"Name" is the descriptive attribute where data is a composite of words that the hosts describe their places. A bag of words is created for each Airbnb name as input x. Our first approach to descriptive data was to build binary classification by defining y: The number of reviews per month higher than 10 is popular (1), otherwise unpopular (0). We run logistic regression, support vector machines and neural networks on training set Xtr and ytr, and test each model's performance by calculating its accuracy. From the above three approaches, we found Neural Network with top 500 features gives the best performance when testing.

Numerical attributes include 'neighbourhood group', 'neighbourhood','latitude','longitude','room type','price', 'minimum nights','availability 365'. First, we process some textual data to be fully numerical. Then we fit both a multi-linear regression and a neural network model on x training set and y training set. We measure the performance by calculating the square loss for multiple linear regression, and computing the mean absolute error for Keras model. We found the Keras model has a better performance as it gives a lower loss than multi linear regression model's loss.

If we have more time, we would try more sophisticated methods in Natural Language Processing on the descriptive attribute. We still want to predict if each textual word corresponds to a popular rent or to an unpopular rent. In order to simplify the problem we will split those into two categories: popular rent have a number of reviews >10; unpopular rent have a number of reviews <= 10. Raw textual "name" was loaded and is splitted into a positive part and a negative part based on our category judgement above. Next, we would do a data transformation for things like lowercasing, tokenizing the text, and removing useless stop words like 'the', 'a' ,and etc. Once we cleaned up the data, we could add sentiment analysis by using Vader from the NLTK module, which will return a neutrality score, a positivity score, a negativity score, an overall score for each token. We want to transform each word into numerical vectors using the word vectors so that we can use those vectors as training features. We can feed word tokens to the Doc2Vec model and then use a TF-IDF model to find the significance of each word in each "name" and filter some words. To recall, "name" is a sentence the host describes their home. We train the model by using a Random Forest classifier on our training set. The performance evaluation of our new model depends on whether our data is balanced or not. The possible methods could be receiver operating characteristic curve, or average precision.

One other question to consider if we have more time is looking at descriptive data and numerical data together and finding out how it affects the output popularity. In this project, we train them separately through different models. Therefore, it is vague to see which data has more weight in determining the popularity.