ORIE 4741: Project Midterm Report

Eric Lin (esl89), Alan Wu (ajw238), Ju Hee Lee (jl2755)

October 28, 2016

1 Introduction

In recent years, we've seen a dramatic increase in Airbnb listings as an alternative to hotels for travelers, due to their convenience and wider outreach. Hosts can turn the homes they own into an overnight stay for traveling guests, by setting a predetermined fee and rules. As the market for Airbnb listings grows, it is important to create a standardized model to predict a reason- able price for each listing, where reasonable is defined as a good compromise between the host and guest. In short, the question is How do we determine a good price for an Airbnb listing? This will help both parties alike, as hosts will not feel that they are charging less than they should for their listing to maximize profit, and guests will feel that they are paying an appropriate price, promoting an overall happier transaction.

2 Dealing With Data

2.1 Data Source

The dataset that we used is provided by Inside Airbnb (http://insideairbnb.com/get-the-data.html), which is a non-commercial set of data on Airbnb listings. Here, the data contains various metrics on listings in the area such as price per night, minimum nights, intended guests, bed and bathrooms, etc. In addition, beyond having information about the housing, there are also records of number of bookings, reviews, and other relevant information from guests.

2.2 Data Selection

Of the many cities listed on Airbnb's dataset, our team picked Los Angeles to fit a model on. LA is a popular destination for tourists, which means that there must be plenty of data for training and testing. It is also large in size, leading to higher potential for price and geographic variation which will be more interesting for us to analyze later on. With more variation among certain features, we could perform clustering algorithms to look for any patterns in our data set.

2.3 Data Cleaning

The original data set is, for lack of a better term, big and messy. A lot of columns have gaps in them, and the data is not presented in a form that can be easily consumed by statistical models. We attempted to clean the data under the following principles of turning everything into an integer, and removing unnecessary columns. As a group, we decided on these criteria when it came to removing columns, since the original data had a lot to begin with (94 features):

- Could this data be converted into a number easily? This meant removing text items (comments on reviews) and anything that was related to pictures, since we did not have any tools for analyzing such data.
- Is the data relevant to analysis at all? This prompted the removal for a lot of metadata (host name, home name, IDs, etc.) In addition, we removed everything that had to do with location since we had agreed on LA as our location already.

- Is the data feasible to process given its current state? Would we be able to include this within the feasibility scope of our project? Some data features had too many gaps to be valid as a feature, so we went ahead and removed those as well.

A finalized list of columns that we originally decided on can be found in our `scripts` directory, under `columns_to_keep.txt`, with about half the original features cut. From then on, we moved on to preprocess the data.

2.4 Data Preprocessing

Columns (features) that are numerically-valued we will leave as is. Features that are boolean (t or f) will be converted into 0 for false and 1 for true. There are several other kinds of features that will be engineered manually, for example `host_verifications` which is a list of the methods in which the host verified him or herself. We will either convert the list to a number representing the size, or a sum of weights where each type of verification is associated a weight.

In addition, there were a few feature columns that were nominal values, which we encoded using the one-hot encoding technique learned in class. This ended up creating a lot more features than not using it, so we will possibly revisit this later. Currently, we have 74 total features, with about 26000 data points.

2.5 Tools Used

For preprocessing, we used Python's pandas library to handle the data (see `cleanup.py`). Pandas is something that requires additional installation past the default python packages, for those interested in replicating our data cleanup process.

3 Preliminary Analysis

3.1 Linear Regression

The first model that we experimented with was the linear regression model. Let $X \in \mathbb{R}^{n \times d}$ be the training data which contains n listings with d features. Let $Y \in \mathbb{R}^d$ be the price for each listing in the training data. Let $w \in \mathbb{R}^d$ be the linear model coefficients computed by using ordinary least squares.

$$\vec{w} = \underset{w}{\operatorname{argmin}} \|Y - Xw\|$$

$$\text{estimated price} = w_0 + w_1 * x_1 + w_2 * x_2 + ... + w_d * x_d$$

3.2 Linear Regression with Feature Combination

The results of linear regression model were not very good, outputting extremely negative numbers for the estimated price of the houses in the training data. We decided to experiment with combining some of the features because we felt that some of the features were not very meaningful, but had extremely negative coefficients, which led to inaccurate predictions. By combining the features, we reduce the number of features and merge the ones that are related to each other.

For example, we combined the security deposit column with the cleaning fee column because they are both one time payments and can be treated as one initial fee. There are also several different features related to review scores, one for each of the following dimensions: cleanliness, check-in, location, etc. We created one column for the review scores by calculating a weighted sum of the different review scores.

With this method of feature engineering, our model improved slightly, but still has a lot of room for improvement.

4 Evaluation Method

4.1 Overfitting and Underfitting

After coming up with a few different models for our data, we may run into cases of overfitting or underfitting. Overfitting is when the model performs well on the training data, but does not generalize well to other data. Underfitting is when the model performs poorly on the training data, and also does not generalize well to other data. Both scenarios should not happen with well-fitted models. To detect overfitting, we can use cross validation to calculate the average validation error. There are several methods to reduce overfitting. The two methods that we are going to consider are *bootstrap aggregating* and *regularizers.* Underfitting means that there is high bias. For our project, we are unlikely to run into underfitting because our data set is quite large. In the case of underfitting, we could experiment with which features we are going to include for our model. The selection of features is important to come up with an optimal model.

4.2 Model Evaluation

At the current stage of the project, we will run our regression and generate a set of weights, and use these weights in order run predictions on our data. For now, we were thinking of using a very forgiving loss, such as Huber, in order to evaluate our model. This is because the range of our data is on an ordinal scale instead of being specific labels, so it will be very difficult for a model to successfully predict the price on the dot, as opposed to a classifier with simpler labels. Hence, we would prefer to use a loss function that is more forgiving of error, as it is more expected in this situation.

5 Future Work

Our main goal for this project is to create a model that determines a suitable price for an Airbnb listing, given its details. So far, we ran some preliminary analyses on our data after preprocessing. In the future We will use linear, quadratic, and cubic regression to come up with a few different possible models. After observing the coefficients of w from linear regression, we observed that some of the coefficients were very negative. One of our future plan is to keep the absolute values of the coefficients small by using ridge regression, or select a few important features to focus on by creating a sparse solution using lasso regression. The selection of features is also not final. We can try adding or removing some features and see how that affects the accuracy of our model. To measure the accuracy, we still have yet to implement cross validation. Once we implement that, we can use the validation error to determine whether there is overfitting.

5.1 Future Evaluation Goals

As our method of evaluation, we chose to use k-fold cross validation, with a default value of $k = 10$. We will split our training data into K equally sized subsamples. In each iteration for $i = 1..K$, we will train our model on $K - 1$ subsamples, which is equal to the entire training data minus the i-th subsample. The i-th subsample will be used as validation data to compute the accuracy of the model. The K results from the folds will be averaged to produce a single accuracy rate. One advantage of k-fold cross validation is that all the data points are used for both training and validation, and each observation is used for validation exactly once.