Charlie Denhart
DS4400
12/07/18

Final Paper

Link to code: https://github.com/cccdenhart/ds4400_final_project

Data Collection

The first step I took in this process was to collect my data. All data was retrieved from InsideAirbnb.com, which provides scraped Airbnb data from dozens of cities and over several years. Although I would have liked to use data from many, if not all, of the cities available, I decided to pull from Boston for now for simplicity. In order to retrieve all of the data, I generated URLs for each dataset by concatenating proper fragments of the dataset links on InsideAirbnb.com with the date of the scrape used. Since the scrapes used took place over several periods of time, each belonging to a separate CSV file, I aggregated all of the data collected into a single Pandas data frame.

Data Cleaning

Once collection was completed, I moved on to viewing and cleaning the data. There were several minor actions that were necessary. For all price values (nightly rates, weekly rates, monthly rates, security deposit, etc.) I removed dollar signs and converted the values to integer type. I conducted similar cleaning on all percentages (host acceptance rate and host response rate). For dates present, I simplified the format to simply containing year and month as that is all that is needed to uniquely identify the instance of a unit in time.

Regarding missing values, I simply dropped variables with significant numbers of missing values and then removed all rows with missing values for now. I will likely do some

imputation later if more observations are needed. Finally, I needed to deal with data stored as strings. For the purposes of this check-in, I will simply be running some regressions and attempting to classify the neighborhood of an Airbnb unit. For the regressions, therefore, I converted all categorical variables to N – 1 dummy variables per categorical variable. For the classification, I did the same, but for the neighborhood feature, I utilized Scikit-Learn's Label Encoder function to convert it to a numerical variable capable of being classified by machine learning algorithms.

Feature Engineering

I wanted to make use of some features that could not be used in their current format, so I did some feature engineering. I generated a feature for the number of amenities in a unit by counting the length of the list of amenities provided in the data. I also calculated the number of days as a host by subtracting a given host's start from the current date.

In addition to providing files with detailed features per unit, the site I collected data from also provided corresponding files listing the daily price and availability per unit. I used this information to derive features for occupancy rate and the number of price changes per unit. I took further advantage of this dataset by reshaping it into a form that provided more insight into how an Airbnb listing's performance responds to price changes. In order to do so, I identified all observations at which a price change occurred for each listing. I then used the observations in-between to calculate the average daily revenue between each price change. Using this extracted information, I was able to create a new data frame of price change observations with each unit containing features for the price change amount, the resulting average daily revenue change, the date, and the listing id.

Feature Selection

Despite narrowing down features in previous work, I was still left with an abundance of variables, many of which are likely irrelevant. Therefore, I decided to do some feature selection using PCA. I cut down the number of features to both 15 and 50 from the original 126 in order to test different impacts on model performance. I also tried using the 'SelectKBest' tool from the Scikit-Learn feature selection module and hand-picking certain features for different models in order to obtain better and more efficient results.

Data Exploration

In order to get a better sense of the data being used, I also did some data exploration. In order to get a sense of how data was correlated, I conducted a Variance Inflation Factor (VIF) test on my independent variables. I found that several variables, including different review scores and the bed type of a unit, exhibited high multicollinearity, so I dropped them from the model. To understand how different variables were distributed, I plotted several histograms. I also wanted to get a better view of how different variables were related so I plotted a few scatter and bar plots.

I also wanted to see how the numbers of Airbnb units laid out in different regions across the city, so I calculated density values for each zip code. Then, via the Mapbox GL JavaScript Library, I plotted the zip code regions on a map and filled them in appropriately with their density values using a GeoJSON file I constructed. This map is available at cccdenhart.github.io/apt-visuals.

Modeling

I wanted to see how well I could predict the performance of different units, so I regressed on a few different metrics including occupancy rate, number of reviews, and number of price changes. The regression for occupancy rate was successful, with an adjusted r-squared value of around 0.7 - 0.8 for several specifications. The number of reviews and number of price change regressions were slightly less encouraging, however, with adjusted r-squared values hovering around 0.3 - 0.5.

I was also curious about how well I could predict the review scores of a listing. Since the different review features were highly correlated, I averaged them into a single feature: average_score. I trained three different models in order to try and predict the review score: a multiple linear regression, a multiple linear regression using ridge regularization, and a support-vector regression using a radial kernel. Results, unfortunately, were poor. R-squared values for the two linear models were both around 0.14 while the support-vector regression was around 0.3. To investigate the reason for this, I plotted the distribution of the average review score and found that it was very left skewed. I tried correcting this by performing a log-transform but obtained a similar distribution. Since the labels were clumped and exhibited little variance, I found it acceptable, though disappointing, for the models to perform so poorly.

I next tried to investigate how well I could predict whether or not a host was a professional. Previous literature (Agent Behavior in the Sharing Economy: Evidence From Airbnb, 2015) labels professional hosts as those with greater than one unit listed on Airbnb, so I found it justifiable to make the same assumption. I engineered a feature for professional hosts simply by taking advantage of the host listing count already provided. In order to ensure my models were predicting based on legitimate listing features, I removed all references to the host's

listing count from the dataset. I then used a logistic regression, a decision tree, and AdaBoost to attempt to classify hosts. Results were satisfactory, with the decision tree obtaining the highest accuracy at 90 percent. Upon further inspection, however, I noticed that the logistic regression results were a bit odd; accuracy was moderate, but the AUC score was 0.5. I plotted the raw probabilities generated by the logistic regression model on my testing data and found that all probabilities were greater than 0.5, resulting in all classifications being 1.

I next attempted to classify the neighborhood that an Airbnb listing resided in. I used k-nearest-neighbors, Naïve-Bayes, and a random forest in order to make my predictions. The outcome was descent as a whole, with the random forest topping accuracy at 91 percent. Naïve-Bayes, however, was notably worse at just 11 percent. This could possibly be attributed to the fact that many of the different features used are highly correlated, foiling the assumption made by Naïve-Bayes that features are independent.

Finally, I attempted the most interesting portion of my project, which was to predict whether a price change for a given listing at a particular time will affect the listing performance positively, negatively, or not at all. In order to do so, I first pulled in the data that I generated for listing price changes. I then added the feature 'days_since', which allowed models to perceive how performance was affected based on how many days into the year the price change occurred at. I then joined the price change data with all of the listing characteristic data, resulting in a 5 million observation dataset with 43 features. Once data was prepared by converting string variables to dummy variables and selecting the top 20 features, I trained three models on my training data: k-nearest-neighbors, a random forest, and a feed-forward neural network. Results were mediocre. K-nearest-neighbors performed the highest with 64 percent accuracy. Disappointingly, however, the neural network performed the worst at just 21 percent accuracy.

This could possibly be attributed to poor implementation of the network, such as not enough layers.

Conclusion

The project was ultimately a success as I was able to train a few well performing models for different listing characteristics and was able to forecast price change affects with moderate success. I hope to further improve this work, however, by trying to increase accuracy levels in the price change models. If I get the performance of these models to a sufficient level, I would like to deploy them to a web site so that anyone with an Airbnb listing can input their property characteristics and experiment with how different price changes might impact their daily revenues.