Jiajing Li, Ari An

Department of Statistics, Columbia University

STATUN3106: Applied Data Mining

Professor Wayne Lee

May 7, 2022

Introduction

The Superhost program at Airbnb gives the host more visibility, earning potential, and exclusive rewards. The requirements include 4.8+ overall rating, 10+ stays, <1% cancellation rate in the past year, and respond to 90% of new messages within 24 hours. However, the original intention of Airbnb is to promote more diverse traveling experience.

From Airbnb's 10K report: instead of traveling like tourists and feeling like outsiders, guests on Airbnb can stay in neighborhoods where people live, have authentic experiences, live like locals, and spend time with locals in approximately 100,000 cities around the world. From our point of view, the hard criteria rather promotes professional management teams who have more labor to respond to messages, decorate and clean the houses and even look for suitable places for their properties. However, the uniqueness of listings on the platform will be negatively impacted if the number of individual hosts does not grow at the same rate. Therefore, in this project, we are promoting a new way of identifying "Honest Host" whose reviews given by residents are matching with the host's description, using text data mining and clustering.

Potential Audience

By identifying "Honest Host," Airbnb can in turn support individual hosts who are willing to dedicate leisure time to decorate their houses, manage their properties, and interact with interested residents. Living with this kind of hosts, guests will have the chance to feel like locals and interact with local residents. This undoubtedly gives a unique experience to people who choose Airbnb over traditional hotels. Therefore, our project can benefit Airbnb by encouraging more individual, diverse hosts to join. The results might be valuable for them to decide for new marketing campaign, or have more personalized marketing campaign for different host group.

Datasets

1. Data source and introduction

My data comes from Insideairbnb.com, which is an investigatory website launched by Murray Cox in 2016. The site was originally established by Murray Cox to identify possible illegal behaviors in NYC, where the same apartment appeared in different listings. The site uses the following Open Source technologies: D3, Bootstrap, Python, PostgreSQL, and Google Fonts, and is served by an Amazon S3 “bucket". Nowadays, Insideairbnb manages the data and updates the newly scraped data of dozens of cities and countries around the world quarterly for public uses.

For this project, I used two datasets: review.csv and listings-details.csv. “Listing-details” has dimension 38277*74, where each row represent a listing on airbnb and features include information of the listing, including bedroom and bathroom number, property type, and price, host information, and neighborhood characteristics. Wording information such as amenities, general description about renting place, neighborhood overview, and host's personal information was concatenated together to gather information given by the host.

“Review” has dimension 891964*6, where each row represents a comment to one listing. The columns includes listing_id, review_id, reviewer_id, date of review, and the comments. By joining these two datasets together, we connect the comments with the individual listings and make comparison between the description given by the host and the reviews given by past residents.

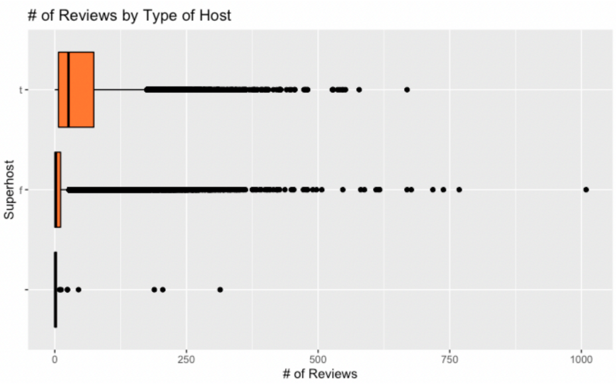
2. Data quality issues

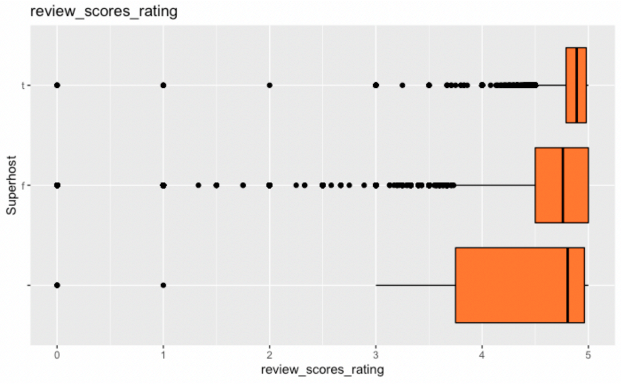
Features like “listing_url", “picture_url", “host_url", “host_thumbnail_url", “host_picture_url", and “calendar_last_scraped" were filtered out based on domain knowledge. By finding column-wise count of the NA values, I filtered out “bathrooms" and “calendar_updated" which consists of only NAs.

Furthermore, by checking the number of comments associated with each listing, we filtered out listings with at least 5 comments.

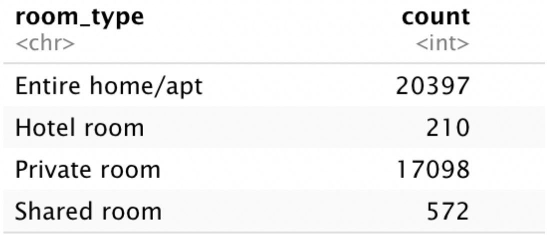
3. Data summary and Visualization

Superhost characteristics:



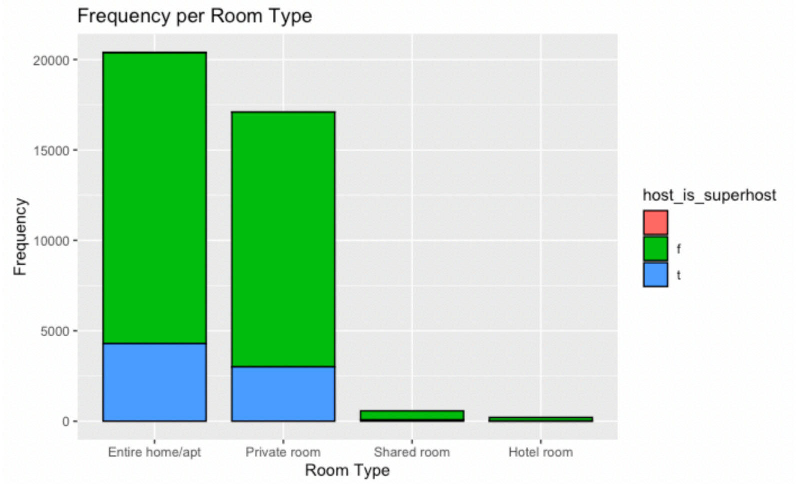


By the two boxplots above, it’s easy to see that if the host has a “SuperHost” badge, he or she will receive a significantly larger amount of reviews as well as higher review scores, even with a much lower variance. This coincides with what's written on the official website: The Superhost program celebrates and rewards Airbnb's top-rated and most experienced hosts.



room_type <chr>	count <int>
Entire home/apt	20397
Hotel room	210
Private room	17098
Shared room	572

In addition, most of the Airbnb listings are entire home/apt or private room. This is true for both Superhost and regular hosts.



Algorithm

1. Data cleaning and transformation

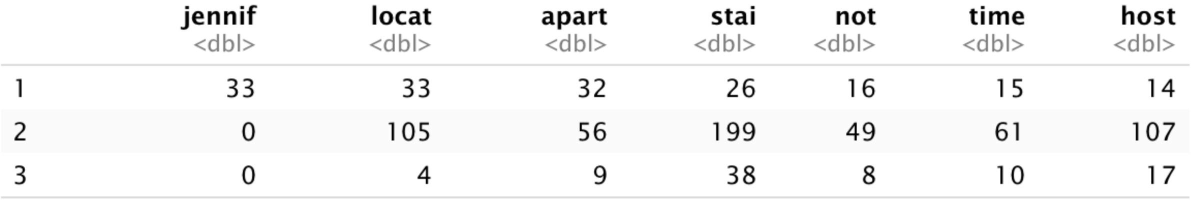
Information of each listings from both the hosts and residents was concatenated by each listing. We collected all descriptions written by the hosts, including description, amenities, neighborhood overview, and host about. For the information from residents of each listings, we concatenate all comments of the listing as the "documents" for future text data mining.

To begin with, we aim to generate a combination of bigrams and trigrams, in replace of unigrams. This is because unigrams may perform a false indication. As an example, if the actural wording is "not clean", unigrams will store "not" and "clean" separately as two words, and therefore the occurrence of "not clean" will be mistakenly counted as "clean". To avoid these build-in errors, we make attempt to store n-gram tokens. That is, we place our eyes on bigrams (2 consecutive words) and trigrams (3 consecutive words). Also, we ace our focus on the descriptive tokens, such as "air conditioner" of "tidy place". We exempt stop words (such as "am", "would") and noisy words (such as "<br>") in the generation stage and perform stemming and lemmatization to avoid repeated words under different forms.

2. Unigram, Bigram, and trigram generation

2.1 Unigrams

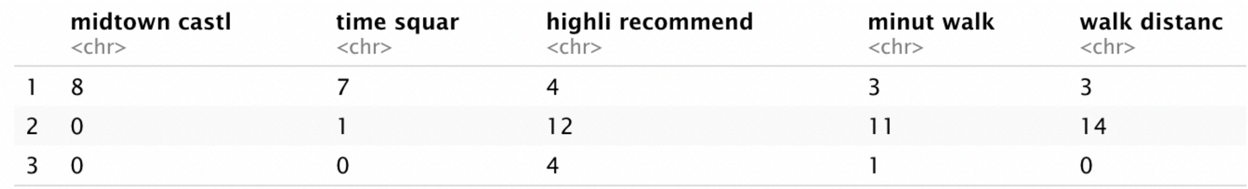
We extract word tokens from residents' reviews to get the word counts of each token.



	jennif	locat	apart	stai	not	time	host
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	33	33	32	26	16	15	14
2	0	105	56	199	49	61	107
3	0	4	9	38	8	10	17

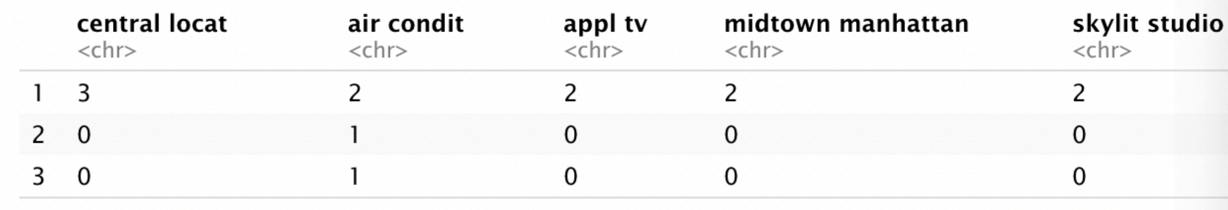
2.2 Bigrams

Using descriptions from hosts' side, we apply data cleaning and convert the wording description to bigrams. Notice that we lower case all words, remove html tags, correct the misspelled words, remove stopwords, and perform stemming to avoid repeated words with different forms.



	midtown castl	time squar	highli recommend	minut walk	walk distanc
	<chr>	<chr>	<chr>	<chr>	<chr>
1	8	7	4	3	3
2	0	1	12	11	14
3	0	0	4	1	0

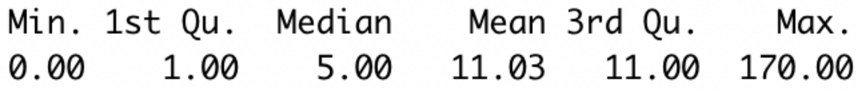
Similarly, same data cleaning process was applied to the review data.



	central locat	air condit	appl tv	midtown manhattan	skylit studio
	<chr>	<chr>	<chr>	<chr>	<chr>
1	3	2	2	2	2
2	0	1	0	0	0
3	0	1	0	0	0

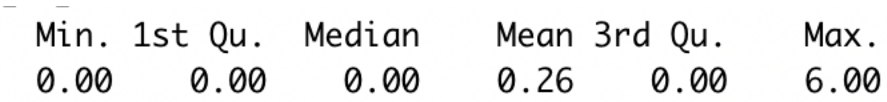
Then, we find intersect bigram tokens from descriptions(hosts) and reviews(residents) and create new dataframe for both sides where the columns are limited to the intersect tokens. Consider the cases that some hosts may have fewer comments than other hosts due to a shorter renting period. That is, the counts of token occurrence are subject to the number of comments, and may not fully imply residents' evaluation on the hosts' honesty. However, we potentially suspect that the hosts with more comments are more likely to have an accurate evaluation on their honesty. To be more specific, if an Airbnb host aims to be elected as an "Honest Host," aside from providing honest description, they also need to offer more renting experiences and gather as much comments as possible. The if_occurrence table summarizes the count of each bigram token that occurs in the selections of comments for at least one time.

Matched score was calculated based on the following criteria: If one word occurs in both review and description, then the corresponding entry in matched_df is 1. In all other cases, the element in matched_df is 0, representing a match failure. A new column called matched_count was computed to count the number of matched cases in each row. A summary statistics of matched_count of bigram is represented below.



```
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.00    1.00    5.00   11.03   11.00  170.00
```

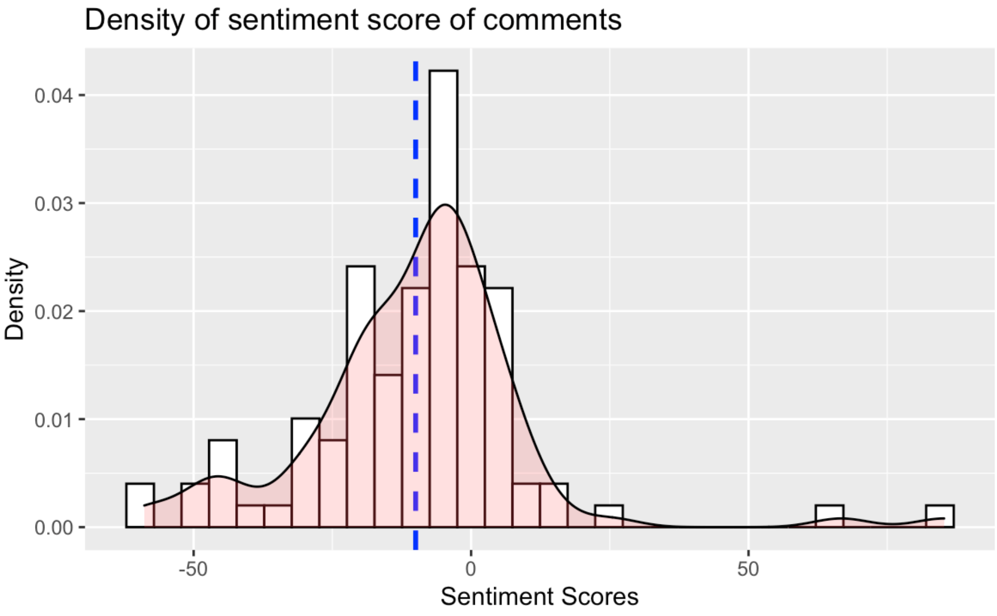
A similar procedure was used to generate trigram by using three consecutive words. Due to the nature of trigrams, there are much less co-occurence of the trigrams in both the review and description data frame. The largest number of matched count is now 6 instead of 170 in the previous case. A summary statistics of matched_count of trigram is represented below:



```
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.00    0.00    0.00    0.26    0.00    6.00
```

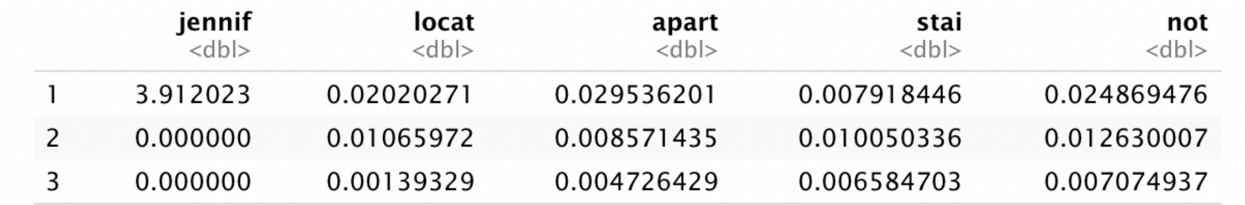
3. Sentiment Analysis

To supplement our analysis, we performed sentiment analysis on residents' comments for each host. Higher score indicates positive sentiment and lower score indicated negative sentiment. The median score is 0.3798. We then normalize the sentiment score based on the median score. The rule is: senti_score = (score - median) / (3rd quantile - 1st quantile). The distribution of sentiment scores after normalization is shown below:



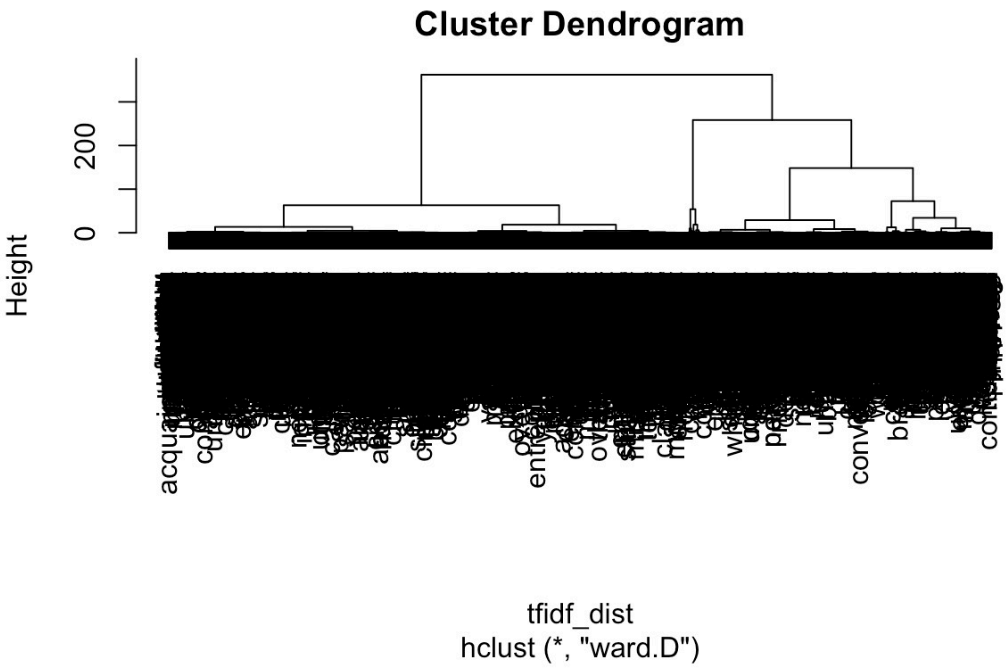
4. Clustering

Then, we used term frequency–inverse document frequency to calculate how important a word is to a document in a collection. Term frequency tf(t,d) is the relative frequency of term t within a document d and is calculated by the raw frequency divided by the raw frequency of the most frequently occurring term in the document. Inverse document frequency is a measure of how much information the word provides: whether it is common or rare across all documents. It is calculated by log(N/n_t), where N is the total number of documents in the collection, and n_t is the number of documents where the term appears. The final TF-IDF matrix was calculated by matrix multiplication tf * idf. High weights in TF-IDF are obtained by a high term frequency in a given document and a low document frequency of the term in a collection of documents. This tends to eliminate common terms. An extract is shown below:



	jennif <dbl>	locat <dbl>	apart <dbl>	stai <dbl>	not <dbl>
1	3.912023	0.02020271	0.029536201	0.007918446	0.024869476
2	0.000000	0.01065972	0.008571435	0.010050336	0.012630007
3	0.000000	0.00139329	0.004726429	0.006584703	0.007074937

Using the resulting TF-IDF, we cluster the comments and reviews using hierarchical clustering.



Clustering results were compared with the dummy variable SuperHost in the listings details dataset: True if the host is SuperHost, else False. By the confusion matrix, we can see that there is a discrepancy between SuperHost defined by Airbnb and Honest host defined by us.

Conclusion

In this project, we tried multiple approaches to tokenize the text data and compute the occurrence of unigram, bigram, and trigram. Although we did not get much information from the trigram text mining, there is a small matched count, indicating that there is a slight chance that both the description and comments of the listings have three consecutive words in common. In addition, by calculating the TF-IDF matrix of our common words, the hierarchical clustering gives an expected results of two clusterings. Compared to the original label, there is a discrepancy between the different definition of good hosts. What's more, based on sentimental analysis, we found out that most of the comments to the listings are positive.

Therefore, if Airbnb wants to promote a more diverse population in their host community, they should alter their current standard of SuperHost Badge or add some alternative badges that promote individual hosts instead of professional management team.

Due to computational constraint, we only run our algorithm on a small subset of the Airbnb listings. To further explore the similarity between description and review, we should perform a similar analysis on Airbnb listings in other major cities in the U.S. to see whether there's a similar discrepancy in SuperHost and Honest host.

Critique for Project by Drake Morey

The initial motivation is to create a stock strength indicator with large-cap, ETF, and individual stock that could increase investor returns and decrease uncertainty of return. The flow of the written report is easy to follow, and even people without much finance background would understand the analysis.

Dataset for stock market data, including Amazon, Nasdaq 100, S&P 500, and Consumer Discretionary Select Sector SPDR Fund, and Vanguard Consumer Discretionary was pulled from Yahoo Finance. In addition, a dataset of historical put/call ratios on the S&P 500 index (SPX) was acquired via CBOE.

He started by showing some introductory exploratory data analysis. For example, the individual daily percent returns of AMZN is highly correlated with various ETF. Two features are engineered to extract information for each ETF and individual sock. For the data mining part, a binary logistic regression was performed using AIC stepwise optimization to select the variables. A confusion matrix was computed to calculate the classification error of the daily percent change. He also tuned the cutoff instead of an arbitrary cutoff of 0.5 to improve his model accuracy. The metrics to examine his algorithm include recall, precision, and F1 score.

As the author mentioned that multiple features were researched to be included in the model but he only included engineered features that provides the strongest relationships. We would also be interested in seeing some other features and some potential reasons why certain features won't provide as significant results based on domain knowledge. In addition, given the strong correlation between AMZN and the ETFs, the provided algorithm could potentially increase investor return and decrease uncertainty by predicting daily percent change of returns. However, would the results changes if other individual stock or ETF are used in the model?