Better Reflection of Airbnb User Experience than Rating Engine via Machine Learning and Natural Language Processing

Riya Danait,[1, a)] Bill Zhuo,[2, b)] Ruijie Mao,[3, c)] and Jeffrey Chen[4, d)]

[1)] *University of Texas at Austin*

[2)] *University of Waterloo*

[3)] *University of Pennsylvania*

[4)] *University of California, Los Angeles*

(Dated: 9 September 2021)

CITADEL | CITADEL | Securities

TheDataOpen

[a)]Electronic mail: riyadanait@utexas.edu

[b)]Electronic mail: w3zhuo@uwaterloo.ca

[c)]Electronic mail: dreamfly@seas.upenn.edu

[d)]Electronic mail: jschen1217@gmail.com

EXECUTIVE SUMMARY

Airbnb has dominated the short-term rental market since its creation 13 years ago, providing an online reservation platform for hosts to accommodate guests with lodging. According to Airbnb's Resource Center, ratings are an essential aspect of business for a host — they can be a deciding factor in a future guest's decision to rent out the host's property. For this reason, a thorough understanding of review score ratings is crucial to help hosts improve their listings, allow guests to have a great experience, and increase the efficiency and success of the sharing economy.

However, the current rating system does not effectively present ratings as a reflection of a guest's experience. While this star-based rating system provides a quick way for guests to give feedback, it can be detrimental to a host when they receive an unexpectedly low rating. Similarly, the vast majority of users might assume that a 4-star rating is perfectly acceptable if their stay was good and do not feel the need to elaborate; what they don't see is the warnings that Airbnb sends to a host if their listing gets too many 4-star ratings. Moreover, different rating criteria of users make it even harder for the ratings to truly reflect user experiences. On the other hand, the discrete quality of the system coupled with the fact that many users provide a reliable succession of 5-star ratings without putting much thought into it can contribute to a false assessment of a listing.

Therefore, we explore the following as a guiding question in our analysis:

> **Can we create a sentiment index from Airbnb reviews that is efficient at reflecting the known features of a listing?**

Language style might be drastically different from region to region. Answering this question can help us identify the second-order relationship between user review comments and Airbnb's listing quality. This can help Airbnb hosts provide a better experience for future guests, as they will be provided with a more objective picture of user feeling to their listing; likewise, future guests will have a better understanding of a listing's past usage. In the sections that follow, we will show the procedure we took to preprocess the data and conduct feature engineering, as well as how we developed a new rating criterion, and performed hypothesis testing using both supervised and unsupervised algorithms.

Based on our results and as shown in Figure 1, we found our sentiment index to be a better reflection of the true user rating as it is less skewed and closer to a normal distribution compared to review score ratings.

TECHNICAL EXPOSITION

I. INTRODUCTION

In this study, we conduct a comparative analysis of machine learning methods and Natural Language Processing on the Airbnb datasets discussed in the following section, and provide a preliminary solution for the problem of a rating review system over the U.S. South region. We do so in the context of introducing a new approach to enhance house-sharing market efficiency.

II. PROBLEM STATEMENT

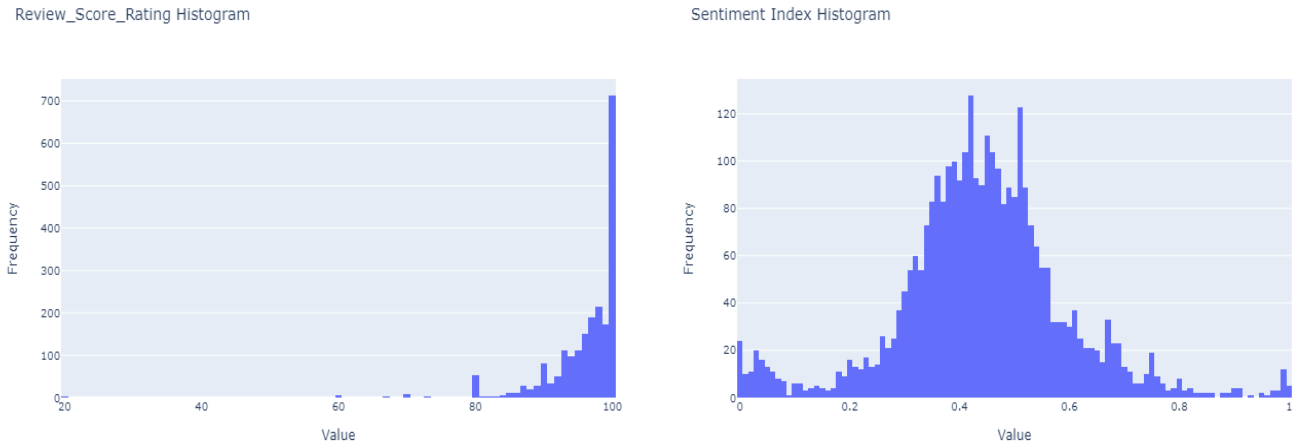
2.1 Motivation

As we investigated the rating system provided by Airbnb, we observed that most of the rating scores are extremely high. The reason is due to the discrete input value on the rating system, typically 1 to 5 stars, that reduces the sensitivity and dimension of the user's feedback (Figure 1a). The overall high rating score could not therefore provide comparison between each listing nor distinguish them from one to another. In addition, as Airbnb business grows, a popular marketing strategy of giving incentives to acquire high ratings is widely exploited (such as free snacks for a 5-star rating). The inflation of the rating system creates an information gap between the host and the user; therefore, we want to construct an index that could introduce efficiency and sensitivity to this objective.

In order to provide a better rating score engine, the most direct and intuitive response would be to use the user's comments. The comments provide realistic user experience on each listing and also reflect the user individual expectation and criteria. With alternative data becoming more and more popular, the idea of creating a sentiment indicator or index has been widely investigated. Therefore, we want to create a sentiment indicator based on the review comments. Effectively, we are trying to create an index that serves as a new response variable for both Supervised and Unsupervised learning.

2.2 How Informative are Review Comments?

We are trying to capture user "experience sentiment" where "Index" is a composite of each review comment according to `listing_id`. However, comments reflect both news and noise. We want to avoid overreacting to non-information and under reacting to genuine information. We also want the production of the sentiment index be robust and less biased. The challenge of creating the index is the Unstructured characteristic of the data; that is, the data is qualitative in nature, making it harder to analyze. The process of handling Unstructured textual data will be further discussed in the following section. The below figure (Figure 1b) shows the distribution of our sentiment index, a probability measurement of positive feedback from user. Compared to the Airbnb rating score distribution (Figure 1a), we introduce more sensitivity and normality into the rating engine.



(a) Airbnb Rating Score Distribution (b) Sentiment Index Distribution

FIG. 1: Distribution of Airbnb Score and Sentiment Score

III. METHODOLOGY

3.1 Data Sources and Preprocessing

In order to effectively create a sentiment index that better reflects a user's true rating for Airbnb listings in the U.S. South region, we leveraged several datasets, which include the Airbnb listings and calendar data, venues data, U.S. zip codes data, and reviews data from Inside Airbnb. As can be seen in Figure 2 below, there are clear clusters of Airbnb listings with review ratings in the U.S. South around the cities of Austin, Asheville, and New Orleans. Our sample begins in January 2015 and ends December 2018. The number of review comments is 352, 343 and after matching with listing id reduce to around 3,000. We restricted calendar, listings, and venues data to the same time periods. These datasets provided a robust analysis of Airbnb review ratings.

For the `listings.csv` file, we began by restricting listings to states in the U.S. South region. The `review_scores` features with *null* values were replaced with 0s, since *null* indicates the scores weren't calculated, which most likely means the listings are new and haven't hosted anyone yet. The same process was applied for features like `bedrooms`, `bathrooms`, and `beds` with `null` values, since it is again likely that the listing is new and there is no information about its amenities yet. Then, to properly extract features from the categorical data such as `cancellation_policy`, `property_type`, etc., we used one-hot encoding to produce a new variable and assign 0 or 1 for each unique categorical value in a column. The `amenities` column values were formatted differently; each row had a set of values. For this, we created a custom function to encode each value. After extracting the amenities and creating a flat list for all entries, we determined the count of each type of amenity, which resulted in 126 individual amenities. We chose all the amenities that appear in at least 1.25% of listings (68 amenities) — a trade-off between including at least 50% of all amenities and

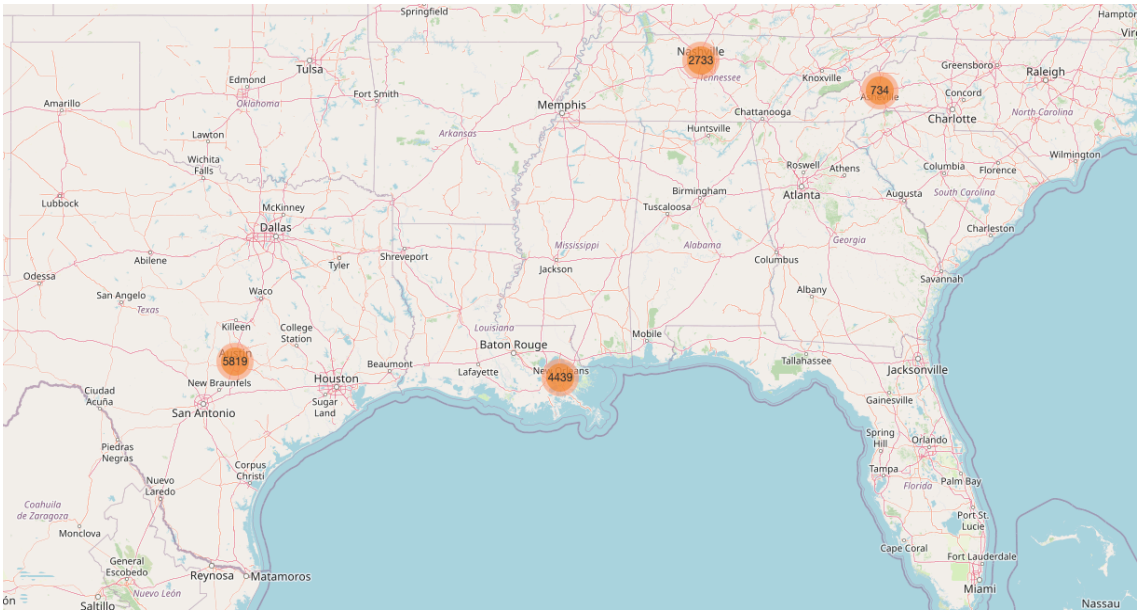


FIG. 2: Airbnb listings clusters on map of U.S. South region

unique amenities that didn't show up in many listings to determine if there was a significant effect on rating scores. Next, the amenities were split into groups based on type of amenity: kitchen, cleaning, safety, household, bedroom, electronics, and extra spaces. The counts for the amenities within each category for each listing were recorded, and these features were merged into the main dataset.

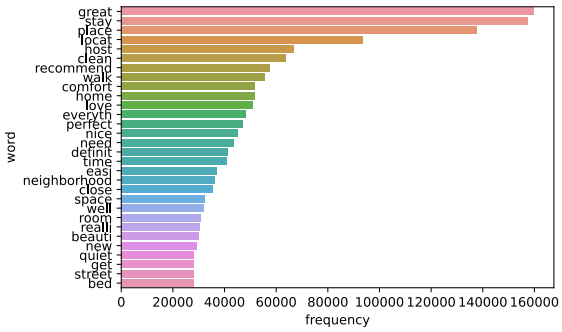
3.2 Sentiment Index with Natural Language Processing

Text is unstructured and the context matters. Our goal is to create an informative numerical signal based on the review comments. However, the noise is a big problem when trying to find meaningful classifications or signals of text data. For instance, there are many ways to express the same meaning and text does not equal to text. Therefore, we want to develop a workflow that might pick up 'positive'and 'negative'as most informative, without inference producing the opposite of the true meaning.

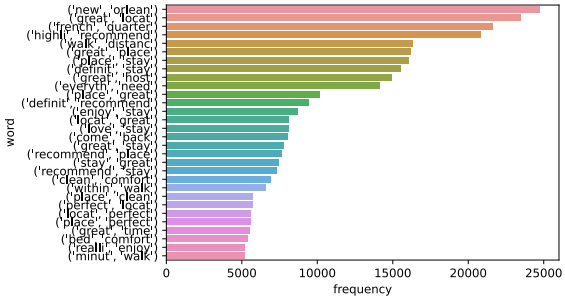
3.2.1 Interpreting Review Comments

To quantify the textual reviews data, we first tokenize the sentences into individual words as elements in string vector. We further clean the vector with 'stopwords', words to exclude from documents (such as 'a', 'the', etc) and 'stemming', to include the stem of a word (e.g. 'comfort'could be stemmed from 'comfortable'). Next, we create a Corpus and Document Term Matrix (DTM), which is the set of text we consider and a matrix of all unique words in each comment. The corpus gives the frequency of the word in that file and allows us to get a sense of our word frequency. The figure below shows top unigram, bigrams and trigrams in our review comments, and (3d) is the wordcloud (Figure 3). This give us the sense of our

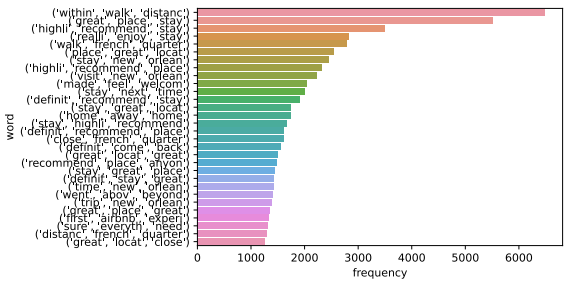
sentiment among the review comments. We can see that indeed most of the comments are positive, but more still are neutral words or without full content, that is, unstructured and sparse. Our goal is to create a sentiment indicator that contains sensitivity, that is, we want to observe the negative sentiment signals that are often overlooked. Based on the Airbnb rating score, we subset our review comments with a cutoff point of 50 as our negative dataset, and we also add common positive words in our 'stopwords 'to filter. The below figure (Figure 4) shows the word frequency of our negative dataset. As we can observe, it appears that a lot of points taken from the rating score are due to the host cancelling the user's reservation. We want to further exploit this result and try to surface any second level insight of the U.S. South region listings as well; however, the current textual analysis is still too sparse and contains too much noise. Are the current results meaningful or significant? Do the words we observed truly represent negative feedback? In order to answer these question and verify our idea, we need to dive deep into positive and negative feedback of the review comments. In the next section, we introduce 'Latent Dirichlet Allocation (LDA)'to help us identify our textual data in both quantitative and qualitative ways.



(a) Top Unigrams



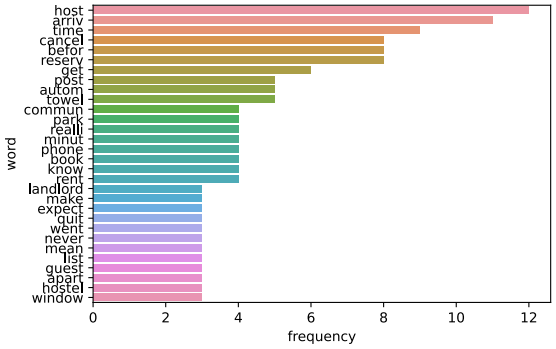
(b) Top Bigrams



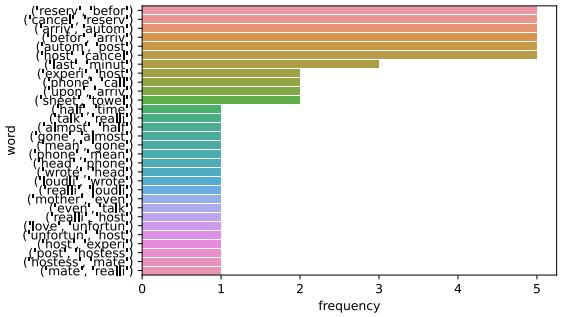
(c) Top Trigrams

(d) Word Cloud

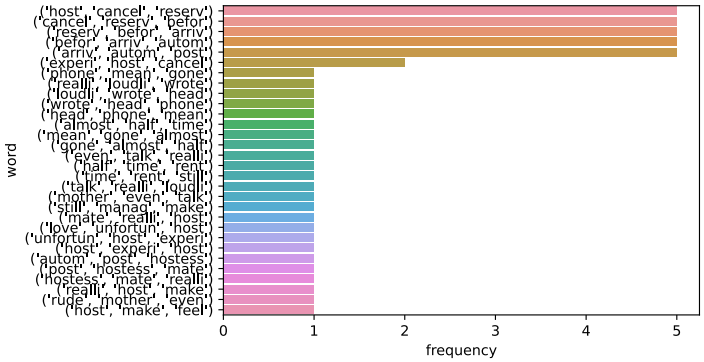
FIG. 3: Word Frequency



(a) Unigrams from Negative Sentiment Subset



(b) Bigrams from Negative Sentiment Subset



(c) Trigrams from Negative Sentiment Subset

FIG. 4: Negative Sentiment Subset Word Frequency

3.2.2 Topic Modeling and LDA

Text Topic Modelling is an unsupervised learning algorithm that tries to extract the main topics from a set of underlying documents (the Corpus). Here we are interested in review comments related to positive feedback and negative feedback of user's living experience. LDA helps us extract the text topics that best fit our corpus, and with topics in hand, we can then perform classification of each document to identify our target topics. In more detail, LDA represents documents as mixtures of topics that split out words with certain probabilities. It assumes that documents are produced in the following fashion. First pick the number of words on documents and then choose a topic of mixture for the document. Assuming this generative model for a collection of documents, LDA then tries to backtrack from the documents to find a set of topics that are likely to have generated the collection.

In the LDA model, we use Bayesian methods to estimate the probability. The priors over topics and words within each topic obey the Dirichlet distribution (we use the symmetric version of this distribution):

$$f(X_1, ..., X_k; \alpha) = \frac{1}{B(\alpha)} \prod_{i=1}^{k} X_i^{\alpha-1}, B(\alpha) = \frac{\prod_{i=1}^{k} F(\alpha)}{F(\prod_{i=1}^{k} \alpha)}$$

The prior distribution over topics is governed by α. The higher the α is, the more likely each document contains a mixture of most of the topics instead of any single topic. The prior distribution of words within each topic is also governed by a Dirichlet distribution as above. Finally we update beliefs based on Bayes' rule:

$$\text{Posterior} = \frac{\text{likelihood} \cdot \text{prior}}{\text{marginal likelihood}}$$

3.2.3 Mapping to Signal

We now find the main topics of our review as a whole. We use the python packages `gensim` and `spacy` for our LDA modeling. After initial tuning of our parameters, we divide our topics into 6 sections as Figure 5a shows. We can observe topics with positive feedback. Figure 5b displays the topic clustering of the negative subset we retrieve earlier. From this we can also estimate the topics and probabilities of each words. The table below shows the

Topics	Expression
Topic 1	0.058 'room'+ 0.033 'people'+ 0.027 'parking'+ 0.022'see'+ 0.017 'still'
Topic 1	0.034 'answer'+ 0.034 'respond'+ 0.034'call'+ 0.034'airbnb'+ 0.018 'bnb'
Topic 2	0.054'host'+ 0.040'reservation'+ 0.040'cancel'+ 0.040'arrival'+ 0.037'place'
Topic 3	0.039'phone'+ 0.023'rent'+ 0.023'time'+ 0.016'never'+ 0.016'suggest'
Topic 4	0.022'house'+ 0.012'quite'+ 0.012'attack'+ 0.012'overpriced'+ 0.012'toilet'
Topic 5	0.023'towel'+ 0.018 'hostel'+ 0.018'time'+ 0.012'make'+ 0.012'sheet'

TABLE I: Probability Constituent of Topics

We then define 'Goodfeedback'and 'Badfeedback'by we taking the probability of each words into our DTM and creating an weighted average counting for positive word and negative word. In addition, we add some bias words to each counting, such as 'rude', 'best'type of extreme words to create impact. Lastly, we simply create our sentiment index using probability measurements as follows:

$$\frac{\text{Goodfeedback}}{(\text{Goodfeedback} + \text{Badfeedback})}$$

We will further verify our idea and setting in the Hypothesis Testing Section later.

(a) Topics of All

(b) Topics of Negative Subset

3.3 Feature Engineering

For the purpose of better predicting the results, we explored the possibility of inventing new features more scientifically correlated with the rating scores. By applying these features, we can predict rating scores at a higher confidence and more accurately reflect the true user rating of each Airbnb listing.

3.3.1 Time Series

We believed occupational rates (how many days in a week/month that the housing are full) are an intuitive indicator of the performance of the Airbnb. If a certain housing is popular, it is natural that its occupational rates are high. The performance of housing is strongly correlated with the user-experience. Therefore, occupational rates can be an important factor that contributes to the high review rates. Also, the change of occupational rate over time is another indicator for the rating. Based on the assumption that users will check the ratings to make their decisions while choosing housing, the drop of occupational rate is an excellent indicator of bad user experience. So, we created a new feature based on change of the occupational rates of each housing compared to the previous unit of time (week and month), and their standard deviations.

We build the feature by applying *Modified Moving Average model*:

$$y_t = \mathrm{c} + \varepsilon_t + \theta_1 \varepsilon_t \ ...$$

But instead to predict the future term, we apply this model to predict the rating by incorporating it into Linear/Logistic Regression. Based on these two criteria, we can explore the

possibility of using time series of availability rates to reflect the true user rating.

3.3.2 Venues

The location of housing is another important factor contributing to user experience. The surrounding highly-rated restaurants can positively affect the rating. Therefore, we investigated the `venues.csv` file and calculated the relative distance to the Airbnb. We calculated the straight line distance through the longitude and latitude of Airbnb and highly-rated restaurants by applying the Haversine Formula. We filtered out those restaurants rated below 3 since low-rated restaurants may not contribute much to the user rating. We considered convenience to be the first reason why a restaurant can contribute to the user rating, so we filtered out restaurants that are farther than 5 KM to the Airbnb since it is not walking distance.

To more precisely find the walking distance it takes from the Airbnb to the restaurant, we applied the GoogleMAP API. By assuming that each person's walking speed is constant regardless of the distance, the walking time is directly proportional to the walking distance. By applying the discount utility model[3], we estimate discounted rating by applying the formula:

$$V_t = e^{\beta t} \times V_0$$

where V_t is the discounted rate over time, t is the walking time to the restaurants and V_0 is the rating.

If there are multiple highly-rated restaurants nearby, according to the utility discount model, the increase of the favor will decrease exponentially. Thus, the total favor is

$$\sum \beta^t \times u(X_t)$$

which is approximately

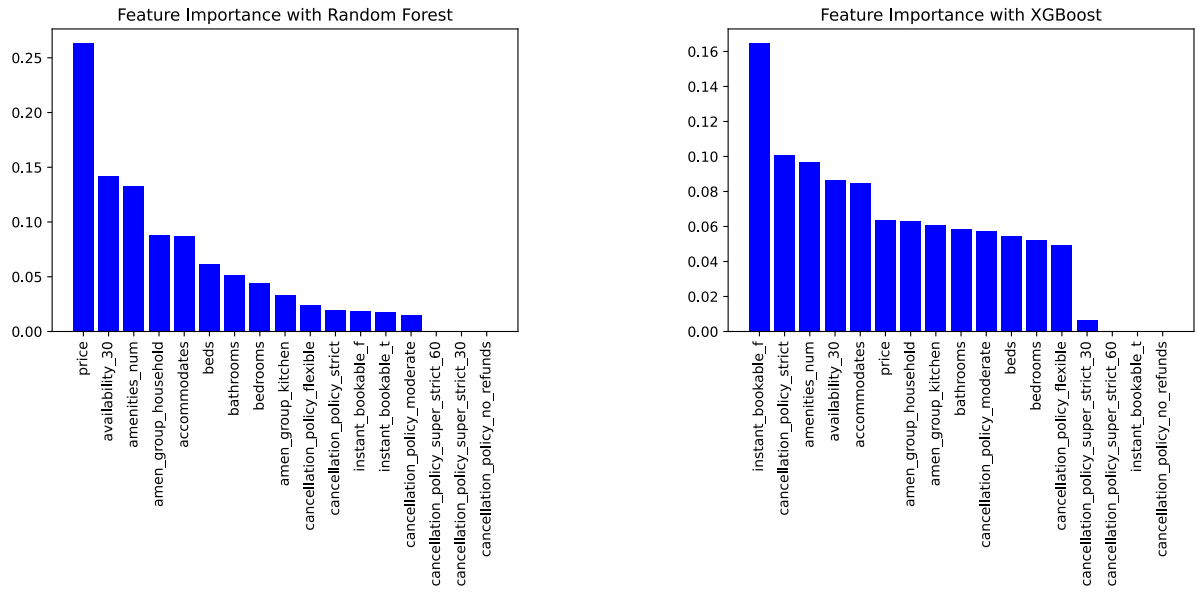
$$\frac{1}{\beta} - \beta^{\sum u(X_t)}$$

where β is the discount factor (between 0 and 1) and $u(X_t)$ is the utility of t^{th} highly-rated restaurant. Based on this formula, we generate a new feature called discounted utility of nearby restaurants to help us predict the user experience.

3.4 Feature Selection

We first started with training two models, a RandomForestRegressor and XGBoost, to predict `review_scores_rating` from the preprocessed `listings` data. We decided to choose these decision tree methods because they are ideal for medium-structured discrete tabular data. A benefit of using ensembles of decision tree methods like random forest and gradient boosting is that they can automatically provide estimates of feature importance from a trained predictive model. This is because after the trees are constructed, it is straightforward

to retrieve importance scores for each attribute. The importance is calculated explicitly for a single decision tree by the amount that each attribute split point improves the performance measure, weighted by the number of observations the node is responsible for. The feature importance are then averaged across all the decision trees within the model. In Figure 6 below, we compare the feature importance for the 17 features used in our regression models.



(a) Random Forest

(b) XGBoost

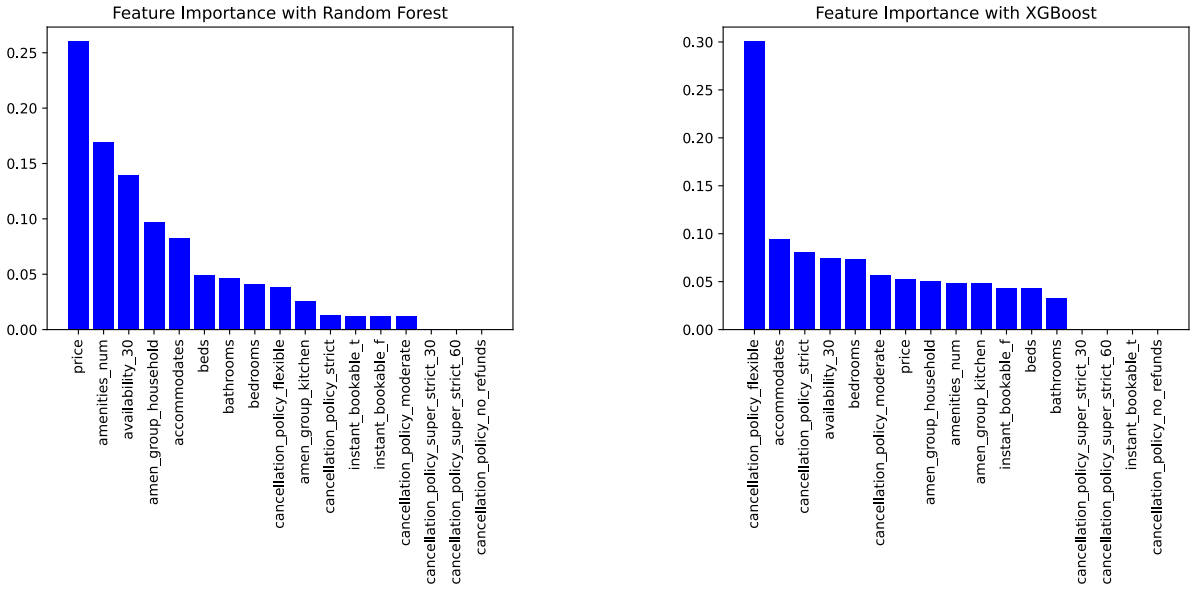
FIG. 6: Feature Importance Index for Random Forest and XGBoost to Predict `review_score_rating`

Next, we again ran random forest and gradient boosting to predict our new index, `feedback_score` from the `listings` data. Figure 7 below shows the feature importances from these models.

While there is not a significant difference in the importance of the features selected for the Random Forest regression model, XGBoost feature selection was significantly different for predicting the feedback score index developed from the review comments.

3.5 Hypothesis Testing on Index Quality

After obtaining our sentiment index and a set of features that can well describe each listing's quality, we need to perform hypothesis testing on this index quality. To do this, we have two approaches. The first one is to use popular supervised machine learning models and perform regressions between the defined listing features and our index and see how well the prediction distribution matches the index we create. The second approach is more qualitative. Using unsupervised clustering algorithms, we aim to observe if the index values are consistent with obvious clusters. The null hypothesis is that this index can well describe the listings quality.



(a) Random Forest (b) XGBoost

FIG. 7: Feature Importance Index for Random Forest and XGBoost to Predict `feedback_score`

3.5.1 Supervised Model Testing

To further quantify our hypothesis testing, we define a discrepancy measure as the root mean squared error (RMSE) between the index test set and the predictions provided by various models, including RandomForestRegressor (RF), XGBoost (XGB), support vector machine regression (SVR), and multi-linear regression (MLR). This set of predictive models are selected to perform this test since they are competent in different types of prediction problems. A wider range of predictive models can indirectly make our testing power strong and more holistic. After tuning the hyperparameters manually and through search processes, like Bayesian optimization[1], we have the following RMSE result in Table II.

Model	RF	XGB	SVR	MLR
RMSE	0.052	0.136	0.129	0.143

TABLE II: RMSE on various testing models against the index

By checking the relative frequency histogram between the prediction and the index created (Figure 8), we note that tree-based models (RF and XGB) seem to be able to fit to this index fairly well.

FIG. 8: Distribution comparisons across various models fitted

This reassures our feature selection process using these two models while all models' predictions seem to have a large portion of overlap with the index, especially around the peaks. We understand that RMSE might not be as exactly interpretable as a p-value in traditional statistical tests, but it can be interpreted as the percentage error that exists between index and the prediction given that our index has been normalized to be between 0 and 1. Based on this lower range of percentage error outputs and reinforcements from the comparative distribution graphs, we believe our current index has significance in reflecting the true quality of each listing. Throughout this hypothesis testing process, we have rejected the null hypothesis multiple times and improved our index modeling along the way until achieving the desired results.

3.5.2 Unsupervised Model Testing

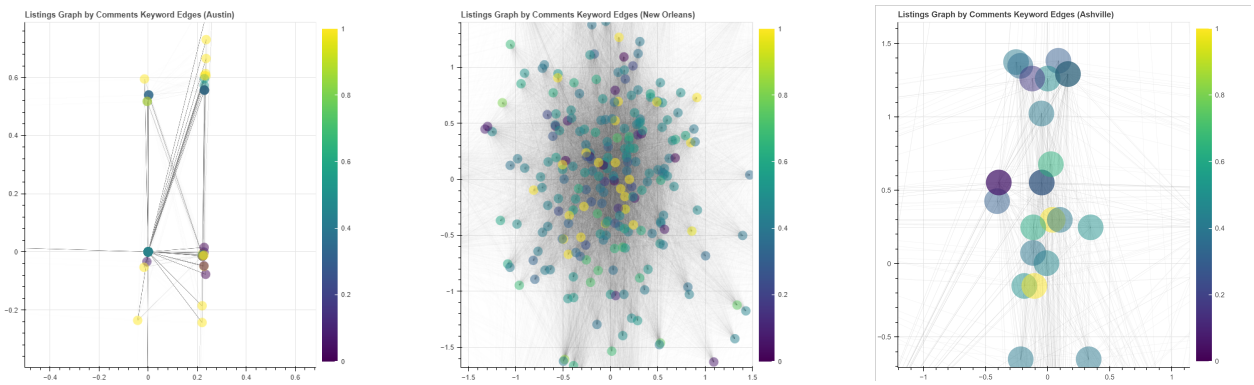
To test this index's quality from other perspective, we created an incidence matrix that tracks the frequency of each keyword used in the creation of our index. We call this matrix D shown in Figure 9.

$$
\begin{array}{c}
\\
\text{Listing 1} \\
\text{Listing 2} \\
\\
\text{Listing N}
\end{array}
\begin{array}{c}
\begin{array}{ccccc}
\text{Word 1} & \text{Word 2} & \text{Word 3} & & \text{Word M}
\end{array} \\
\begin{bmatrix}
1 & 0 & 5 & \cdots & 0 \\
0 & 0 & 1 & \cdots & 2 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 10 & \cdots & 1
\end{bmatrix}
\end{array}
$$

FIG. 9: Keyword incident matrix illustration

To create a graph from this incident matrix D, we need adjacency matrix A, which can be efficiently computed using $A = DD^{\top}$ as a common graph theory trick. We proceed to

perform two types of clustering. First one is spectral graph clustering, which is a common technique inspired by singular value decomposition (SVD). Without diving into the details, it provides us the number of clusters based on the multiplicity of the eigenvalue $\lambda = 0$, which indicates partitioned subgraphs (potential clusters). We performed this analysis on three samples from Austin, New Orleans, and Ashville shown in Figure 10. We note that



(a) Spectral clustered graph (Austin)

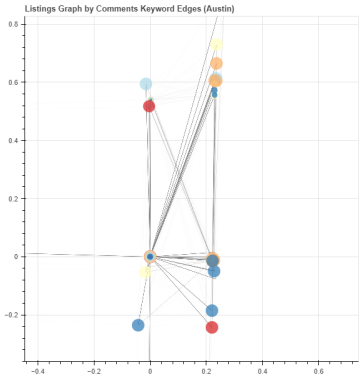
(b) Spectral clustered graph (New Orleans)

(c) Spectral clustered graph (Ashville)

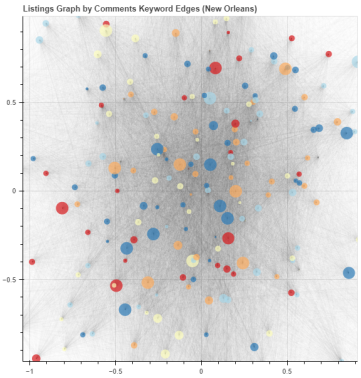
FIG. 10: Spectral clustering graphs with index value coloring

this clustering method has different effects on different region's listings. In particular, in Austin, we have have very clear clusters shown in 10a. Their coloring also seems concentrated around the same color range this implies that our index is quite consistent within each cluster. For New Orleans' spectral graph, we did not notice clear clusters but indeed observed similar pattern in Ashville's data. Overall, our index seems consistent with the listing graph clustering whenever a cluster does appear.

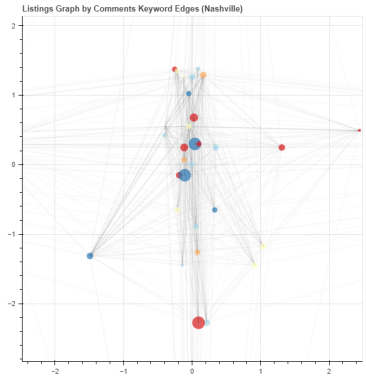
Another clustering analysis that we conducted was more on community detection, essentially looking for subgraph that are somehow more isolated compared to the rest of the listing graph. This can tell us whether our index produces similar values within a community cluster. To do this, we utilizes the asynchronous fluid communities algorithm[2].



(a) Fluid community clustered graph (Austin)



(b) Fluid community graph (New Orleans)



(c) Fluid community graph (Ashville)

FIG. 11: Asychronous fluid community clustering graphs with cluster coloring and size on index values

As shown in Figure 11, we note that there isn't any major cluster but whenever there is a small community of the same color, their sizes tend to be similar, which means their index values tend to be similar. Even though this unsupervised method to assess our index quality seems qualitative, the overall pattern is in-line with what we expect this index can achieve.

IV. CONCLUSION AND FUTURE IMPLICATIONS

The home-sharing economy is on the rise and disrupting the hosteling industry. As consumers shift from traditional renting outlets to online renting and short-term lodging platforms such as Airbnb, the conventional 5-star rating system may not be enough to effectively judge a Uber and Doordash offer more in-depth delivery experience comments, such as "Speed and efficiency", "Handoff", "Care taken with delivery", and "Communication." As more businesses involved in the sharing economy improve their rating systems, it becomes imperative to utilize the alternative text and reviews data that Airbnb collects to create a tokenized rating system that more accurately reflects user experience and allows hosts to improve their listings.

As future research, another challenging problem to tackle would be to use reviews to detect any fake/fraudulent listings. Going a step further, we can utilize image data to identify such listings, and analyze which regions contain the most fake listings. This, combined with a tokenized rating system, would provide Airbnb with tools to determine which listings would need to be removed and which listings to warn users about.

REFERENCES

[1]B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, "Taking the human out of the loop: A review of bayesian optimization," Proceedings of the IEEE **104**, 148–175 (2016).

[2]F. Parés, D. G. Gasulla, A. Vilalta, J. Moreno, E. Ayguadé, J. Labarta, U. Cortés, and T. Suzumura, "Fluid communities: A competitive, scalable and diverse community detection algorithm," in *Complex Networks & Their Applications VI*, edited by C. Cherifi, H. Cherifi, M. Karsai, and M. Musolesi (Springer International Publishing, Cham, 2018) pp. 229–240.