Airbnb Prices in NYC

Comp_Sci 396 Intro to Data Science Pipeline

Team Guardians

December 8, 202

Partners: William Cohen (williamcohen2023@u.northwestern.edu)
Jeffrey He (jeffreyhe2022@u.northwestern.edu)
Bill Wang (billwang2022@u.northwestern.edu)
Hugo Zhang (hugozhang2023@u.northwestern.edu)

Instructor: Dr. Huiling Hu

Introduction

In our project, we look at data from New York City Airbnb listings to try and understand what factors of an Airbnb listing have the greatest impact on that listing's price. We think that multiple groups would find the results of this project useful. For people who own NYC apartments and are considering listing them on Airbnb, the results of this project can help give them a more systematic way to accurately price their listings based on the qualities of the apartment. For people looking to book an NYC Airbnb, the results can help them understand what they are really paying for (as an example, are the higher priced Airbnbs due to that Airbnb being more popular, or is it because it is located in a particular part of the city?).

Dataset

Our dataset consists of over 100,000 entries of Airbnb listings in New York City. Each listing has 26 different features. Among the features that we focused on were the listing price, host identity verification, construction year, minimum nights, number of reviews, review rate number, and host listings count.

Feature Name	*Feature Type*	*Feature Description*
Price	Numerical	The nightly price of the listing
Construction year	Numerical	The year in which the building of the apartment was constructed
Minimum number of nights	Numerical	The minimum number of nights you need to book that particular listing for
Number of reviews	Numerical	The number of reviews the listing has
Review rate number	Numerical	The average rating of the listing (integer format, so only options are 1, 2, 3, 4, 5)
Host listings count	Numerical	The total number of listings the host of this listing has posted on Airbnb
Neighborhood group	Categorical	The NYC borough (Manhattan, Brooklyn, etc) that the listing is located in
Description	String	The description that the host wrote about the particular listing

Figure 1: Table of dataset features

We found this data set on Kaggle, and were drawn to it due to the quantity of the data and the relevance to us. All of our group members use Airbnb, and we were immediately interested in understanding how Airbnb hosts set the price of their listings.

Data Cleaning

For data cleaning, we used OpenRefine to remove or modify our data. Some examples of data cleaning that we completed prior to EDA include the following:

- Removed all blank values using text facet for each relevant column (removed approximately 1500 entries).
- Adjust for spelling mistakes of neighborhood groups, found a particularly large number of mistakes for Brooklyn and Manhattan.
- Modified the price and service fee to remove the dollar sign.
- Remove negative values for minimum nights needed to stay in the Airbnb, as that is physically impossible.
- Removed the 14 entries that had a greater than zero number of reviews but no last review date using custom faceting to identify positive review amounts.

Next, we used Tableau to geographically map all of our data points' longitude and latitude and removed any outliers we saw. Since our dataset is supposed to be only New York City locations, it was very easy to

spot the listings with incorrect coordinates. Finally, we took our final output and exported it as a clean version of the data and proceeded to use this dataset for the remainder of the project.

EDA

The goal of our EDA was to learn some of the basic relationships that existed in our data before diving into the more advanced models. We first started with linear regression, attempting to see if there was a correlation between any of the features and the price of the Airbnb listing. Some of the specific linear regression pairings that we tried:

- Construction year and price.
- Minimum number of nights and price.
- Number of reviews and price.
- Review rate number and price.
- Calculated host listing count and price.
- Number of reviews and review rate number.

Ultimately we got extremely low regression scores for each of these pairings, suggesting that there is no easy linear correlation between them. An example of pairing that we thought would yield good results is construction year vs price (graph to the right, with construction year on the y-axis and price on the x-axis). Our thinking was that listings with lower construction years, representing newer buildings, would command higher listing prices than higher construction years. As seen from the graph and our resulting linear model score of $-\ 4\ *\ 10^{-6}$, this turned out not to be the case.

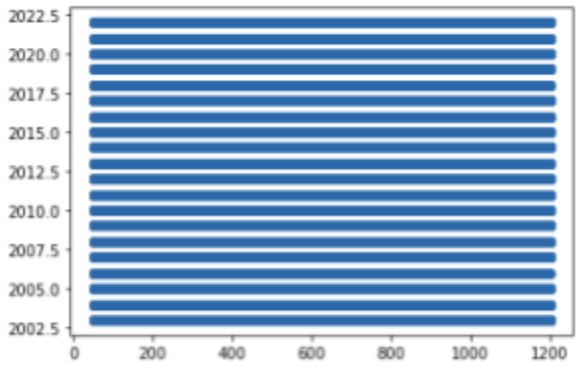

Figure 2: Correlation between construction year and price

After the very poor results from the linear regressions, we understood that the relationship between our other features and the price of the listing is not clear or visually obvious. However, we decided to keep our original research question of trying to predict listing prices, as we thought that there was a good chance that more advanced data science techniques would help uncover the hidden relationships between the features.

Data Modeling

ANOVA

One of our first modeling attempts was to use ANOVA testing to determine if the price of a listing was impacted by the host listing count. We tested this by grouping the host listing count into three distinct categories: 1-3 listings, 4-6 listings, and 7+ listings. From this, we then split the data and followed the requirements of an ANOVA test.

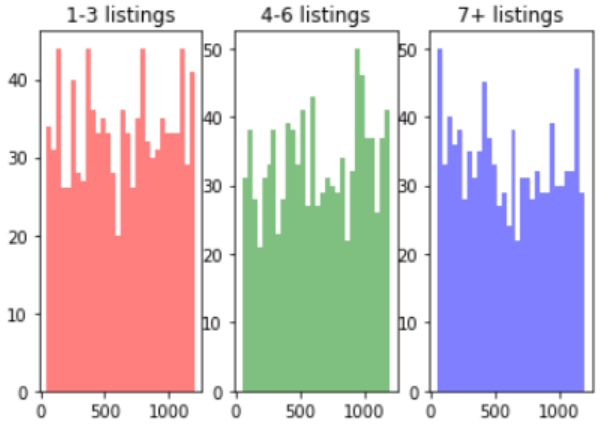

Figure 3: Histogram of Anova categories

From Figure 1, we see that the histograms are a relatively normal distribution. For the second condition, we found that the ratio of the lowest and highest standard deviation was 1.03, which is less than the 2.0 threshold. Finally, we used F_oneway to conduct the test which resulted in a p-value of 0.041. Since this is less than 0.05, we thought that looking more into these relationships could be beneficial.

Regressions

After the initially poor results displayed in our EDA section, we decided to use feature engineering as a tool to potentially subdue some of the noise of the data. Specifically, we used various forms of binning in order to simplify the data.

For the price, we decided that it would be too difficult for the model to assign an accurate precise price number. To combat this problem, we binned the prices into thirds (high price, medium price, and low price) to try to get better results. For the construction years, we used a min-max scaler to get greater relative distances of the values. For the number of reviews, we binned it into quintiles. The results of this binning are demonstrated in the table below.

Original Features				**Engineered Features**		
# Reviews	*Rating*	*Price*	*Construction Year*	*Price Tertile*	*Construction Year Scaled*	*Reviews Quintile*
9	4.0	966.0	2020	3	0.894	3
45	4.0	142.0	2007	1	0.210	5
1	2.0	837.0	2016	3	0.684	1

Figure 4: Table of feature engineering

This binning again led to very disappointing results. We tried correlation coefficients for all of the same pairing as we did in our EDA section (Pearson and Spearman), and arrived at very low correlations and very high p-values each time. According to our regression results, price is simply not predictable using the features that we thought would be predictive. As an example, we assumed that listings that had a higher number of reviews, higher ratings, and more recent construction years would yield higher listing prices, but this does not seem to be the case. The pricing of the listing is not predictable using any of the features that the dataset provides.

Ensemble

In an effort to improve the price tertile prediction accuracy of a listing, we constructed and tested a variety of custom and packaged ensemble classifiers, including a VotingClassifiers, BaggingClassifier, GradientBoostingClassifier, AdaBoostClassifier, RandomForestClassifier, ExtraTreesClassifier, and HistGradientBoostingClassifier.

For our custom VotingClassifier and BaggingClassifer, we first tuned the parameters of our existing decision tree through a GridSearch, which then served as our base classifier for both. Using these ensemble classifiers, we received similar results across the board, ranging from ~0.33 to ~0.36, with our BaggingClassifer achieving the highest accuracy performance of ~0.355. The full results of our ensemble classifiers can be found in the table below. We would assign a “base level” of accuracy at 33.3% (since the pricing data is divided up into thirds).

Method	*Accuracy*
VotingClassifier	~0.3531
BaggingClassifier	~0.3548
GradientBoostingClassifier	~0.3482
AdaBoostClassifier	~0.3412
RandomForestClassifier	~0.3533

ExtraTreesClassifier	~0.3523
HistGradientBoostingClassifier	~0.3520

Figure 5: Table of ensemble classifier results

As seen above, even after performing ensemble classification, our results largely indicated that there is still little to no predictive power when trying to determine the tertile (low, medium, and high) a listing's price would fall into, based on its features.

Decision Tree

Again leveraging our splitting of price into tertiles, we next attempted to predict the price tertile of a listing using a decision tree. Since correlations typically use only one feature to predict another, we wanted to increase the complexity of our models by incorporating the information value of multiple features. We decided to try to use a decision tree to do this. Our decision tree x-values included the following features: number of reviews (quintiles), listing rating, and construction year (min-max scaled).

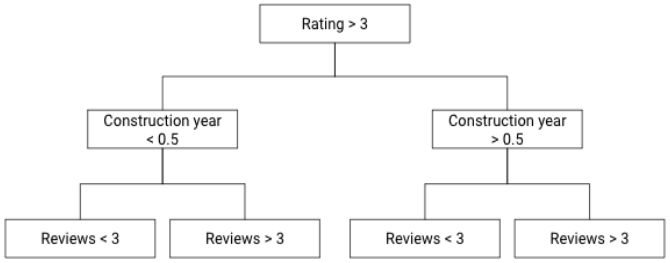

Figure 6: Decision tree

When training with 80% of our data and reserving the other 20% for testing, we got an accuracy of 35.22% for our predictive pricing tertile. Using binning, we split the prices of listings into three categories: low, medium, and high (corresponding to 1, 2, and 3, respectively). This means that if we were to randomly assign listings a rating from 1-3, using no data from that listing to inform our decision, we would expect an accuracy of 33.33%. Put in this context, our accuracy score of 35.22% for this decision tree shows essentially zero predictive power whatsoever.

KNN

Next, we tried to use classification to relate price and host listing count. Since this was not a binary classification, we decided to use KNN instead of logistic regression. Through a grid search, we determined that 11 neighbors would optimize performance and a 3-fold cross-validation was conducted. The average test score was 61%.

We used KNN again to try and draw conclusions about a different relationship, that being the relationship between the price and neighborhood group. The question we were asking is if we could cluster what neighborhood group (Manhattan, Brooklyn, Bronx, Staten Island, or Queens) the listing was in based on the price of the listing. To complete this, we first used feature engineering to convert the neighborhood group (originally a string) into a numerical value. Once we had the five boroughs represented as numbers, we ran a KNN model using five neighbors (to represent each of the five neighborhood groups). Our test scores averaged around 0.42, suggesting that the model is mildly predictive, but not as much as we would have expected or liked.

Logistic Regression

The first step of our logistic regression model was to binarize our target feature. We did this by creating a new feature that we called *super-host*. We defined a listing to be a super-host if its 'calculated host number of listings' feature was greater than or equal to 7. We tested various thresholds before settling on 7, as shown in the chart below, and found that having approximately 10% of listings be super-hosts was a good cutoff.

Cutoff	*% Listings Superhosts*	*Test Score*
4	13.9%	86.1%
7	9.5%	90.4%
10	7.5%	92.4%
20	5.6%	94.3%

Figure 7: Superhost cutoff criteria and test score

Next, we ran 3-fold cross-validation upon a logistic regression and obtained an average test score of 90%. To validate our results, we tried retesting using the different cutoff numbers, and our results are shown above in Figure 4.

As you can see, the numbers in columns 2 and 3 (% listings super hosts and test scores) always add up to nearly exactly 100%. Put differently, our test score would be inversely proportional to the percentage of super hosts. What this essentially means is that, although the test score is high, the model probably has zero predictive power at all.

To understand why, consider the example where the cutoff is 10. Based on Figure 4, the accuracy of the model, when the cutoff is 10, is 92.4% and the percentage of listings that are super hosts is 7.5%. This means that if the model were to guess that the listing was *not* a super-host for each listing, regardless of the information that lasted, it would receive an accuracy of exactly 92.5%. Thus, the fact that the % listings super hosts and test score columns always sum to one shows that this is not a good model.

We also utilized logistic regression for another test to attempt to predict whether or not a user (host) is verified based on the number of calculated listings they have on Airbnb. We chose to use a logistic regression because the verification status of a user is binary: the user is either verified (1) or unverified (0). Prior to running tests, we expected the classifying score to be rather high as we thought users with a higher number of listings would be more likely to put in the effort to become verified. The average test score of the logistic regression, in this case, was 62%, which we view as being a mildly predictive model in predicting whether or not a user is verified.

K-Means Clustering

Using K-means clustering, we took the minimum nights and number of reviews from each entry and clustered them based on price. We split them into 4 groups, and the average price for each was as follows: $193, $483, $770, and $1057, indicating that the model was successful in splitting them into different price levels. However, when looking at the associated entries for minimum nights and the number of reviews, the average of each feature across all price levels was the same, with each having an average of 7 minimum nights and 27 reviews. This shows that there is no relationship between the different price levels and the other features, which further reinforces the fact that there are no features that can be used to predict price.

Mean	*Price*	*Minimum Nights*	*# Reviews*
Group 1	$193	7	27
Group 2	$483	7	27
Group 3	$770	7	28
Group 4	$1057	7	27

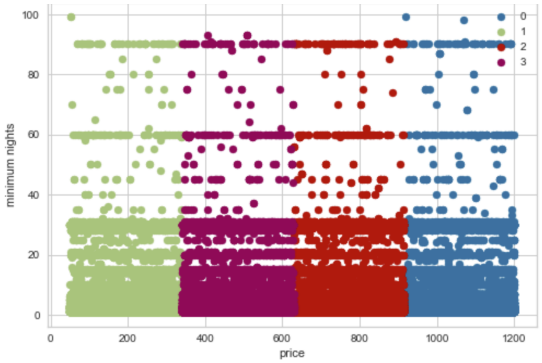

Figure 8 and 9 : Table (left) and graph (right) of k-means clustering results

Association Testing

While not directly related to finding the price, we found it interesting to run some association testing on the name of the listing and the rules of the listing. Our results are shown in Figure 6 below.

Listing Name				**Listing Rules**			
Pair	*Support*	*Confidence*	*Lift*	*Pair*	*Support*	*Confidence*	*Lift*
New, brand	0.006	0.994	30.58	Occupancy, building	0.055	0.953	7.30
Lower, east	0.006	0.851	15.01	Building, rules	0.050	0.913	6.99

Exchange, stock	0.002	0.984	81.16	Parties, no	0.052	0.950	1.41
Home, away, from	0.005	0.958	25.98	Smoking, no	0.095	0.981	1.46
Private, bathroom	0.004	0.820	5.59	Guests, allowed, are	0.058	0.914	2.536

Figure 10: Association using Apriori

Conclusion

Our initial goal was to try to find the predictive features that went into the pricing of Airbnb listings in New York City. After our extensive data analysis, we concluded that there were no features in our dataset that could be used to adequately achieve this initial goal.

However, we were still able to predict and confirm several relationships in our dataset through our models such as clustering neighborhood groups by price, determining if a verified host by their listings, and analyzing variance on the price of listings versus the number of listings by an individual host.

Looking back on our work, there were some indications through our EDA that it would be difficult to predict the price of the Airbnb listings. However, we continued pursuing our original objective because we felt that if the patterns were already incredibly apparent through the basic EDA, our project would not have been very interesting. Additionally, we were not too discouraged in the beginning because we felt that the more advanced data science tools introduced later in the quarter would allow us to uncover clearer, hidden patterns.

Overall, we concluded that it is not possible to predict the price of an Airbnb listing given the features in the data set. Despite these lackluster results, we surmised that the factors that most contribute to the price of an Airbnb listing must be qualitative instead of quantitative as those factors are not present in our dataset. These results support the idea that every listing is so different and offers its own unique advantages to the point that the price that the host decides to set for that listing is completely idiosyncratic and unique to that particular listing. While disappointing, we feel that this conclusion is still useful as it allows hosts of Airbnb to be more flexible and less formulaic when deciding what price to charge.

Lastly, we believe that this project was a good reminder that, in data science, you must be as willing to accept a “no relationship” result as you are a “strong relationship” result in order to maintain intellectual honesty. While the strong relationship result is certainly a more exciting and desirable one, it simply was not the case for our data set.