Airbnb Recommendation Quote System

May 11, 2021

Abstract

This paper analyzes the "Airbnb_NYC_2019" dataset and describes the two applications that we built up based on the dataset. The "Airbnb_NYC_2019" dataset contains the information of the Airbnb houses in New York city in 2019. It contains 48895 data points with 16 features for each of the points. The dataset contains a lot of null values but no duplicate data points. To clean our data, the null values were replaced with 0s. Analysis of the relationship between variables was demonstrated in the Exploratory Analysis section with the help of distribution maps, histograms, pie charts, and box plots. After the Exploratory Data Analysis, the data is processed through the clustering algorithms. Through the clustering algorithms, the data points show some patterns with clustering which suggest that the houses are able to be recommended to users as some small groups. The two applications are presented at last as the results of the project. The section for the quote system shows the using of XGBoost Feature Importance that shows which housing features are more important and Multiple Linear Regression that used to build up the quote system. The section for the recommendation system presents an example of using the system and a brief explanation of how the system is built up through TfidfVectorizer.

1 Introduction

Airbnb is one of the largest housing platform in the world. The dataset "Airbnb_NYC_2019" has contains the information about the Airbnb's houses in New York city at 2019. For each houses, the dataset contains their prices, renter's reviews, locations, room types, etc. Since there are a lot of null values in the dataset, we have fill up all the null values with 0s, so it will be easier to manipulate. Moreover, we have add some columns for better interpretation. We analyze the dataset through graphs and tables, and we decide to build up some applications for users that are convince to user. The applications we build up are the recommendation system and the quote system. The recommendation system is a system that will recommend houses based on the user's input or previous stay with Airbnb. It recommend houses base on the house' features like the price and the room type. The quote system are a system that estimate the housing price by the neighborhood, room type, and staying nights. The quote system are mainly based on the multiple linear regression model. Meanwhile, before we build up the recommendation system, we have use the clustering algorithms to check if there are clusters within the houses. If there are clusters within the houses, then the recommendation system will be reasonable.

2 Data description

The "Airbnb_NYC_2019" dataset used is a csv file, which we import into Python as a pandas dataframe. It has 48895 rows of observations and 16 columns of features, where each row of data represents an airbnb listing. All the listings in the dataset are located in New York city, and the data was collected in 2019. The data types we have are:

- floating point numbers
- integers
- objects

We can see that the variables associated with Airbnb reviews: 'last_review' and 'reviews_per_month', have a significant amount of missing data. We will investigate further about why, possibly because of the pandemic, or maybe because a few hosts' airbnb listings have been unavailable for a significant portion of the year this data was taken from.
So here we can think of the dataframe as a matrix such that:

- The rows $X_1, ..., X_{48895}$ are the vector representations of our $n = 48895$ observations, and each observation is an airbnb listing
- The columns are the d dimensions of our data. Each column represents a feature
- Each observation X_i has $d = 16$ associated features which describes each individual airbnb listing (observation)
- Each X_i is a $d = 16$ dimensional vector, and we have $n = 48895$ of them

2.1 Exploratory Data Analysis

2.1.1 Understanding our Data

Data Types

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48895 entries, 0 to 48894
Data columns (total 16 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   id                              48895 non-null  int64
 1   name                            48879 non-null  object
 2   host_id                         48895 non-null  int64
 3   host_name                       48874 non-null  object
 4   neighbourhood_group             48895 non-null  object
 5   neighbourhood                   48895 non-null  object
 6   latitude                        48895 non-null  float64
 7   longitude                       48895 non-null  float64
 8   room_type                       48895 non-null  object
 9   price                           48895 non-null  int64
 10  minimum_nights                  48895 non-null  int64
 11  number_of_reviews               48895 non-null  int64
 12  last_review                     38843 non-null  object
 13  reviews_per_month               38843 non-null  float64
 14  calculated_host_listings_count  48895 non-null  int64
 15  availability_365                48895 non-null  int64
dtypes: float64(3), int64(7), object(6)
memory usage: 6.0+ MB
```

Figure 1: Data types

There are 7 integer, 3 float, and 6 object type variables. Also, there are nearly 10,000 null values in "last_review" and "reviews_per_month" (38843 Non-Null data points).

Unique Values

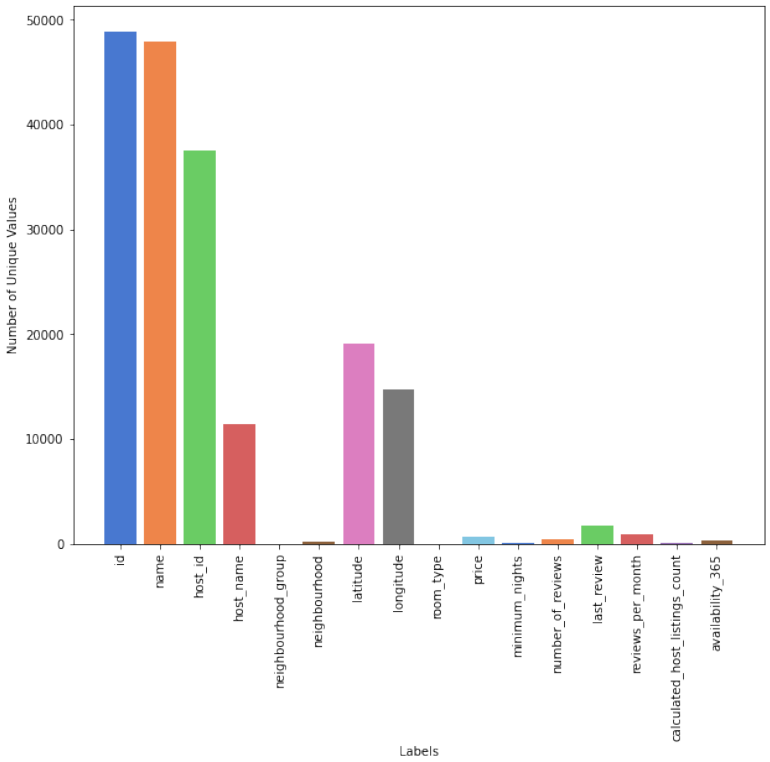

Figure 2: Unique value count

Here we can see the unique values of each variable.

Null Values

```
id                                  0
name                               16
host_id                             0
host_name                          21
neighbourhood_group                 0
neighbourhood                       0
latitude                            0
longitude                           0
room_type                           0
price                               0
minimum_nights                      0
number_of_reviews                   0
last_review                     10052
reviews_per_month               10052
calculated_host_listings_count      0
availability_365                    0
dtype: int64
```

Figure 3: Null values in the data

Here we see that there are 16 null values for "name", 21 null values for "host_name", and 10052 null values for "last_review" and "reviews_per_month" which is quite significant.

Duplicate Rows

```
# Rows containing duplicate data
duplicate_rows_df = df[df.duplicated()]
print('Number of duplicate rows: ', duplicate_rows_df.shape)
```

```
Number of duplicate rows:  (0, 16)
```

Figure 4: Duplicate rows in our data

There are no duplicate rows in the dataset.

Descriptive Statistics

```
# Provide the count, mean, standard deviation, minimum and maximum values and the quantiles of the numerical data

df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
id	48895.0	1.901714e+07	1.098311e+07	2539.00000	9.471945e+06	1.967728e+07	2.915218e+07	3.648724e+07
host_id	48895.0	6.762001e+07	7.861097e+07	2438.00000	7.822033e+06	3.079382e+07	1.074344e+08	2.743213e+08
latitude	48895.0	4.072895e+01	5.453008e-02	40.49979	4.069010e+01	4.072307e+01	4.076311e+01	4.091306e+01
longitude	48895.0	-7.395217e+01	4.615674e-02	-74.24442	-7.398307e+01	-7.395568e+01	-7.393627e+01	-7.371299e+01
price	48895.0	1.527207e+02	2.401542e+02	0.00000	6.900000e+01	1.060000e+02	1.750000e+02	1.000000e+04
minimum_nights	48895.0	7.029962e+00	2.051055e+01	1.00000	1.000000e+00	3.000000e+00	5.000000e+00	1.250000e+03
number_of_reviews	48895.0	2.327447e+01	4.455058e+01	0.00000	1.000000e+00	5.000000e+00	2.400000e+01	6.290000e+02
reviews_per_month	38843.0	1.373221e+00	1.680442e+00	0.01000	1.900000e-01	7.200000e-01	2.020000e+00	5.850000e+01
calculated_host_listings_count	48895.0	7.143982e+00	3.295252e+01	1.00000	1.000000e+00	1.000000e+00	2.000000e+00	3.270000e+02
availability_365	48895.0	1.127813e+02	1.316223e+02	0.00000	0.000000e+00	4.500000e+01	2.270000e+02	3.650000e+02

Figure 5: Statistical description of the data

Looking at price, the 5th row of the table, we can see that the mean is higher than the median which suggest that the distribution of the price is right skewed. But, we also see that the minimum price is 0 indicating 'free' housing. Since it is unnatural that there is 'free' housing, let us investigate that.

```
len(df[df.price == 0])
```

```
11
```

Figure 6: Number of rows with zero cost

So, there are 11 rows of data with no price which is most likely an error. These places were perhaps 'NA' values that were inputted as 0's.

Host Name Issues

```
# Highighting some inconsistencies in host names.

# Filling in "None" in place of NA values

df[["name", "host_name"]] = df[["name", "host_name"]].fillna("None")

print("There are " + str(len(df[df["host_name"]=="None"]))
      + " unindentifiable hosts and " + str(len(df.host_id.unique()))
      + " unique identifiable ones")
```

```
There are 21 unindentifiable hosts and 37457 unique identifiable ones
```

Figure 7: Number of unique and NA values in "host_names"

There are 21 hosts that do not have a name, and there are 37457 unique hosts that do have a name. Also, many hosts have multiple listings under their ID which is why the amount of unique identifiable hosts does not match the length of our data set (48895 rows).

Cleaning the Data Set

To continue with cleaning our data set, we decided to fill our NA values with zeroes. The zero values for the "last_review" and "reviews_per_month" variables did not affect our analysis. Meanwhile, we added "price_range" and a "price_range_difference" column to our data. "Price_range" categorizes the date into three categories based on the three quantiles (75%, 50%, 25%): high, medium, and low. These are useful in our analysis for our keyword matching system for the recommendations. On the other hand, "price_range_difference" calculated the difference from mean price as an additional tool to our interactive map. The value appears when the mouse hovers over a point in the map.

2.1.2 Visualizing Relationships in the Data

Neighbourhood Groups Count

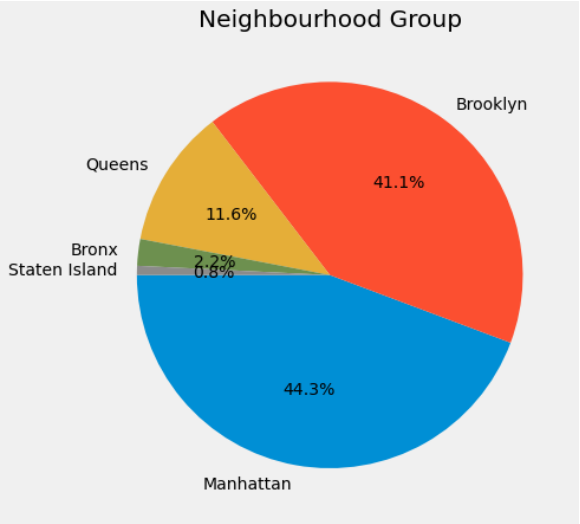

Figure 8: Neighbourhood Group Count

Highest percentage of listings are in Manhattan(44.3%). Followed by Brooklyn(41.1%), Queens(11.6%), Bronx(2.2%), and Staten Island(0.8%) respectively. There are about 85% of the Airbnb housing are in the Manhattan and Brooklyn neighbourhood.

Map of Neighbourhood Groups

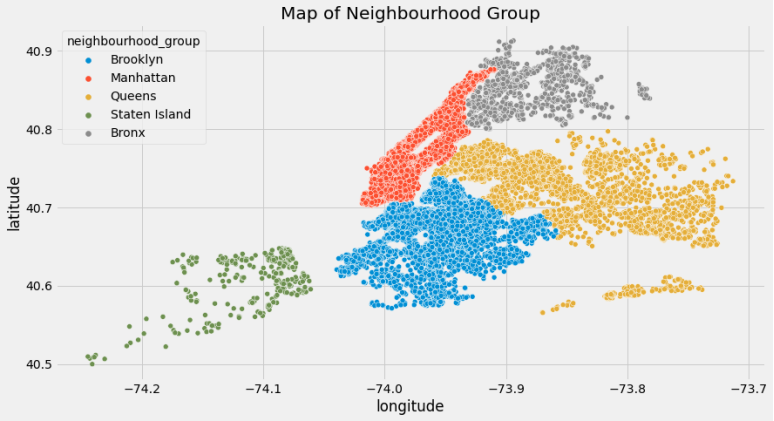

Figure 9: Neighbourhood groups on a map

This map shows the concentration of listings in each neighbourhood group.

Types of Rooms

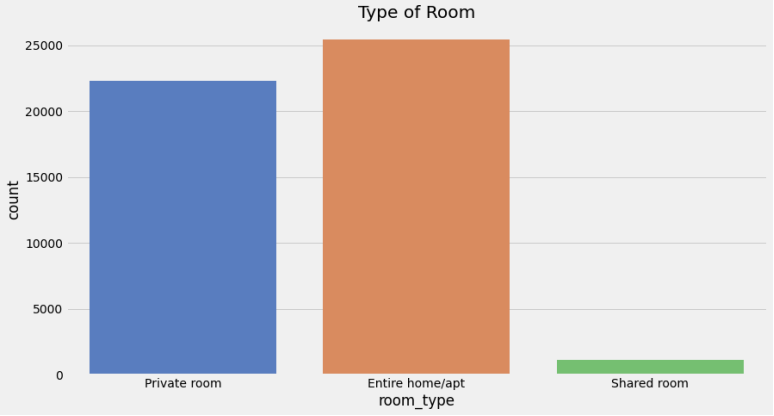

Figure 10: Room type count

This graph displays the amount of rooms of each type. The amount of shared room is significantly fewer than others which is quite interesting to see.

Types of Rooms based on Neighbourhood Groups

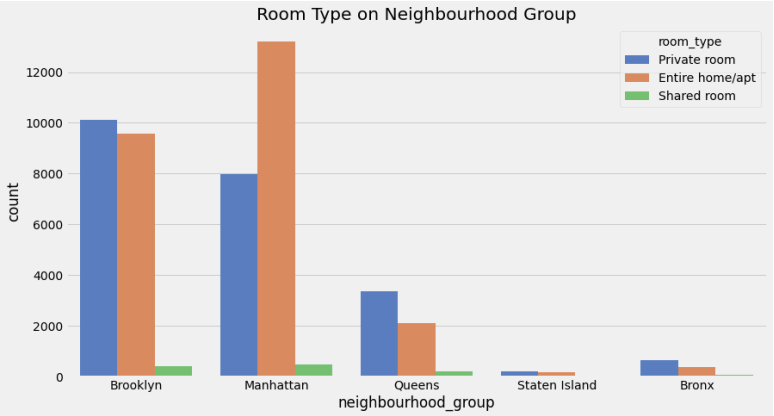

Figure 11: Room type count per neighbourhood group

Here we observe that Manhattan has the highest amount of home/apartments whereas Staten Island has the least. Also, Manhattan is the only place that has more Entire home/apt than private room while all the other places have the Entire home/apt slightly fewer than the private room. Moreover, even Manhattan has the highest amount of houses, the amount of shared room in Manhattan are very much same as Brooklyn and Queens.

Neighbourhood Group and Availability

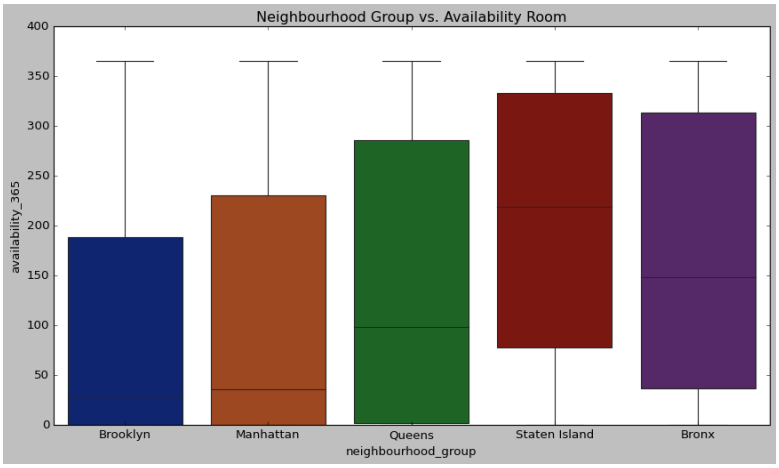

Figure 12: Box plot of neighbourhood group vs availability

Here we see that the median availability of listings in Staten Island is higher relative to the rest of the neighbourhood groups especially Manhattan and Brooklyn. This could be attributed to the popularity of listings and/or population dynamics.

Price Distribution Map

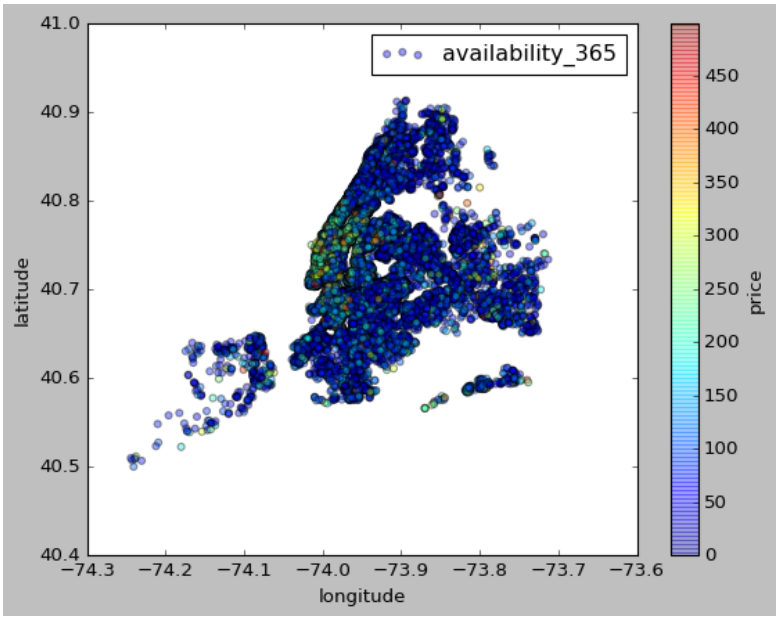

Figure 13: Price distribution map along with availability

This plot reveals that the highest prices lie in the Manhattan neighbourhood with high concentration.

Neighbourhood Group Price Distribution

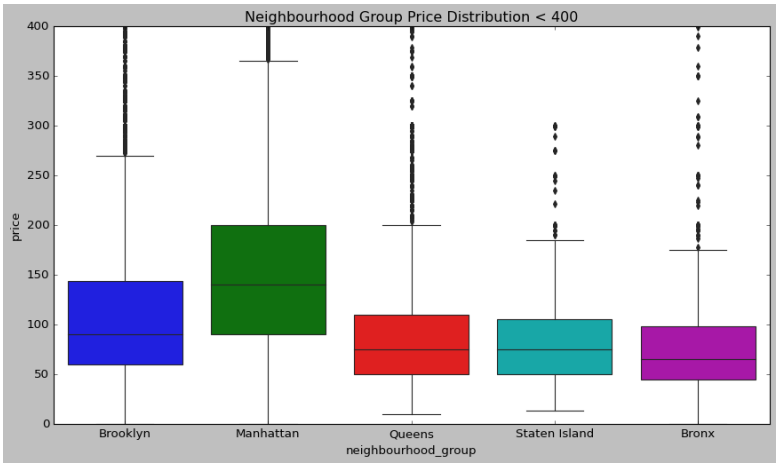

Figure 14: Price distribution box plot based on neighbourhood group

We can state that Manhattan has the highest price range with a median of $140, followed by Brooklyn with $90. Queens and Staten Island follow a similar distribution while the Bronx is the least expensive among all of them.

Correlation Heatmap

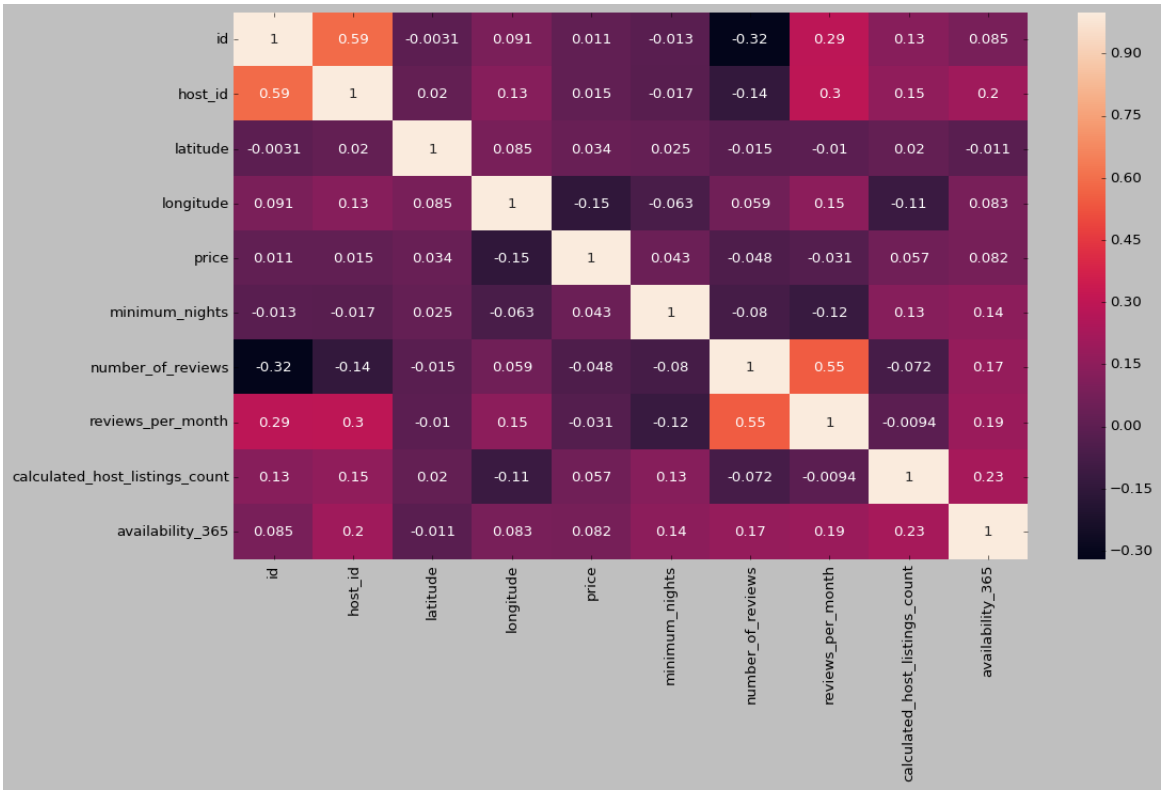

Figure 15: Correlation heatmap of the variables

Here we see that correlation values are highest between host_id and id which is not very meaningful. However, we can see a high correlation for number_of_reviews and reviews_per_month. Moreover, id has a weak correlation with number_of_reviews and reviews_per_month, and the correlation between the id and number_of_reviews are negative.

2.1.3 Visualizing Prices for Each Room Type

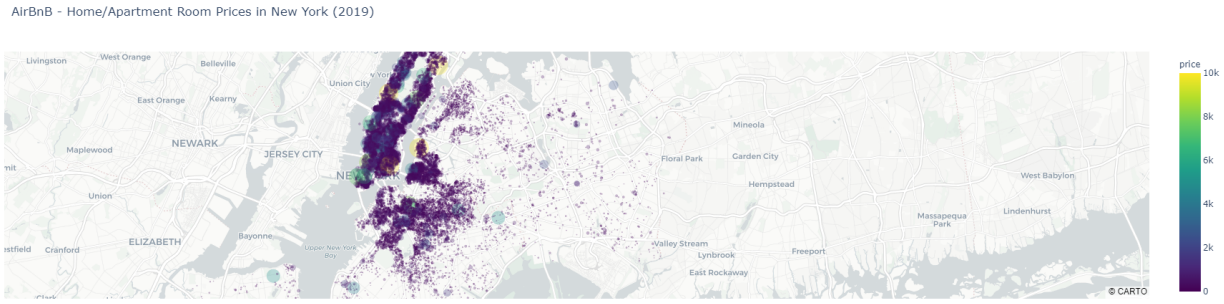

Figure 16: Scatter Map of Home/Apartment Airbnb prices in NY

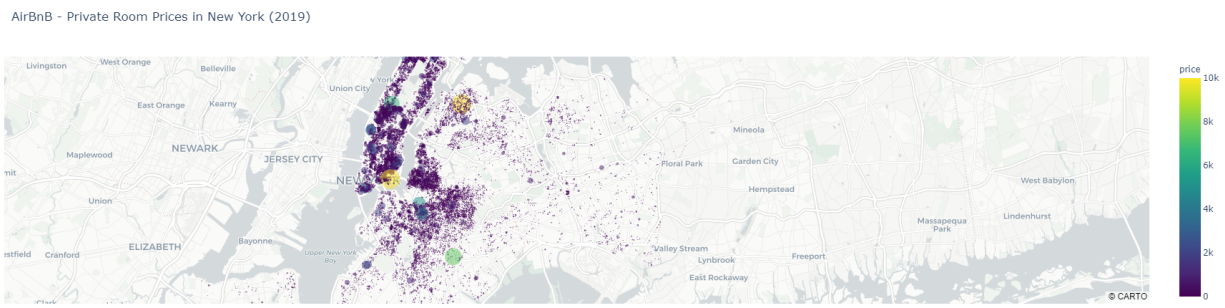

Figure 17: Scatter Map of Private Room Airbnb prices in NY

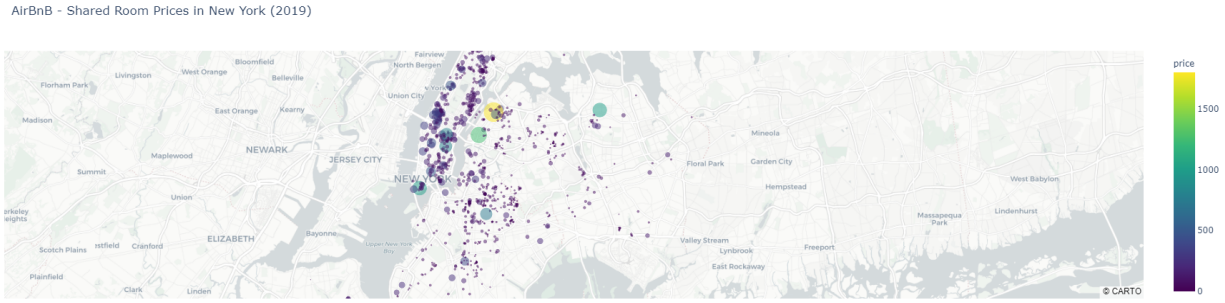

Figure 18: Scatter Map of Shared Room Airbnb prices in NY

No matter the room type, there are more Airbnb rooms in Manhattan as well as more expensive rooms. It also seems like the closer to the East River (where the bridges are located) the rooms are, the more expensive the room is.

2.2 Clustering

In order to see if it is reasonable to build up a recommendation system, we will apply clustering methods on the dataset, specifically k-modes and k-means. The goal of clustering is to bring out the underlying structure of the given data that may not be apparent by it's given shape. In k-means and k-modes, the algorithm initializes random centroids (equal to the number of clusters we assign in the arguement of the function) as cluster centers. The cluster means are used as centroids in k-means, and in k-modes, we use the cluster modes as the centroids. Then, k-means iterates through all the data points and assigns them to the cluster in which it is closest to the centroid. In the discrete case, k-modes calculates the dissimilarity between pairs of data points and assigns the data to the cluster in which the dissimilarity score is the least from the centroid. Both the kmeans and kmodes algorithms are very sensitive to the initialization of the number of clusters and the the number of initialized centroids. So, although these methods are unsupervised learning algorithms, they perform the best when you have some prior information about the data and some sense of the number of groupings it should have.

K-modes is a clustering algorithm that is specialized for categorical variables. Though it is expansion of

the K-means algorithm, it is different from k-means clustering. Instead of using Euclidean space to measure distance between pairs of data points, K-modes uses the likelihood that the data belongs to a cluster, based on the number of times it has appeared in the sample. [1] So it makes more sense to use k-modes, since we are dealing with many discrete features. If the result of the clustered data shows us patterns about the underlying structure, then it is reasonable to create a recommendation system that recommends similar rentals based on the information a user inputs.

2.2.1 K-modes Algorithm

We use the **kmodes** library to perform clustering on the categorical data, and we initialize the function using 'Huang' and number of clusters = 6.

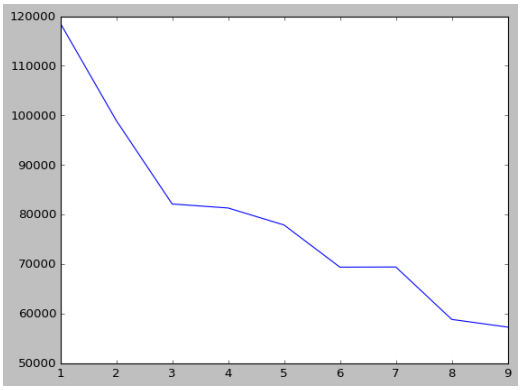

Figure 19: K-modes Cost Function on 1 to 9 clusters

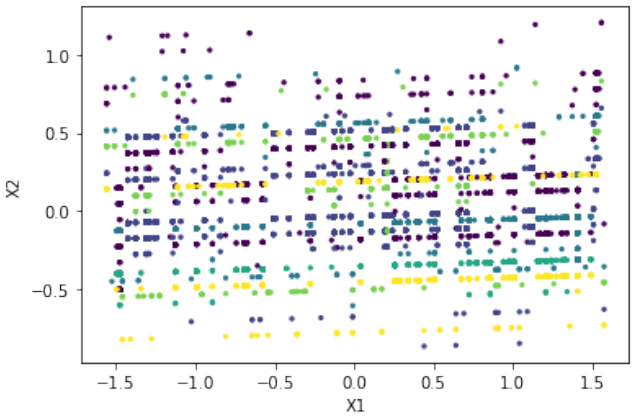

Figure 20: K-modes clustering with dimension reduction

We draw out the cost function to compute the measure of error from k-modes method with a range of cluster from 1 to 9, shown in figure 19. We notice there are elbow points at 4, 6, and 8. Then, after displaying the results, we decide to use 6 clusters for best visualization. Meanwhile, in order to visualize the result of clustering, we use the factor analysis to reduce the dimension of the data set from to 2 dimensional.

Figure 20 shows the clustering result with a 2 dimensional plot. Each color represent a cluster, and since the dimension is reduced, the axis itself does not contain any specific meaning. From the plot, we can see that there is not any cluster that are really "cluster" together. However, they show some horizontal patterns, and each cluster forms about 4 horizontal lines. That is, since there are the similar patterns for each clusters, we assume that it is meaningful to build a recommendation system.

In attempting to get better interpretability and performance with clustering, we took the following measures. We converted the following categorical features: *neighbourhood_group*, *neighborhood*, and *room_type* from a python data type **object** to the data type **category**. Now that our categorical variables are recognized in python as category arrays, each category within the array has an integer array of categorical codes, which maps to the real value of each corresponding observation. We replace the categories with the corresponding numeric categorical codes, so that we have quantitative data as inputs to our clustering algorithm. This now becomes our 48895 x 10 dimensional data matrix.

Then, we scale our data matrix using the *StandardScaler* module from the *sklearn* library. Standardizing our

data is very important when it comes to clustering, because the K-means algorithm uses distance in Euclidean space as a measure of similarity between data points, so we need to make sure all of our data is being compared on the same scale, with $\mu = 0$ and $\sigma = 1$. Otherwise, our clustering algorithm will not perform well. This is due to the different scales across the features of our data, which makes the chosen algorithm think the distance and similarity between data points is farther than it actually is. This is what we believe made the k-modes model with reduced dimensions above not perform well, so it could not distinguish between clusters. Now, we will use our standardized data all on the same scale to perform kmodes and kmeans, using commands from the sklearn library.

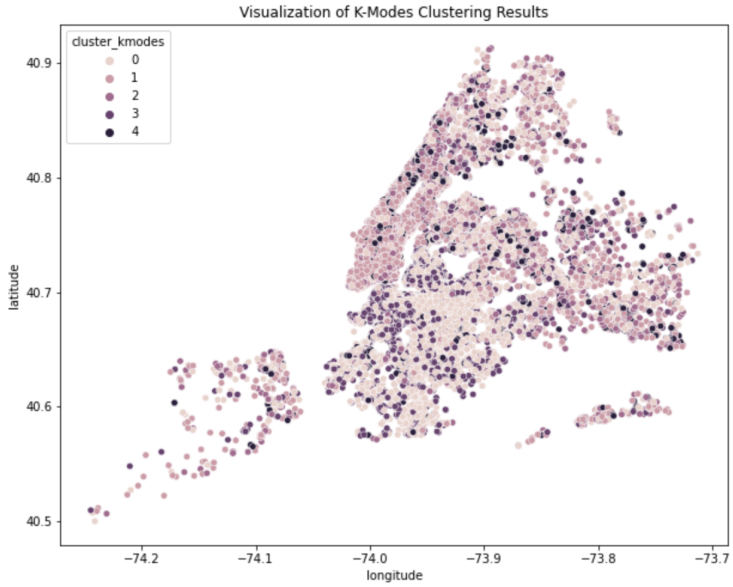

Figure 21: Cluster Visualization as a Result of K-modes on Standardized Data

We initialize each algorithm with number of clusters = 5, because we have prior information about our data having 5 neighbourhoods. We want to see if the k-modes and k-means algorithms capture this structure in the data within groupings it assigns to each data point. Furthermore, instead of reducing the dimensions of the data for clustering visualizations, we plot the clusters using the longitude and latitude tuple corresponding to each data point as axes for better interpretability of possible grouping of neighbourhoods.

2.2.2 K-means Algorithm

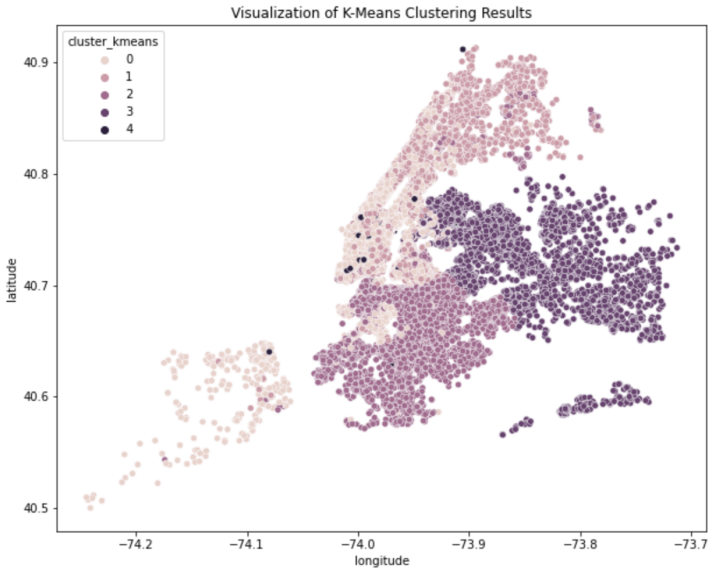

Figure 22: Cluster Visualization as a Result of K-means on Standardized Data

We can see that the k-means algorithm with our standardized data performed remarkably better than k-modes. While k-means wasn't perfect at separating the clusters completely, it did group majority of the data into distinguished clusters. K-modes seemed to capture 2 main clusters, where most of the data is mapped to cluster 0 and cluster 1. However, none of the clusters are well separated. K-means was able to extrapolate the true underlying structure of the data coming from the 5 different neighbourhood groups in New York.

	neighbourhood_group	room_type	cluster_kmeans
1	Manhattan	Entire home/apt	0
4	Manhattan	Entire home/apt	0
5	Manhattan	Entire home/apt	0
9	Manhattan	Entire home/apt	0
10	Manhattan	Entire home/apt	0

Figure 23: First 5 observations in Cluster 0

	neighbourhood_group	room_type	cluster_kmeans
2	Manhattan	Private room	1
7	Manhattan	Private room	1
8	Manhattan	Private room	1
11	Manhattan	Private room	1
13	Manhattan	Private room	1

Figure 24: First 5 observations in Cluster 1

	neighbourhood_group	room_type	cluster_kmeans
0	Brooklyn	Private room	2
3	Brooklyn	Entire home/apt	2
6	Brooklyn	Private room	2
12	Brooklyn	Private room	2
16	Brooklyn	Entire home/apt	2

Figure 25: First 5 observations in Cluster 2

	neighbourhood_group	room_type	cluster_kmeans
46	Queens	Private room	3
77	Queens	Private room	3
143	Queens	Private room	3
161	Queens	Private room	3
181	Queens	Entire home/apt	3

Figure 26: First 5 observations in Cluster 3

	neighbourhood_group	room_type	cluster_kmeans
700	Manhattan	Entire home/apt	4
754	Manhattan	Entire home/apt	4
1305	Brooklyn	Entire home/apt	4
1449	Manhattan	Entire home/apt	4
1758	Brooklyn	Entire home/apt	4

Figure 27: First 5 observations in Cluster 4

Looking deeper into the clustered data from k-means, we observe that the algorithm did in fact capture the structure of the 5 neighbourhood groups. Clusters 0 and 1 captured almost all of the Manhattan data, separated by private room vs entire home as the type of listing. We found it interesting that the algorithm split Manhattan data into 2 different clusters and separated it by room type. This might be because the Manhattan neighbourhood has the most airbnb listing counts out of all the neighbourhood groups, and k-means/k-modes perform best when cluster groups have the same cardinality, which we do not have here. Moreover, cluster 2 captures Brooklyn data with mixed room types, and cluster 4 contains the remaining "spill over" data from Manhattan, along with the rest of Brooklyn. Cluster 3 is well separated containing only Queens data. all the data from Queens, but it doesn't separate by room type. Overall, k-means performance was still relatively well in general, and much better than k-modes.

3 Results and Interpretation

3.1 AirBnb Quote System

3.1.1 XGBoost Feature Importance and Multiple Linear Regression

Using XGBoost to determine feature importance, we discover the main factors in determining the price of the Airbnb is the neighbourhood group that it is in, the room type, and minimum number of nights. Specifically, we find that if the AirbnB is a home/apartment or in Manhattan to be the biggest factor in determining price. Reviews per month also has high importance meaning that the Airbnb is most likely a popular spot and has high demand causing a difference in prices.

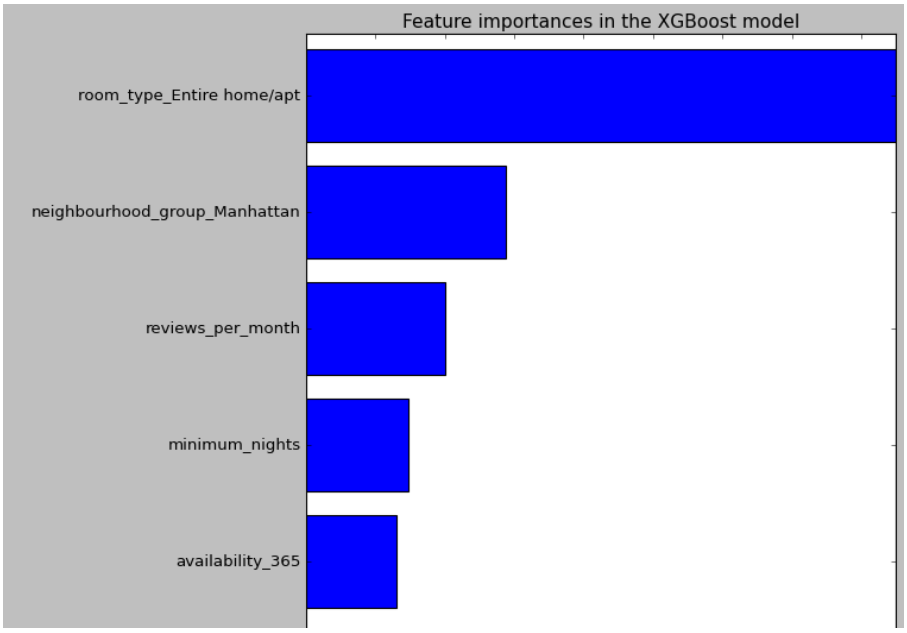

Figure 28: Feature Importance

We use Multiple Linear Regression for the quoting system since we want to provide a rough estimate using only the neighbourhood group, room type, and minimum number of nights (the three biggest factors in determining price according to XGBoost, shown in 28). We don't use other variables in our linear regression model since a user is more likely to know what kind of neighbourhood they want to stay in as well as the room type and number of night as opposed to something like the number of reviews they want for a potential Airbnb. Furthermore, Multiple Linear Regression gives better accuracy than XGBoost for this combination of variables.

3.2 User-Based Airbnb Room Recommendation System with Interactive Map

The idea of the recommendation system is that it will take the user's previous Airbnb stay, and based on the features of that Airbnb room, the user is returned similar Airbnb rooms that they may be interested in.

Here in Figure 29, the user's last stay was inputted to be at A Beautiful Brownstone which is in Brooklyn, Bedford-Stuyvesant which is an home/apartment at a price of 170. The user's last stay is then mapped along with other Airbnb rooms that have similar features as A Beautiful Brownstone.

1	2137796	A Beautiful Brownstone	10909829	Richelle And Pela		Brooklyn	Bedford-Stuyvesant	40.68478	-73.92971	Entire home/apt	170	3

Figure 29: User's Last Airbnb Info

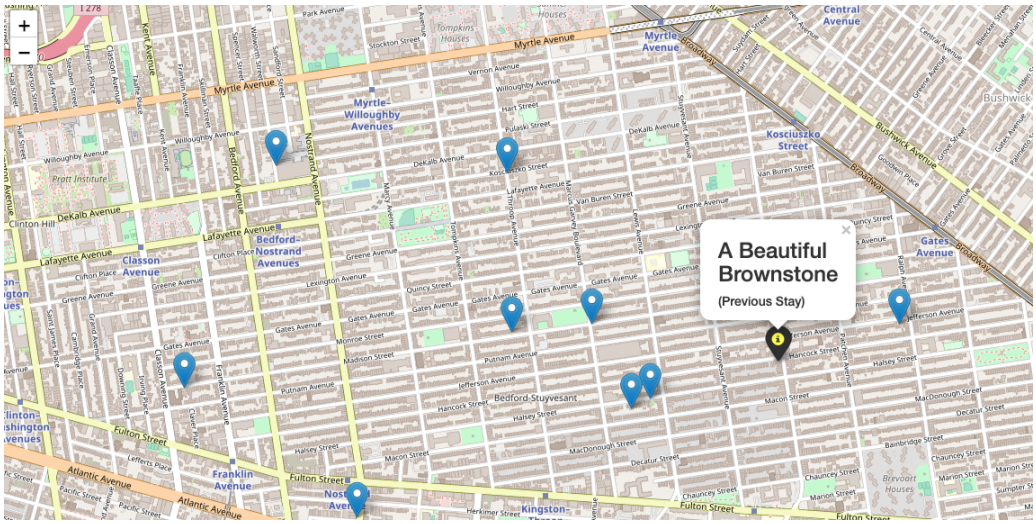

Figure 30: User's Last Stay

Using **TfidfVectorizer** for term frequency and importance, based on location, price range, room type, and host of the user's previous Airbnb, the top 10 recommended Airbnb rooms are then given and mapped out for the user. In Figure 31, we see that a majority of the recommended rooms (blue markers) are also home or apartments, fall within the same price range as the user's last Airbnb, and/or is in a similar location. Thus, showing that our recommendation is effective.

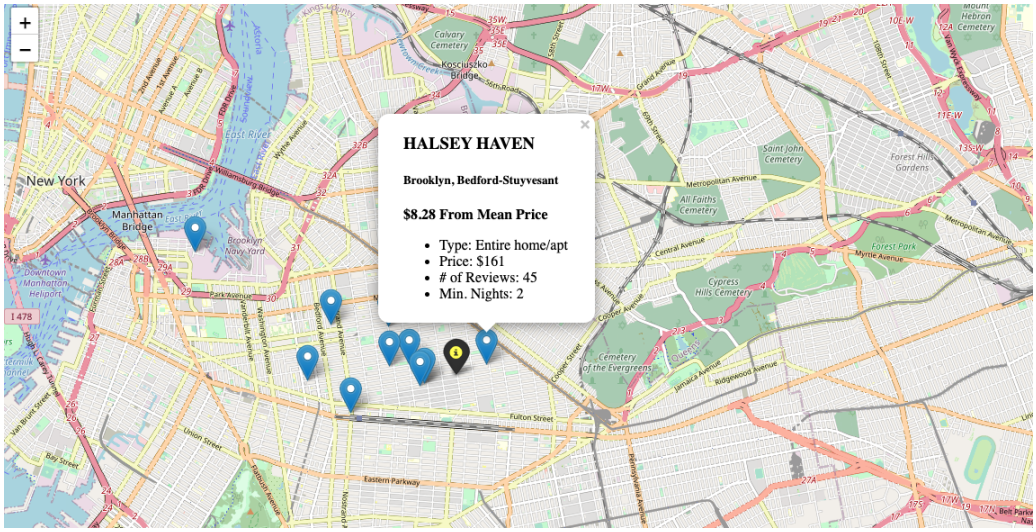

Figure 31: Recommended Airbnb Rentals

4 Discussion

4.1 Low Accuracy in Multiple Linear Regression/XGBoost

These methods give low Accuracy mostly likely due to the unexplained variance in prices. This could be due to the AirBnb data missing known key factors in determining prices such as number of rooms. If this was included in the data, accuracy would most likely be increased.

4.2 Future Improvements

K-means was able to extract the underlying neighbourhood group structure within the standardized data relatively well. However, there are improvements that can be made to get better cluster distinctions. K-means relies on the assumption that the cardinality between each cluster is equal, and this assumption was not met. It can be clearly seen as a result of the clusters, that Manhattan data was disproportionately larger than the other 4 neighbourhood groups, thus resulting in 2 Manhattan clusters plus some spill over into cluster 4. In the future, we can undersample the data from Manhattan for the purposes of clustering to get better results. K-modes did not perform well on the standardized and quantitative data. We believe this could be due to the fact that the standardized data is measured on the same scale, so computing frequency counts for the cluster modes may have been affected. In the future, we plan to combine k-means and k-modes and implement an algorithm known as *k-prototype clustering*, which is used for mixed data types as we encounter among the features in our Airbnb dataset.

5 Conclusion

In this project, we analyzed the "Airbnb_NYC_2019" data set and created a recommendation and a quote system. The data set is cleaned by filling up the null values with zeroes and adding two columns to the data set. The visualizations and exploratory analysis of the data set revealed various relationships between the features of the data set. Moreover, by processing the data set through clustering algorithms, the data points displayed patterns of clusters and confirmed that the recommendation system is in fact meaningful. Lastly, the quote and recommendation systems, are present with models in the results and interpretations.

References

[1] Zhexue Huang. "Extensions to the k-means algorithm for clustering large data sets with categorical values". In: *Data mining and knowledge discovery* 2.3 (1998), pp. 283–304.