ANALYSING AIRBNB AND FOURSQUARE DATA

Banki Jey

MARCH 22, 2021

Contents

I. Introduction

A. Background

Sharing economy is currently a major global trend. One popular example of sharing economy is Airbnb, which offers apartment owners an opportunity to make access niche cashflows from their properties and visitors an inexpensive alternative to hotels. There are over 7 million listings on Airbnb with more than 150 million users and 193 million bookings as at 2020. (https://www.businessofapps.com/data/airbnb-statistics/).

B. Problem Statement

The current pandemic has brought travel restrictions. Tourism is one of the most missed activities and a lot of tourists cannot wait for the times when they can freely plan and make their holidays. However planning a holiday might be a cumbersome task. Searching for the best flights and organizing accommodation can be underestimated time consuming tasks.

In this project, I attempt to examine Airbnb listings in Berlin and using clustering algorithms to group listings based on prices, popularity and popular locations around them. A protype program to help prospective visitors narrow down listings based on the nearby locations is also developed.

II. Data

A. Data Sources

The core data for this project is sourced from the website: insideairbnb.com. On this website, data for the Airbnb listings for different cities are available. The datasets include detailed data on listings, a simplified (reduced columns) and cleaner listings dataset, detailed data on reviews as well as calendar data useful for time series analysis. These were compiled on the 20th of February 2021.

Furthermore, location data from Foursquare is used to gather data on nearby venues around the listings. Four square's places API is able to provide detailed information on places around the world including and not limited to reviews, location and category. For our purposes, we are mostly interested in the data on the categories of venues.

B. Data Cleaning and Feature Extraction

The Airbnb listings data has 19978 rows and 74 columns. The information include the listings unique identifiers, names and URL, host details, details on reviews, and characteristics of the listings. The columns (features) are split between categorical and numerical.

The first cleaning step would be to drop the irrelevant features. Columns with a lot of missing values were dropped. Details on the description of the listings are dropped. Also hosts information, except whether the host is a super host, are dropped.

Some columns have very similar information. For example, the 'property_type' column has 68 unique values for type of property. These are summarized in the 'room_type' column into 4 unique property types: 'Entire home/apt', 'Hotel room', 'Private room', and 'Shared room'. Another example is the multiple columns on minimum nights.

There exist a lot of information on reviews of the listings. However, on further investigation, a lot of these seem redundant. About 25% of the data either have no review information whatsoever while most of the reviews seem pretty high. The detailed information on reviews are also dropped as most rows are more than 85% of the highest score.

	number_of_reviews	reviews_per_month	number_of_reviews_ltm	number_of_reviews_l30d	review_scores_rating	review_scores_accuracy	review_
count	19978.000000	15868.000000	19978.000000	19978.000000	15475.000000	15456.000000	
mean	21.937631	0.688556	2.415307	0.065272	94.520711	9.665631	
std	47.952994	1.152330	6.868268	0.556867	8.293357	0.807999	
min	0.000000	0.010000	0.000000	0.000000	20.000000	2.000000	
25%	1.000000	0.100000	0.000000	0.000000	93.000000	10.000000	
50%	4.000000	0.270000	0.000000	0.000000	97.000000	10.000000	
75%	18.000000	0.780000	2.000000	0.000000	100.000000	10.000000	
max	618.000000	49.130000	208.000000	34.000000	100.000000	10.000000	

Therefore, only information on the number of reviews, review score rating and reviews per month are selected.

After selecting the columns, rows with null values are dropped. The last review date is used to select only recently active data by only selecting listings that had their last review before 2018. Also, assuming that most people spend a maximum of two weeks holidays, listings with more than 14 days minimum nights are dropped.

Finally the columns are renamed into more convenient names.

To affirm that the features have the right types, the pandas.convert_dtypes() method is used to convert the data types.

```
Data columns (total 17 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   id                    10839 non-null  Int64
 1   superhost             10839 non-null  string
 2   neighbourhood         10839 non-null  string
 3   district              10839 non-null  string
 4   latitude              10839 non-null  float64
 5   longitude             10839 non-null  float64
 6   type                  10839 non-null  string
 7   accommodates          10839 non-null  Int64
 8   bedrooms              10839 non-null  Int64
 9   beds                  10839 non-null  Int64
 10  price                 10839 non-null  Int64
 11  minimum_nights        10839 non-null  Int64
 12  last_review           10839 non-null  string
 13  number_of_reviews     10839 non-null  Int64
 14  reviews_per_month     10839 non-null  float64
 15  review_scores_rating  10839 non-null  Int64
 16  instant_bookable      10839 non-null  string
dtypes: Int64(8), float64(3), string(6)
```

Now having our desired listings, the coordinates of these listings are then used to scrape nearby venues of each listing from Foursquare. It is quite time consuming to get this information, so it is advisable to save this into a file for future use.

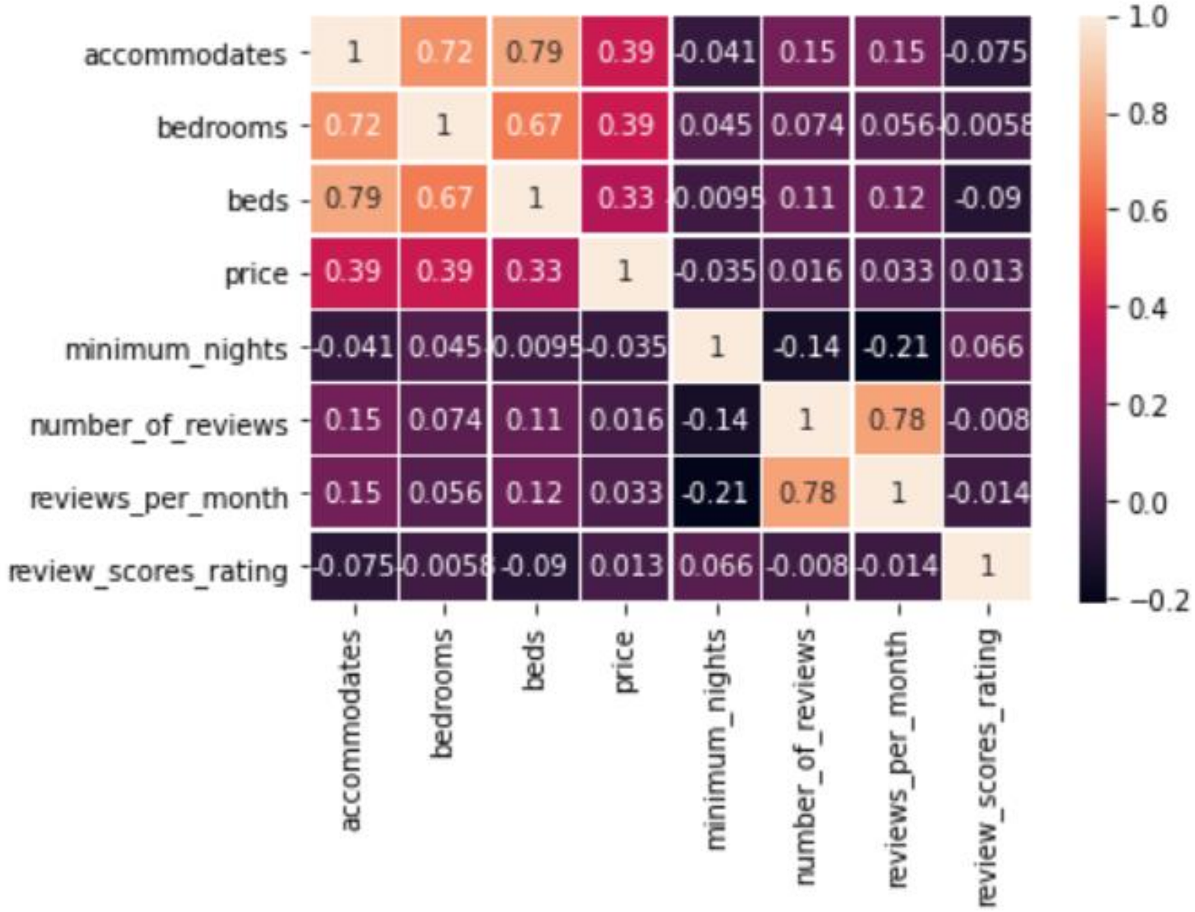
A snippet of the data from foursquare is shown below. The data contains more than 238,450 rows and requires no cleaning yet. However some transformation would be required for analysis.

	id	Neighbourhood Latitude	Neighbourhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
35002	2246464	52.48653	13.43727	Tristeza	52.486598	13.430715	Bar
17292	794284	52.54684	13.41092	Cholila	52.546832	13.403928	Ice Cream Shop
76087	7209443	52.55404	13.40262	avesu	52.551335	13.407923	Shoe Store
177629	18956417	52.50384	13.46436	Wühlischplatz	52.507628	13.464603	Park
215630	21627714	52.48407	13.36114	Spielplatz Cherusker Park	52.480594	13.358692	Playground
215346	21622631	52.51333	13.45827	Al Gazali	52.510514	13.459297	Falafel Restaurant
159235	17262468	52.49793	13.41577	Concierge Coffee	52.496077	13.422149	Coffee Shop
118309	13051894	52.51065	13.29487	Van May	52.512021	13.297199	Vietnamese Restaurant
87780	8589516	52.53819	13.42689	Five Elephant	52.539194	13.421335	Coffee Shop
160450	17383785	52.53692	13.42401	Akemi	52.537620	13.421440	Asian Restaurant

III. Methodology

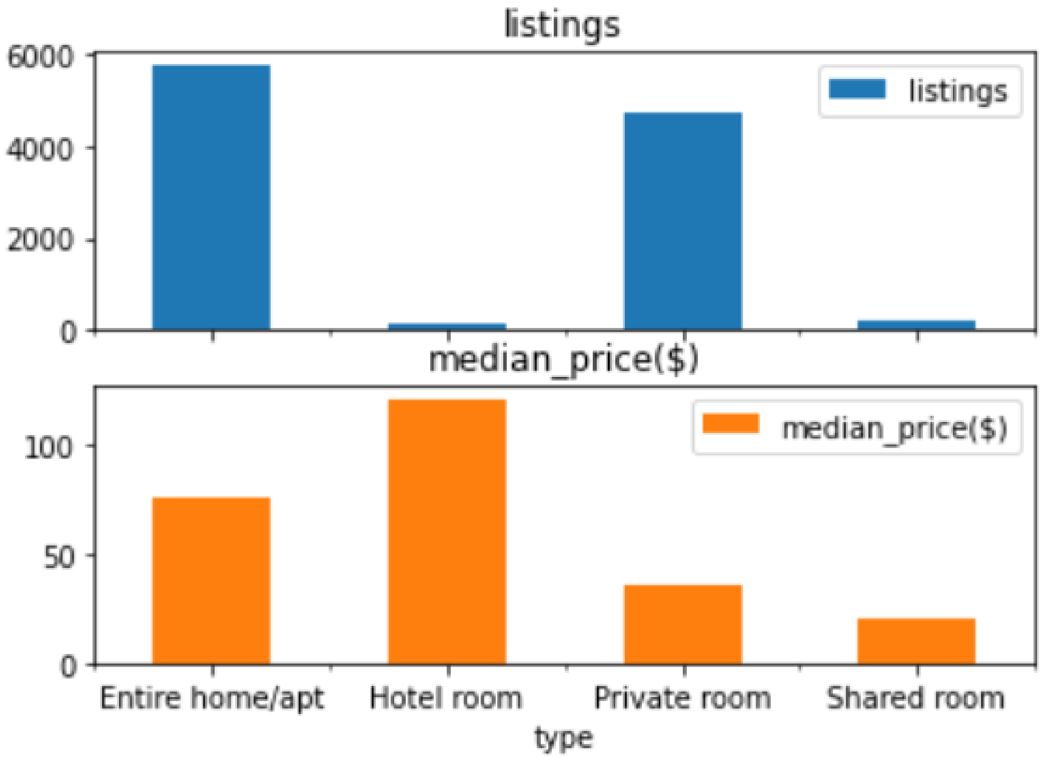
A. Exploratory Analysis

The first in this section is to confirm the correlation between the numerical features of the Airbnb listings. This is achieved using the Pearson correlation metrics. The result is show below, plotted with a seaborn heat map.



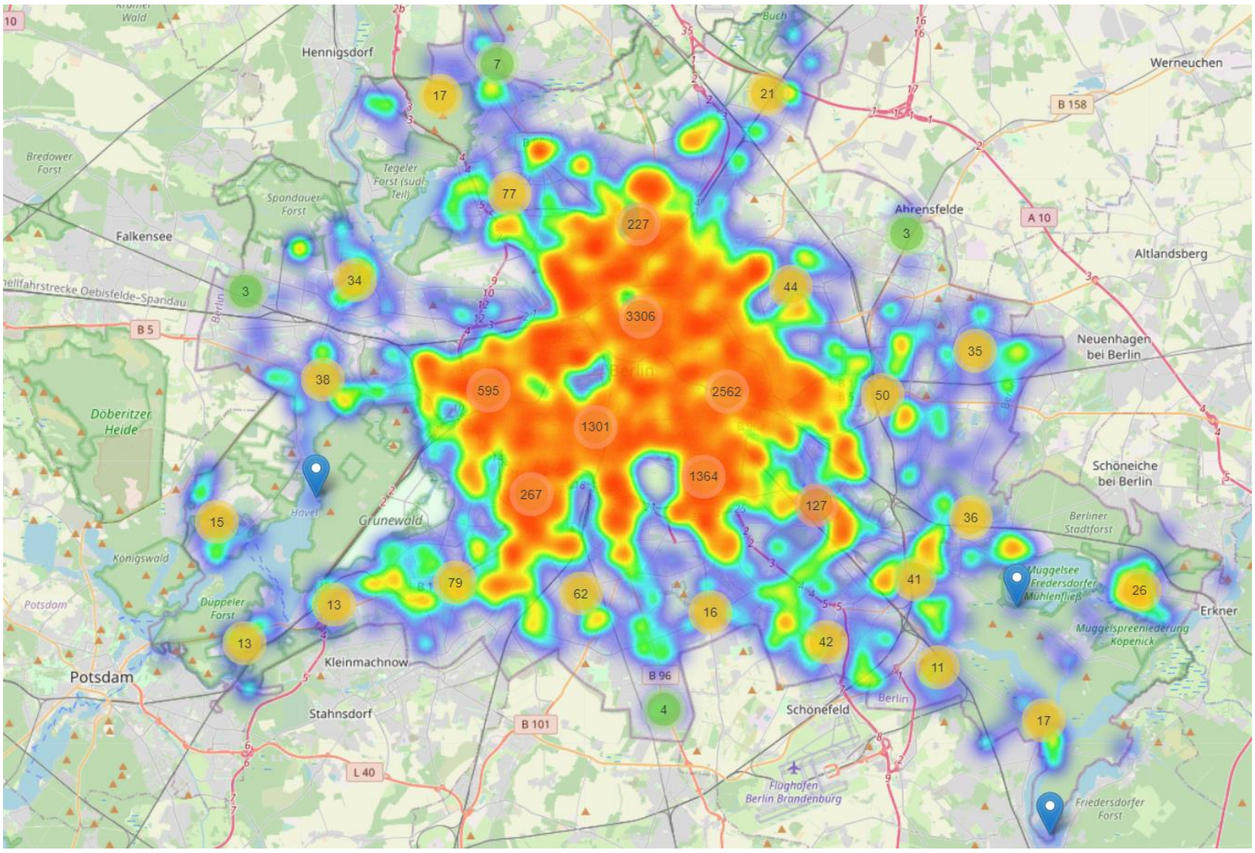
A general low correlation between the data points is observed. The only features with relative significant correlation are the 'accommodates', 'beds' and 'bedrooms' features which basically describe the size of the listings. Theirs is also significant correlation between the number of reviews and the review rate. The correlation map shows that the features cannot be relied on to predict the price and reviews effectively.

With the categorical variables, I decided to compare the median price of each listing type. This is shown in the chart below.

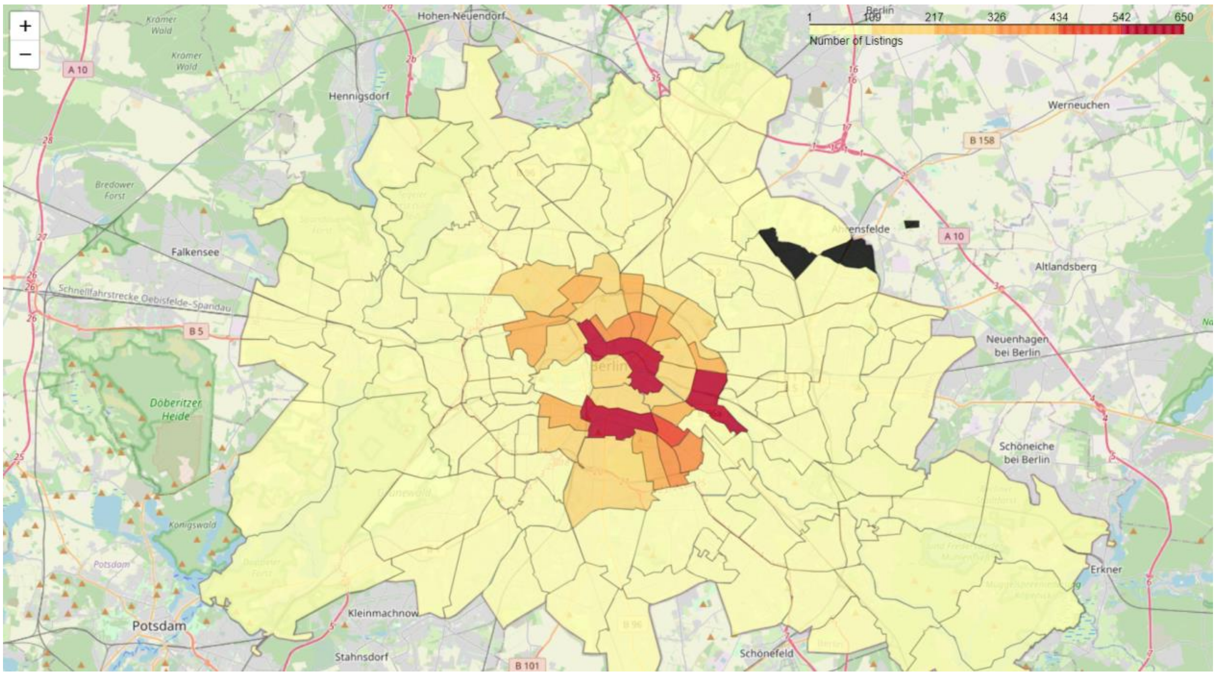


We see that Hotel rooms are more expensive and shared rooms are the cheapest. The price of hotel rooms compared to the other groups demonstrates further the advantages of using Airbnb.

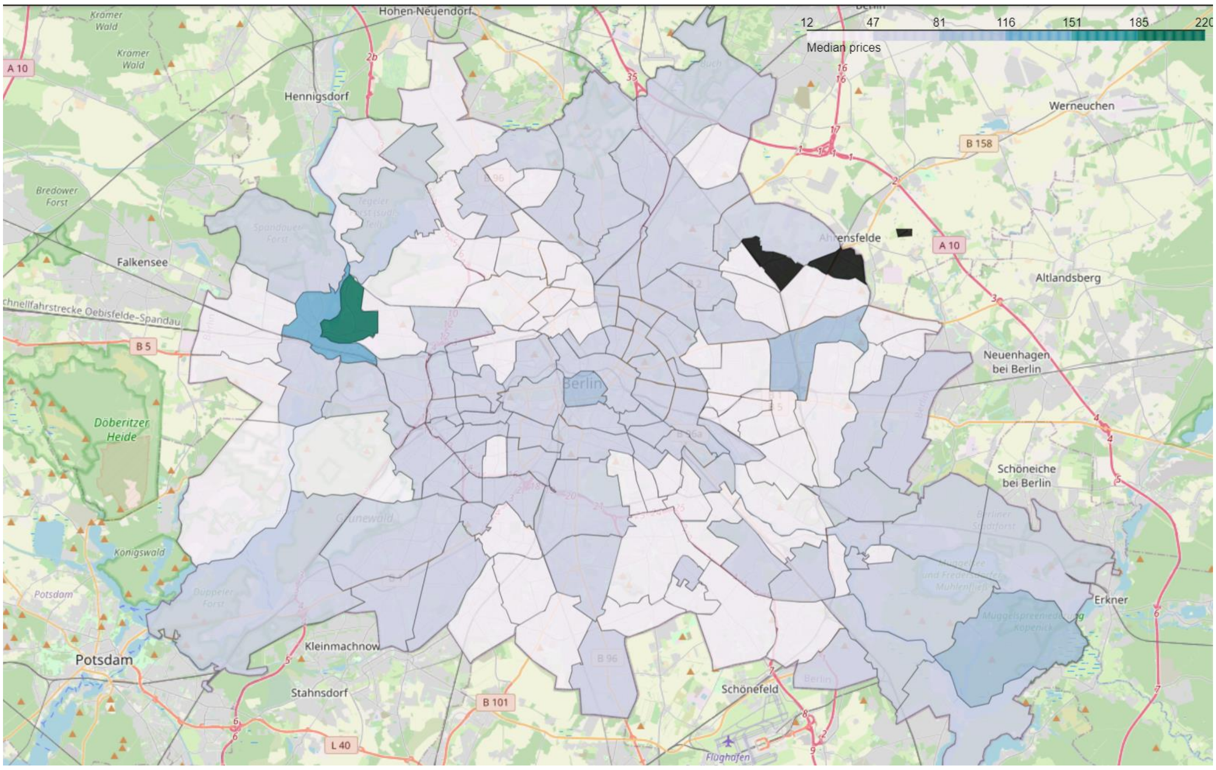
To get an overview, we create a heatmap of the whole distribution of our selected listings across Berlin. This shows that more listings are available as we move towards the city centre.



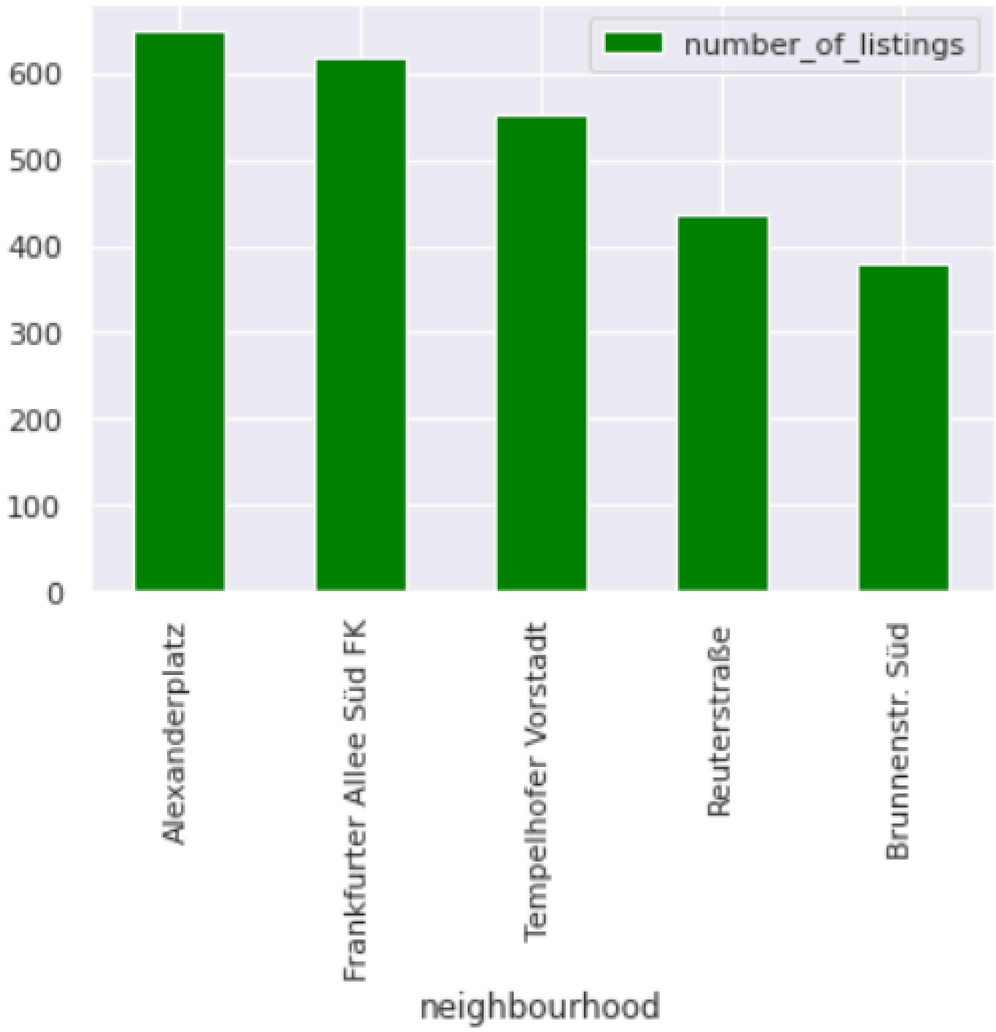
The heatmap doesnt actually clearly show how the listings differ according to the Neighbourhoods (Stadtvierteil) and Districts (Bezirks) in Berlin. Berlin has about 140 Neighbourhoods and 13 Districts. For that we use a GeoJson file and the foliums chlorpeth plug in.



Obviously, the listings increase as we move towards the city centre. This is not the fortunately not the same with the median price with the centre having some of the cheapest prices.



The top 5 neighbourhoods according to number of listings are shown in the chart below



B. Clustering

For clustering the listings, the K-Means clustering algorithm would be used. This algorithms groups the data into a predefined number of clusters. It is quite simple but has the disadvantage of not dealing well with outliers and requiring the number of clusters to predefined. Luckily, there are some methods that can be helpful in determining the clusters including the Elbow, Silhouette and Gap methods. We would use the elbow method

In other to be able to use the clustering algorithms, we first have to make sure that the data is in numerical form. This condition is already satisfied for the numerical features. The categorical features however have to be converted OneHot enconding. This creates dummy variables with the value of 1 for listings that fall under a particular category.

	superhost_f	superhost_t	instant_bookable_f	instant_bookable_t	type_Entire home/apt	type_Hotel room	type_Private room	type_Shared room
0	1	0	1	0	1	0	0	0
1	1	0	1	0	0	0	1	0
2	0	1	1	0	1	0	0	0
3	0	1	0	1	0	0	1	0
4	1	0	1	0	1	0	0	0

Regarding the numerical data, it is always advisable to scale the data before using it in the algorithm. Different scaling methods are available but I choose the MinMaxScaler. This takes an entire column and scales the values to be between 0 and 1. I chose this as it makes all variables to have similar range. Scaling also spreads importance across all features.

	id	accommodates	bedrooms	beds	price	minimum_nights	number_of_reviews	reviews_per_month	review_scores_rating	superhost_f	superhost_t	instant_
0	2015	0.133333	0.000000	0.000000	0.018843	0.166667	0.215559	0.091317	0.9125	1	0	
1	3309	0.000000	0.000000	0.058824	0.008764	0.500000	0.042139	0.011007	0.8625	1	0	
2	6883	0.066667	0.000000	0.058824	0.030675	0.500000	0.217180	0.040359	0.9875	0	1	
3	7071	0.066667	0.000000	0.117647	0.010517	0.000000	0.473258	0.084386	0.9625	0	1	
4	9991	0.400000	0.272727	0.411765	0.074934	0.416667	0.011345	0.004077	1.0000	1	0	
...	...	...	...	...	...	...	...	...	...	...	...	
10834	47844935	0.200000	0.000000	0.176471	0.008764	0.083333	0.000000	0.039951	1.0000	1	0	
10835	47896815	0.066667	0.000000	0.058824	0.020158	0.083333	0.000000	0.039951	1.0000	1	0	
10836	47920604	0.200000	0.090909	0.235294	0.045136	0.083333	0.000000	0.039951	1.0000	1	0	
10837	47949902	0.066667	0.090909	0.176471	0.013585	0.000000	0.000000	0.039951	0.5000	1	0	
10838	48062964	0.200000	0.000000	0.117647	0.022349	0.083333	0.000000	0.039951	1.0000	1	0	

10839 rows × 17 columns

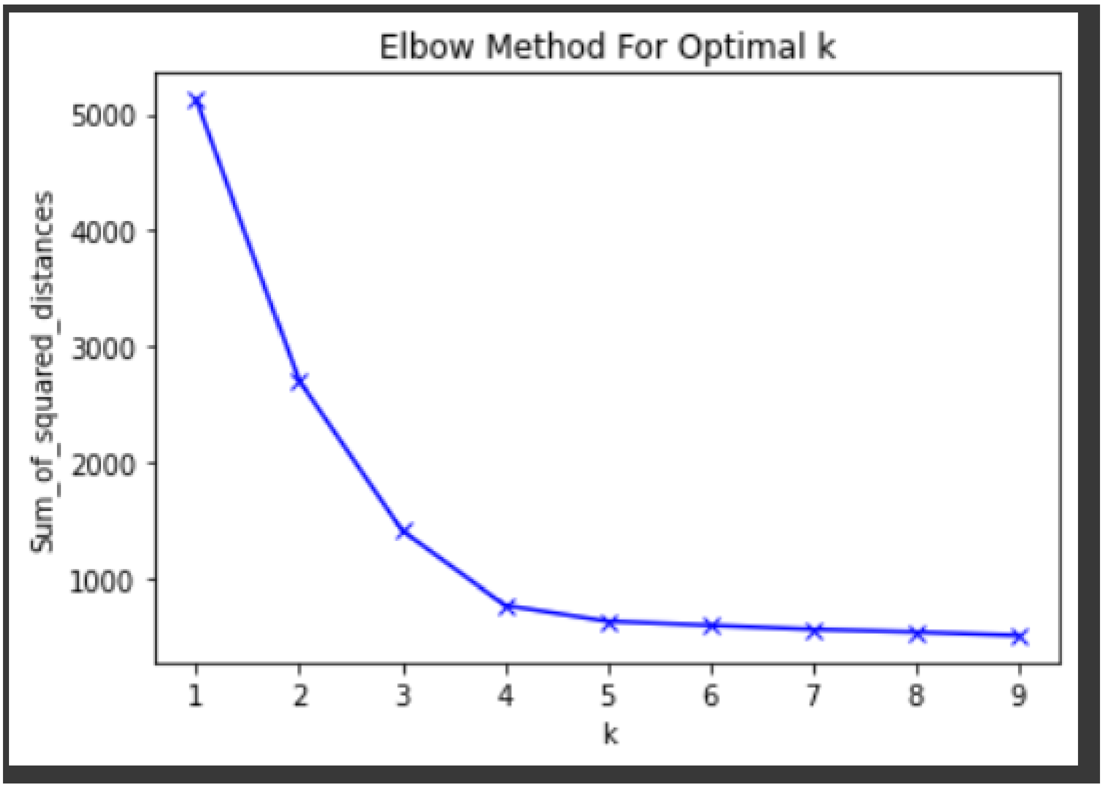
This data is merged with the categorial one hot data. The final part of the clustering data involves grouping in the nearby venues. This is the point where the transformation of the venue data occur. The venues are grouped, the unique categories are converted into one hot variables an merged with the clustering data. This is to create clusters not just with the Airbnb listing characteristics but also the type of venues around the listings.

Another dataframe which has the listings and the top 10 venues around the created for searching through the listings by popular venues nearby.

	id	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue	9th Most Common Venue	10th Most Common Venue
0	2015	Bar	Vegetarian / Vegan Restaurant	Café	Coffee Shop	Italian Restaurant	Park	Vacation Rental	Boutique	Bakery	Pizza Place
1	3309	Café	Cocktail Bar	Coffee Shop	Gay Bar	Men's Store	Hotel	Falafel Restaurant	Indian Restaurant	Organic Grocery	Supermarket
2	6883	Café	Vegetarian / Vegan Restaurant	Coffee Shop	Middle Eastern Restaurant	Thai Restaurant	Italian Restaurant	Plaza	Falafel Restaurant	Bagel Shop	Pizza Place
3	7071	Café	Bar	Vietnamese Restaurant	Bakery	Pizza Place	Ice Cream Shop	Falafel Restaurant	Seafood Restaurant	Doner Restaurant	Pastry Shop
4	9991	Bar	Coffee Shop	Playground	Hotel	Vietnamese Restaurant	Cocktail Bar	French Restaurant	Historic Site	Supermarket	Bakery
...	...	...	...	...	...	...	...	...	...	...	...
4988	23259358	Café	Bakery	Supermarket	Organic Grocery	Beer Garden	Drugstore	Filipino Restaurant	Trattoria/Osteria	Soup Place	Mediterranean Restaurant
4989	23259976	Café	German Restaurant	Plaza	Bakery	Italian Restaurant	Vietnamese Restaurant	Dim Sum Restaurant	Metro Station	Trattoria/Osteria	Supermarket
4990	23263156	Café	Bar	Plaza	Breakfast Spot	Middle Eastern Restaurant	Ice Cream Shop	Park	Cocktail Bar	Grocery Store	Eastern European Restaurant
4991	23264117	Supermarket	Italian Restaurant	Platform	Hotel	Asian Restaurant	Hookah Bar	Vietnamese Restaurant	Bakery	Boat or Ferry	Laser Tag
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	French [illegible]	[illegible]	Italian [illegible]	[illegible]

About half of the listings are dropped due to lack of information about the nearby venues.

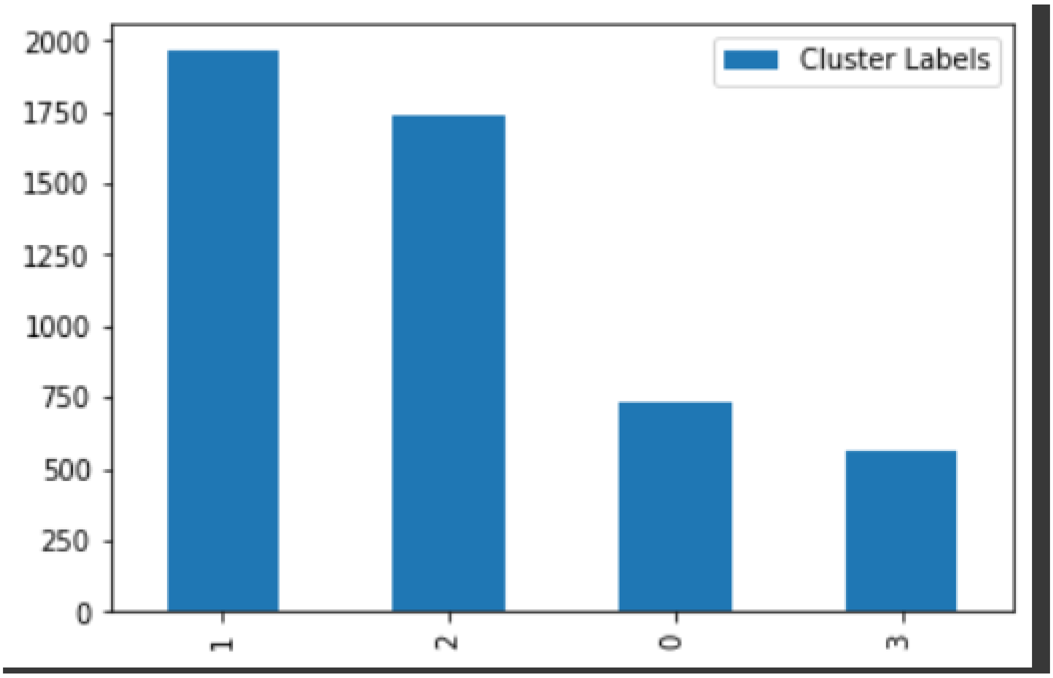
To determine the optimum number of clusters as stated earlier, the elbow method is used. The elbow method graph is shown below.



The results suggest the classification of the listings into 4 clusters.

IV. Results and Discussions

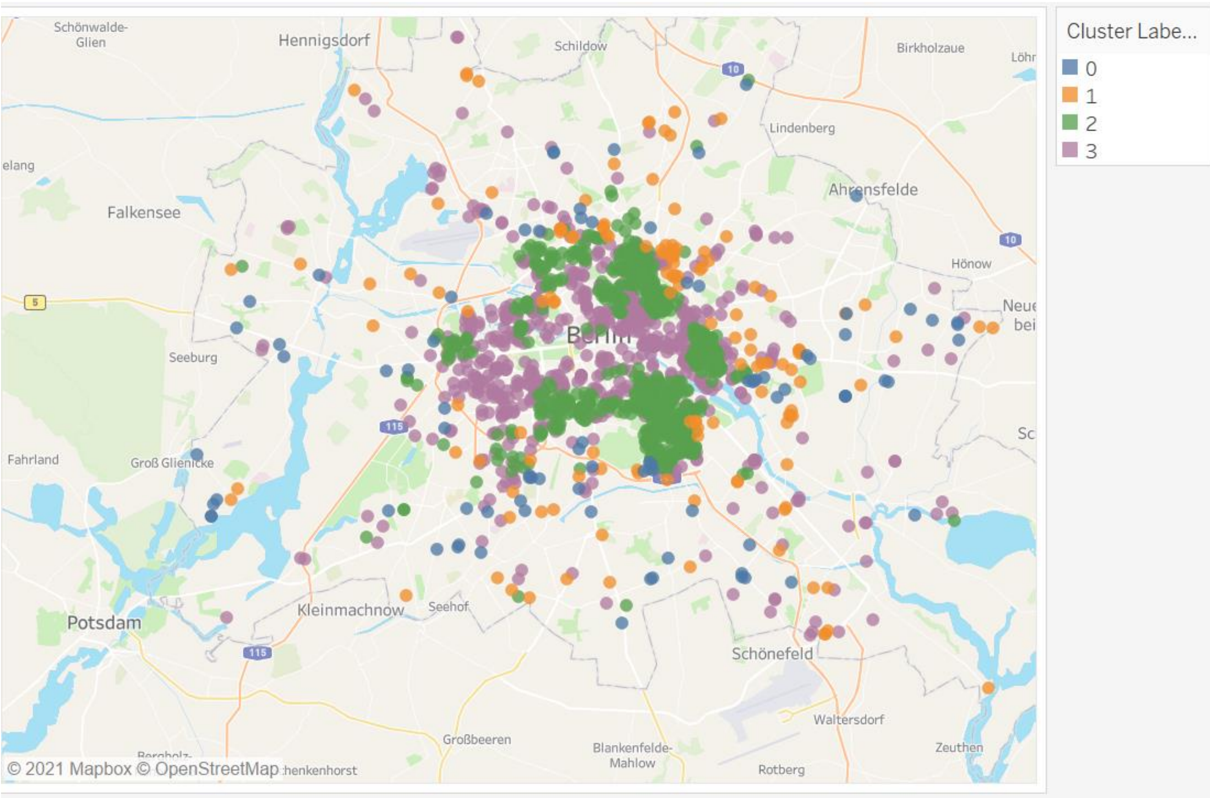
The figure below shows the distribution of the listings among the clusters.



To analyse the properties of the clusters, we compare the median price and reviews per month for the clusters.

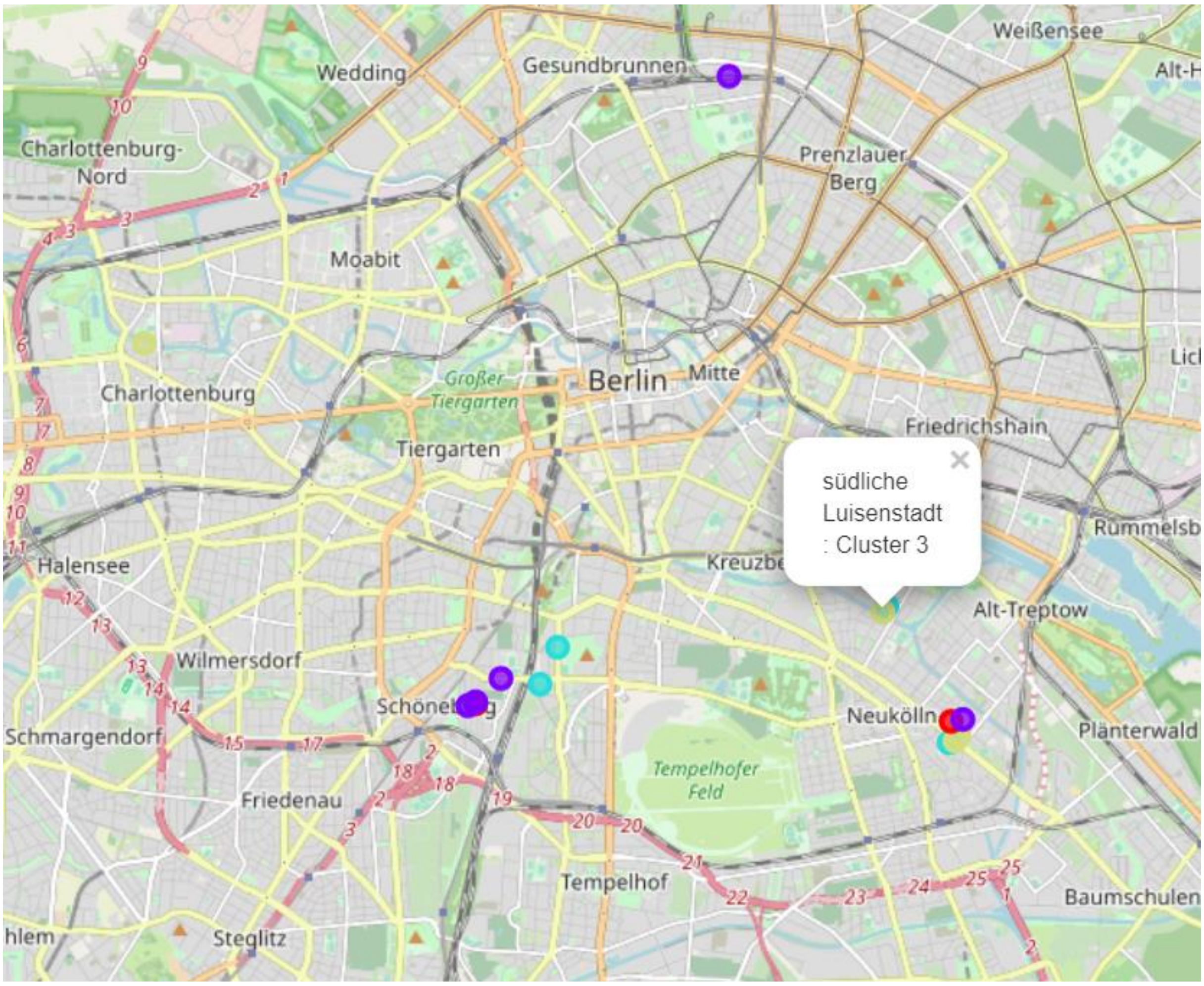
The data shows that that listings in cluster 3 would likely be the best choice for a listing. These listings are cheaper and reviewed more (popular).

We can also plot this on the map to see the geographic location of the clusters.



Observing, the map, it can be inferred that cluster 2 and 3 are more prominent in the city center. However the cheapness of cluster 3 and popularity in addition to location proves that listings in this category would likely be highly sort after.

Finally we try to search through the listings by passing in desired nearby venues to see narrow the search to the best listings. For example, a user who wants to be within walkable distance of a park, a bar, a gym and a supermarket would get the 16 listings shown in the map below.



V. Conclusion

This project has attempted to cluster Airbnb listings and data according to the characteristics of the clusters and the nearby venues. This can easily aid people in narrowing down the search for the perfect accommodation easily.

There is further room for improvement for this work. Other clustering algorithms might be tested and optimised, better feature engineering can also be achieved with PCA algorithm and also further data can be incorporated such as GTFS data. This can be tasks for the future.