Choosing a Suitable Location for living in London using K-means Clustering and Choropleth Maps.

Christian McBride Jordan

11 April 2020

1 Introduction

London is the hub of the UK, it is the centre of beautiful views, west end performances, business conferences and (most importantly) places to eat.

In this project I would like to explore the areas of London by utilising data from the Foursquare API, and use segmentation and clustering to discover the areas of London that are most like his own. Another thing that might affect a persons choice of preferred habitation is rental prices. Therefore I will also prepare data for rental prices in each area and visualise.

2 Data

Firstly, we need some data about the different areas and postal codes in London, which I have managed to find a detailed table on Wikipedia. The table give details of the different boroughs in London and can be referred to below.

`https://en.wikipedia.org/wiki/List_of_areas_of_London`

The Python library **Geopy** provides detailed location data via a search field. As long as we can input a search address which yields the correct results, we can get latitude and longitude data using this function.

`https://geopy.readthedocs.io/en/stable/`

The Foursquare API takes the location data and can yield results for nearby venues. In fact, Foursquare can do a lot more this, but in this task we will be requesting venue data within the areas of London.

`https://foursquare.com`

Next, I found great rental data for 2011 through to 2019 available on the Gov UK website in the form of an excel file.

`https://data.london.gov.uk/dataset/average-private-rents-borough`

Another great source of data to help with this project was found on the Github page below. This provided the GeoJSON file for all of the London boroughs, that is, the coordinates for the boundaries surrounding a particular area.

`https://skgrange.github.io/data.html`

3 Methodology

Clustering is an unsupervised machine learning technique, meaning that the algorithm itself tries to learn from the data without any preparation. In this example, I use the K-means clustering technique to try and arrange different areas of London according to which food venues occur the most frequently within a given radius.

3.1 Webscraping HTML using BeautifulSoup

To begin, we can first obtain the data from the sources outlined in the data section. Using the Python library BeautifulSoup, then strip the appropriate text section from the Wikipedia page.

From here, extracting the correct table containing different borough information for London and converting it to a dataframe was easy. In the end, we end up with a dataframe that looks like Figure 1 below.

	London Borough	Postcode
0	Barking and Dagenham, London, UK	IG11, RM9, RM8, RM9, IG11, RM9, RM10, RM6, RM7
1	Barnet, London, UK	EN5, NW7, NW7, EN5, EN5, NW2, NW4, N11, HA8, N...
2	Bexley, London, UK	SE2, DA5, DA14, DA7, DA1, DA17, DA5, DA6, DA7,...
3	Brent, London, UK	HA0, NW10, NW6, NW10, NW2, NW10, NW10, NW6, HA...
4	Bromley, London, UK	SE20, TN16, BR3, SE20, BR3, TN16, BR1, BR3, BR...

Figure 1: Dataframe containing London areas

3.2 Requesting Location Data using Geopy

This brings us to our next data source, Python library **Geopy**. In the London Borough column of our dataframe, we have the search address readily available to find location data. Iterate through the column *London Borough* finding the longitude and latitude values for each.

Whilst performing these iterations, notice that some of the data extracted from our table is flawed for this task. The two rows which contain London Borough names as following:

+ Haringey and Barnet
+ Kensington and ChelseaHammersmith and Fulham

These places confused the Geopy database and no location results were returned. This is probably because both of the locations contain more than one entry! A quick search of our London Borough column shows that we have the four different locations in our dataframe seperately, so these rows can be dropped. Following these data preparations we are left with the dataframe in Figure 2.

	London Borough	Postcode	Latitude	Longitude
0	Barking and Dagenham, London, UK	IG11, RM9, RM8, RM9, IG11, RM9, RM10, RM6, RM7	51.554117	0.150504
1	Barnet, London, UK	EN5, NW7, NW7, EN5, EN5, NW2, NW4, N11, HA8, N...	51.653090	-0.200226
2	Bexley, London, UK	SE2, DA5, DA14, DA7, DA1, DA17, DA5, DA6, DA7,...	51.441679	0.150488
3	Brent, London, UK	HA0, NW10, NW6, NW10, NW2, NW10, NW10, NW6, HA...	51.563826	-0.275760
4	Bromley, London, UK	SE20, TN16, BR3, SE20, BR3, TN16, BR1, BR3, BR...	51.402805	0.014814

Figure 2: Dataframe updated with lat, long location data

3.3 Requesting Venue Data from Foursquare API

Using the latitude and longitude values found in the last section, we can now use the Foursquare API to request data for food venues in a given radius.

The radius has been set for this project at 1000 meters. Since I only have a free account on Foursquare, I can only obtain a maximum number of 100 results. Some neighbourhoods may not provide this many results, however some may have produced even more if we had a premium account. Therefore, our results may be slightly inaccurate.

Requesting venue data will provide the Venue Name, Venue Category, Latitude and Longitude for each individual venue. Iterating through the London Boroughs, we are able to obtain venue data for each entry in our dataset in the format shown below in Figure 3.

	Borough	Latitude	Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Barking and Dagenham, London, UK	51.554117	0.150504	Lara Grill	51.562445	0.147178	Turkish Restaurant
1	Barking and Dagenham, London, UK	51.554117	0.150504	snack bar	51.552840	0.147773	Café
2	Barking and Dagenham, London, UK	51.554117	0.150504	Heath Pie Shop	51.560414	0.147655	Diner
3	Barking and Dagenham, London, UK	51.554117	0.150504	The Golden Fish	51.560512	0.147334	Fish & Chips Shop
4	Barking and Dagenham, London, UK	51.554117	0.150504	Tarka	51.552033	0.162370	Indian Restaurant

Figure 3: Venue data for each London Borough

A histogram which shows the count of venues returned by the Foursquare API shows us that a large portion of London boroughs returned 90-100 results, and therefore we could have seen more results with a premium account. The same graph shows that there are a number of boroughs with a low number of results, which may also have an adverse on the clustering accuracy.

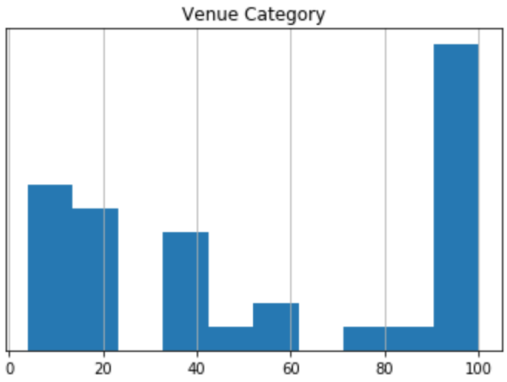

Figure 4: Number of venues in count of venues bracket.

Next, we look at the most commonly returned results to see the most popular restaurants among all London boroughs. From this, we can see that café is the most popular venue with over 200 results. The graph only shows the top 10 occurring categories, and there will be categories with only singular results.

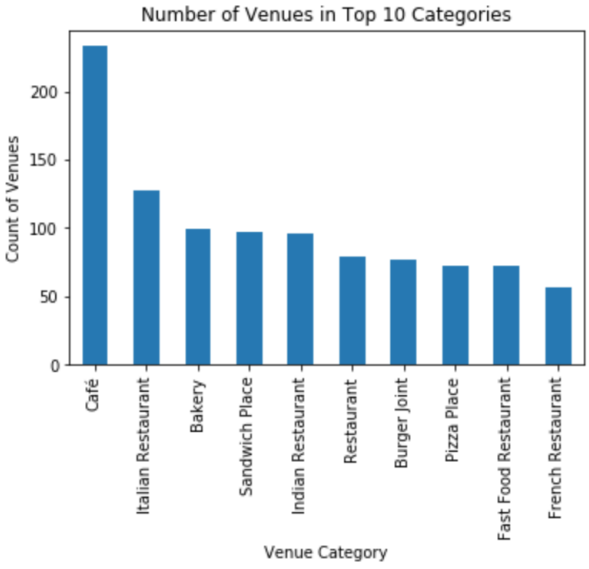

Figure 5: Most commonly occurring venues.

3.4 Data Preparation

Now we have assigned categorical data to our London boroughs which forms venues for each entry, we need to convert this to numerical data. Luckily for us, we can use the **get dummies** function from the Pandas library which will create a column for each venue category and assign numerical data based on whether it exists in a row or not. In short, by using this function and grouping the dataframe to form singular entries for each borough, we can calculate a proportion of total venues each category represents.

For example, looking at the dataframe in Figure 6, African Restaurants in Brent, London, UK represent 0.025641% of the total venues in this area.

	Borough	Afghan Restaurant	African Restaurant	American Restaurant	Argentinian Restaurant	Asian Restaurant	Australian Restaurant	Austrian Restaurant	BBQ Joint	Bagel Shop	...	Sushi Restaurant	Taco Place	Taiwanese Restaurant	Tapas Restaurant
0	Barking and Dagenham, London, UK	0.0	0.000000	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0	...	0.000000	0.0	0.0	0.0
1	Barnet, London, UK	0.0	0.000000	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0	...	0.000000	0.0	0.0	0.0
2	Bexley, London, UK	0.0	0.000000	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0	...	0.000000	0.0	0.0	0.0
3	Brent, London, UK	0.0	0.025641	0.051282	0.0	0.025641	0.0	0.0	0.0	0.0	...	0.000000	0.0	0.0	0.0
4	Bromley, London, UK	0.0	0.000000	0.000000	0.0	0.044444	0.0	0.0	0.0	0.0	...	0.044444	0.0	0.0	0.0

Figure 6: Venue proportion for each London Borough

3.5 Machine Learning using K-Means Clustering

The standard algorithm for K-means clustering, sometimes referred to as Lloyds algorithm, is a simple unsupervised learning technique. The number of clusters in our data must be specified before the algorithm can begin.

Once we have assigned the number of clusters, they are assigned equally spaced within the data. The algorithm is then based on iterations of the two following steps.

1. Each observation is then assigned to the nearest centroid (cluster point), that is, with the nearest Euclidean distance.
2. The centroid is recalculated as the mean of all observations assigned to a cluster.

These two steps are then repeated until the centroids no longer change position. In this project, I decide that using this algorithm with less than 3 clusters would not provide much insight into the data.

The number of clusters that best provide the most information can be estimated using the silhouette score. This is a numeric estimation of how different the clusters are when calculated at each value of k. A silhouette score of 1 would show the number of clusters is perfectly composed, whereas a score of 0 would show that no similarities can be found. See below Figure 7.

From these calculations, we run our algorithm with 3 clusters to gain the most information. The results will be discussed in the results section of this report.

```
Value of k =  3 Silhouette Score =  0.33083400652286155
Value of k =  4 Silhouette Score =  0.26540295761538796
Value of k =  5 Silhouette Score =  0.30343569960069416
Value of k =  6 Silhouette Score =  0.29454247285700436
Value of k =  7 Silhouette Score =  0.25376730464903147
Value of k =  8 Silhouette Score =  0.08455678311305083
Value of k =  9 Silhouette Score =  0.07152137479063492
Value of k =  10 Silhouette Score =  0.0790263483863808
```

Figure 7: Silhouette score for different values of k.

3.6 Average Rental Prices in London

The rental costs for different areas of London were pulled from the Gov UK website, and loaded into a dataframe. From this, we can extract the latest data from 2019 and get an average rental cost for each location.

	Area	Year	Count of rents	Average	Lower quartile	Median	Upper quartile
0	Barking and Dagenham	2019.0	231.428571	1111.142857	1014.000000	1111.428571	1210.714286
1	Barnet	2019.0	804.285714	1512.571429	1242.714286	1385.428571	1592.857143
2	Bexley	2019.0	235.714286	1002.285714	902.857143	1005.000000	1124.571429
3	Brent	2019.0	574.285714	1459.285714	1228.857143	1395.428571	1596.428571
4	Bromley	2019.0	650.000000	1240.714286	1035.714286	1180.000000	1365.714286

Figure 8: Rental prices in London boroughs

We can then use this data and the GeoJSON file acquired from the Github page outlined in the data section to create a choropleth map using the Folium Python package. Again, results of this will be discussed in the results sections of this report.

4 Results

4.1 K-means Clustering Results

So we have our value of k set to 3 in order to gain the most information, and train the algorithm using the Python library **Sklearn**. Fitting the data will then assign a cluster to each observation in the data.

We can then use Python library Folium to visualise a map of London using these results. The results of this can be seen in Figure 9 below.

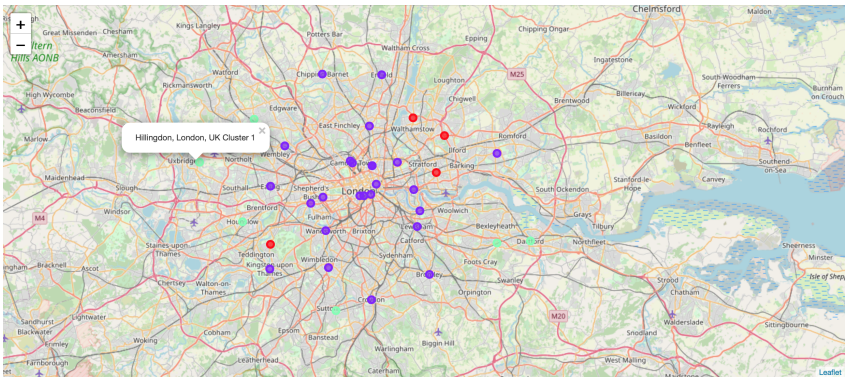

Figure 9: Map of London with Cluster Markers

The next thing we need to do is find out what the different clusters represent, and one way to do this would be to visualise the top venues in each cluster.

I found it quite difficult to find the best way to visualise the results of these clusters individually, and opted for a word cloud built on the number of occurrences of a particular venue category. I started by creating a large string variable of all the venue categories for each cluster.

Then, building a word cloud for each cluster using Python library Wordcloud, which calculates the sizing and positioning of all the words (in this case venue categories). Overleaf in Figure 10 the word clouds are presented for each cluster.

Cafe's are predominant in Cluster 0 and Cluster 2, whereas Indian Restaurants seem common in all clusters. Individuals will be able to make there own observations from these word clouds to depict which cluster locations would be best for them based on preference.

(a) Cluster 0

(b) Cluster 1

(c) Cluster 2

Figure 10: Word Clouds Representing Different Clusters

4.2 Rental Price Results

A choropleth map is a colour coded map where the different shades represent the density of the target variable. In this case, the target variable is average rental price. Using the GeoJSON file outlined in the data section we are able to colour code using the boundaries. The outcome is shown in Figure 11.

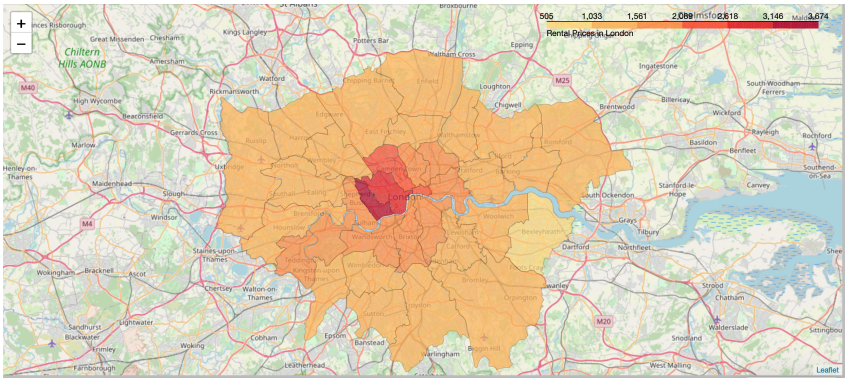

Figure 11: Choropleth Map of London Based on Average Rental Price

As we would expect, the warmest colours which represent the highest rental prices are concentrated towards the centre of London. The cheapest place to live on London based on this plot is Bexleyheath where the most expensive would be Chelsea and Kensington. For anybody who is familiar with costs of housing in London this would make a lot of sense.

4.3 Final Results

The final results will be a formation of the rental price results and the clustering results. An individual who is looking at these results can choose a location from a cluster they see most suitable and then look to select a location that suits their financial needs also.

See Figure 12 for the final resulting interactive Folium map. The full interactive map can be found at ¡address¿.

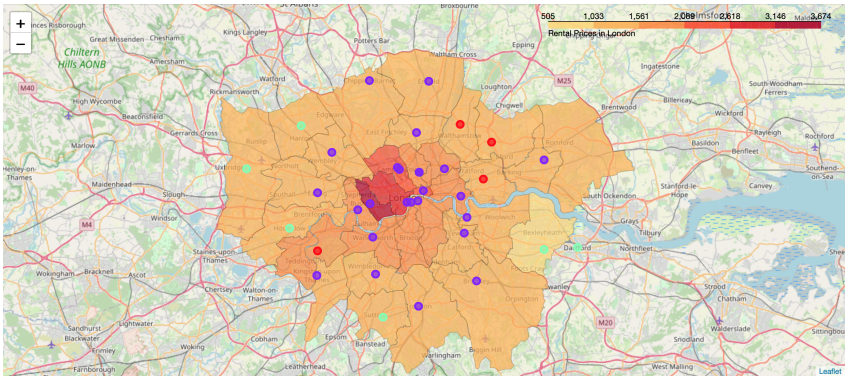

Figure 12: Clustered London Map with Rental Price Density

5 Discussion

In the discussion section I will outline any problems with the accuracy of the machine learning technique or the data that has been collected.

5.1 Effectiveness of Clustering Algorithm

When trying to find the best number of clusters (value of k) to run our K-means algorithm, we found the silhouette score maximum was just above 0.30. This means that the information gain from clustering is not fantastic, which could say that the data is not ideal for clustering.

5.2 Rental Price Accuracy

The rental price data extracted for 2019 includes all rental types, that is, single room, 2+ room occupancy's. If we knew that an individual was looking for single room occupancy, we could alter the data pre-processing to look at only 1 room rentals.

5.3 Search Radius and Venue Results

The search radius was kept low to try and ensure we could find differences between locations so that clustering was most effective. This meant that some queries produced a low number of search (venue) results, whereas others produced a lot more.

If we had a premium Foursquare account we could perhaps search more in depth to find more similarities and differences using a larger search radius.

6 Conclusion

To conclude, K-means clustering is relatively effective in building groups of similar locations using food venues, however we may want to acquire a higher quantity and quality of data in future to ensure our results are best.

Rental prices came from a very reliable source and created a very nice visualisation so that an individual can choose a suitable location based on a preferred food category cluster and rental prices.