Recommender System for Neighbourhoods

Leonardo Ramirez

March 2021

Introduction

Background

Large cities are diverse, having many distinct neighbourhoods where different activities take place and where different demographics tend to live together. Some neighbourhoods are famed for its nightlife and are likely to attract students and people in their 20s for example Shoreditch in London. Families with children might prefer neighbourhoods with good schools, play areas and green spaces, for example Dulwich and Chiswick in London. Some neighbourhoods contain a high concentration of office-based business such as the City of London or Canary Wharf. Some neighbourhoods have many historical places and landmarks and might be preferred by tourists and have a high concentration of hotels and restaurants. Certain neighbourhoods are famous for their shopping, such as Oxford Street in London. There are also neighbourhoods known for their cultural offering such as the South Bank, Barbican or South Kensington in London with venues such as concert halls, museums and theatres.

As we have seen above different neighbourhoods will have different venues and these venues are likely to give us a good indication as to the type of neighbourhood it is. We can then look at the different types of venues as features or characteristics of the particular neighbourhood. With this information we can apply machine learning methods to classify together different neighbourhoods, to cluster different neighbourhoods into different groups and even to create a recommender systems. The aim of this project is to build a neighbourhood recommender system based on a user rating.

Problem

GIven the large size of certain cities it becomes very difficult for individuals to get to know all neighbourhoods. This problem becomes even worse when moving to an unknown but similarly-sized city in a different region or country.

Interest

The recommender system will be of interest to individuals and certain businesses
For example if a lecturer changes jobs from a University in West London to one in East London, she might want to live in a similar neighbourhood but closer to work. The recommender system for neighbourhoods will be a great help for him. While a family might be very happy with their neighbourhood in London, if they had to move to Toronto they will not know which neighbourhood to choose from. Once again the recommender system can help them make a better choice about neighbourhoods in the new city.

A business-owner such as a dry-cleaner planning on expanding his business by opening a new shop could use the recommender system for neighbourhoods. His existing shop might be in a neighbourhood with many coffee shops and restaurants but few landmarks or businesses. A neighbourhood with such characteristics might indicate a high-income demographic who will have a high demand for dry-cleaning services (as opposed to a neighbourhood with many coffee shops and restaurants but lots of landmarks which is likely to have a high transient demographic like tourists who do not use many dry-cleaning services). Therefore the recommender system will help the business owner make a choice as to what neighbour it will be profitable to open a new shop. It can also be used by real estate agents suggesting new neighbourhoods to customers. It can also be used by real estate investors looking for similar neighborhoods.

Data

Data Sources

We are aiming to build a recommender system for neighbourhoods in a city. So the first step we need is to choose a city and then define the neighbourhoods within that city. I have chosen London as the city I will investigate. During the course we have already looked at the cities of New York and Toronto along with their neighbourhoods. The second step is to get venues in each neighbourhood.

Defining Neighbourhoods - Wikipedia

In London (and the whole of the United Kingdom) every place has a postcode. This is alphanumeric and varies in length between 6 and 8 characters (including a space separating the outward and inward code). The postcode is composed of several parts:

POSTCODE			
Outward code		Inward Code	
Area	District	Sector	Unit
SW	1W	0	NY

The Area in the Outward code indicates a large geographic region, a city or part of a large city. There are several Areas in London. The Outward code, i.e. the Area together with the District generally indicate a neighbourhood. We have chosen to use the Outward code as a single district. Note Outwards codes vary in size but geographically smaller Outward codes tend to be more densely populated so we can safely assume Outward codes identify neighbourhoods.

We obtained the London Area Postcodes from Wikipedia[1]. There were 8 Area Codes for London (references [2] to [9]) each with several Districts making up a total of 170 Neighbourhoods. The neighbourhoods were stored in a dataframe London_df:

```
London_df.head()
```

	Postcode district	Post town	Coverage	Local authority area(s)
0	E1	LONDON	Algate	Tower Hamlets, Hackney, City of London
1	E1W	LONDON	Wapping	Tower Hamlets
2	E2	LONDON	Bethnal Green	Tower Hamlets, Hackney
3	E3	LONDON	Bow	Tower Hamlets, Newham
4	E4	LONDON	Chingford	Waltham Forest, Enfield, Epping Forest (Essex)

Finding Neighbourhood coordinates - FreeMapTools

Once we had the list of neighbourhoods, we had to find a datasource that would return the latitude and longitude of each postcode district. For this purpose we downloaded a csv file containing the outward code from FreeMapTools.com. Note data attribution statements and data licences in ref [10]. Once imported this file contained the outwards code and the latitude and longitude coordinates as shown below:

```
lon_coor.head()
```

	postcode	latitude	longitude
0	AB10	57.13514	-2.11731
1	AB11	57.13875	-2.09089
2	AB12	57.10100	-2.11060
3	AB13	57.10801	-2.23776
4	AB14	57.10076	-2.27073

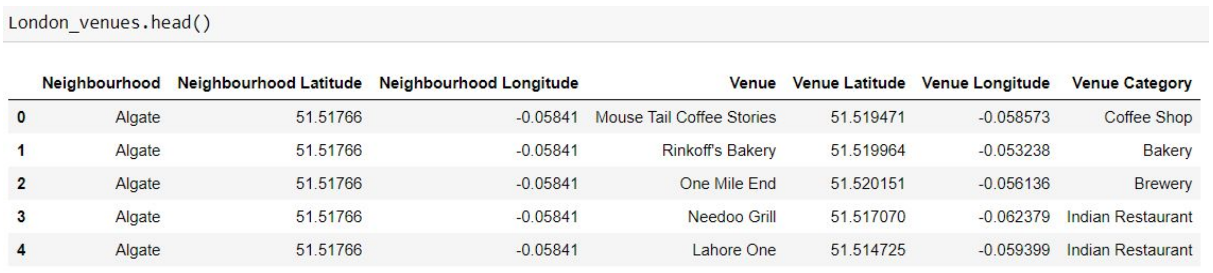
These two tables were joined to show the London Neighbourhoods with their coordinates:

```
London_df.head()
```

	Postcode	City	Neighbourhood	Borough	Latitude	Longitude
0	E1	LONDON	Algate	Tower Hamlets, Hackney, City of London	51.51766	-0.05841
1	E1W	LONDON	Wapping	Tower Hamlets	51.50775	-0.05739
2	E2	LONDON	Bethnal Green	Tower Hamlets, Hackney	51.52939	-0.06080
3	E3	LONDON	Bow	Tower Hamlets, Newham	51.52789	-0.02482
4	E4	LONDON	Chingford	Waltham Forest, Enfield, Epping Forest (Essex)	51.62196	-0.00339

Finding Neighbourhood Venues - FOURSQUARE API

Once we had a list of neighbourhoods and their respective centre coordinates, the next step was to find information about the venues in each neighborhood. This was done using the FOURSQUARE API [11] as described in the course. We used the coordinates of each neighbourhood to find the first 100 venues within a 500 metre radius of the coordinates.



```
London_venues.head()
```

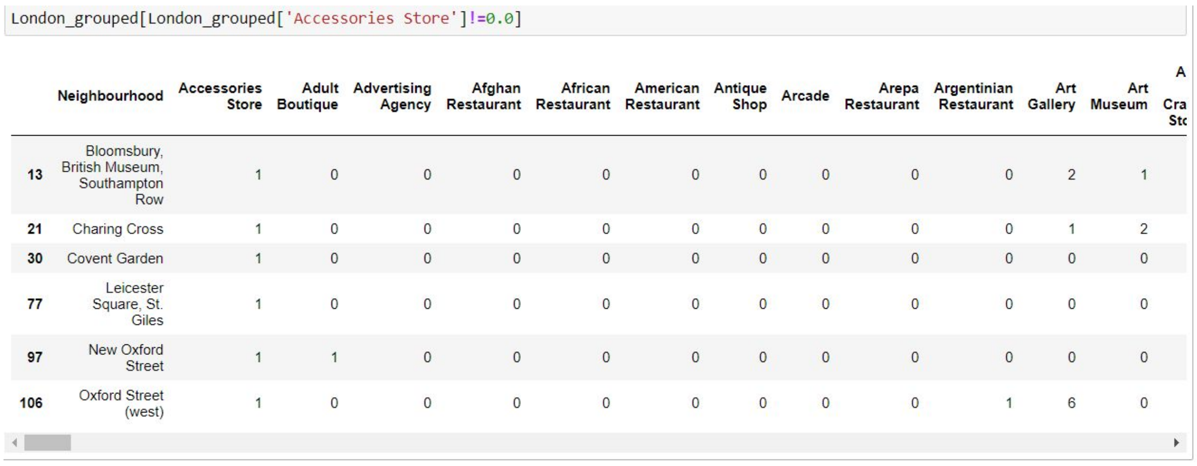
	Neighbourhood	Neighbourhood Latitude	Neighbourhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Algate	51.51766	-0.05841	Mouse Tail Coffee Stories	51.519471	-0.058573	Coffee Shop
1	Algate	51.51766	-0.05841	Rinkoff's Bakery	51.519964	-0.053238	Bakery
2	Algate	51.51766	-0.05841	One Mile End	51.520151	-0.056136	Brewery
3	Algate	51.51766	-0.05841	Needoo Grill	51.517070	-0.062379	Indian Restaurant
4	Algate	51.51766	-0.05841	Lahore One	51.514725	-0.059399	Indian Restaurant

Data Cleaning

As there were 8 different postcode Areas in London each with a different Wikipedia page, we had to scrape all 8 different pages. The data cleaning process initial step involved removing non-geographic postcodes (special postcodes assigned to a particular building). The second step involved removing the text after “district:” in each neighbourhood as this just added unnecessary information about that particular neighbourhood. The third step involved giving names to all the subdistricts (e.g. SW1V) falling within a head district (SW1?). For example postcode district “SW1V” was assigned Borough or neighbourhood name “Victoria”. To complete this task you need context knowledge but this information is not necessary as we could simply refer to neighbourhoods by the postcode outward code (Area code together with District code). Once each of the 8 postcode Areas in London were cleaned they were all stacked together to form a large dataframe London_df as shown above.

The coordinates were added by joining on the postcode outward codes (Area code together with District code) from the csv file which returned the table shown earlier.

From the FOURSQUARE API all the venues in each neighbourhood were obtained using the coordinates. One hot encoding was performed on the venue categories. All venue categories were summed over the neighbourhood so as to keep a count of each venue category. For example there were 6 neighbourhoods with “Accessories Stores”:



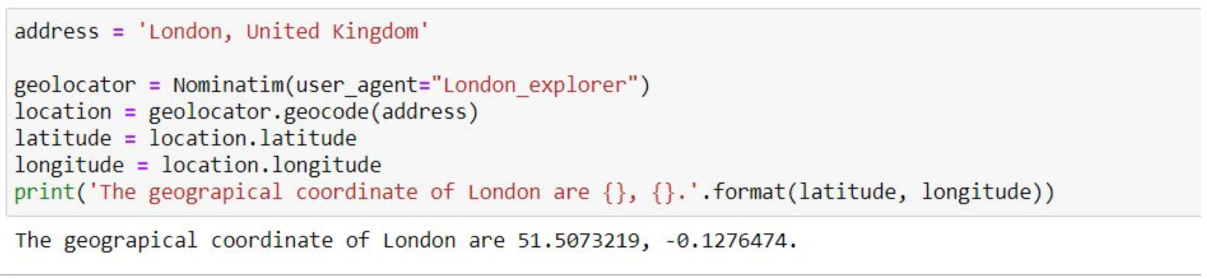
```
London_grouped[London_grouped['Accessories Store']!=0.0]
```

	Neighbourhood	Accessories Store	Adult Boutique	Advertising Agency	Afghan Restaurant	African Restaurant	American Restaurant	Antique Shop	Arcade	Arepa Restaurant	Argentinian Restaurant	Art Gallery	Art Museum	A Cra Stc
13	Bloomsbury, British Museum, Southampton Row	1	0	0	0	0	0	0	0	0	0	2	1	
21	Charing Cross	1	0	0	0	0	0	0	0	0	0	1	2	
30	Covent Garden	1	0	0	0	0	0	0	0	0	0	0	0	
77	Leicester Square, St. Giles	1	0	0	0	0	0	0	0	0	0	0	0	
97	New Oxford Street	1	1	0	0	0	0	0	0	0	0	0	0	
106	Oxford Street (west)	1	0	0	0	0	0	0	0	0	1	6	0	

We can also see that Oxford Street (West) has 6 art galleries but this exploratory analysis will be developed later on.

Exploratory Data Analysis

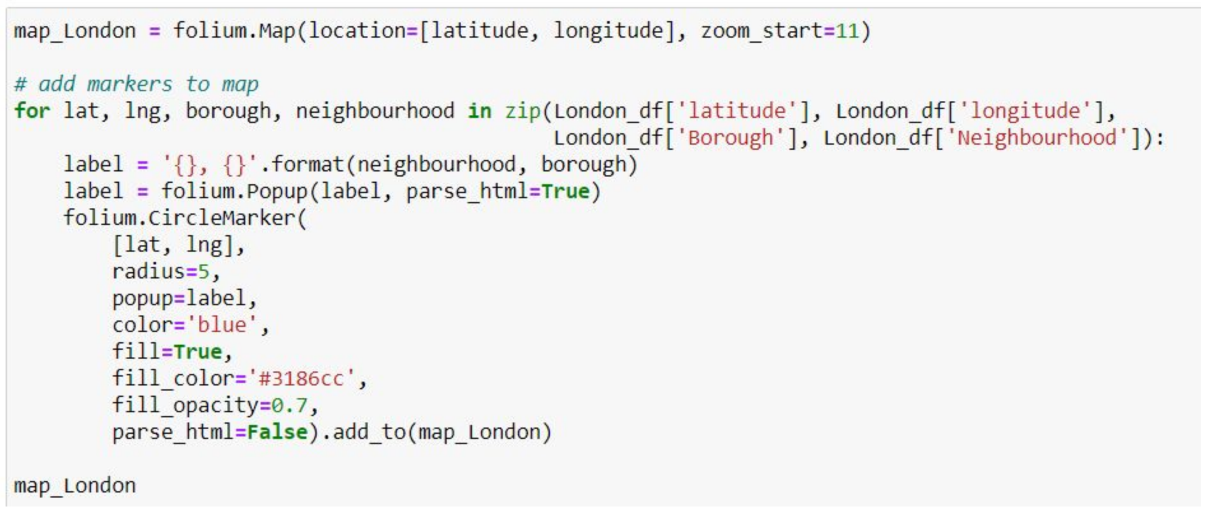
As we are exploring London Neighbourhoods, we created a map of London with all the neighbourhoods we have found from the postcodes. We used the python Folium library to visualize geographic details.



```python
address = 'London, United Kingdom'

geolocator = Nominatim(user_agent="London_explorer")
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude
print('The geograpical coordinate of London are {}, {}.'.format(latitude, longitude))
```

```
The geograpical coordinate of London are 51.5073219, -0.1276474.
```

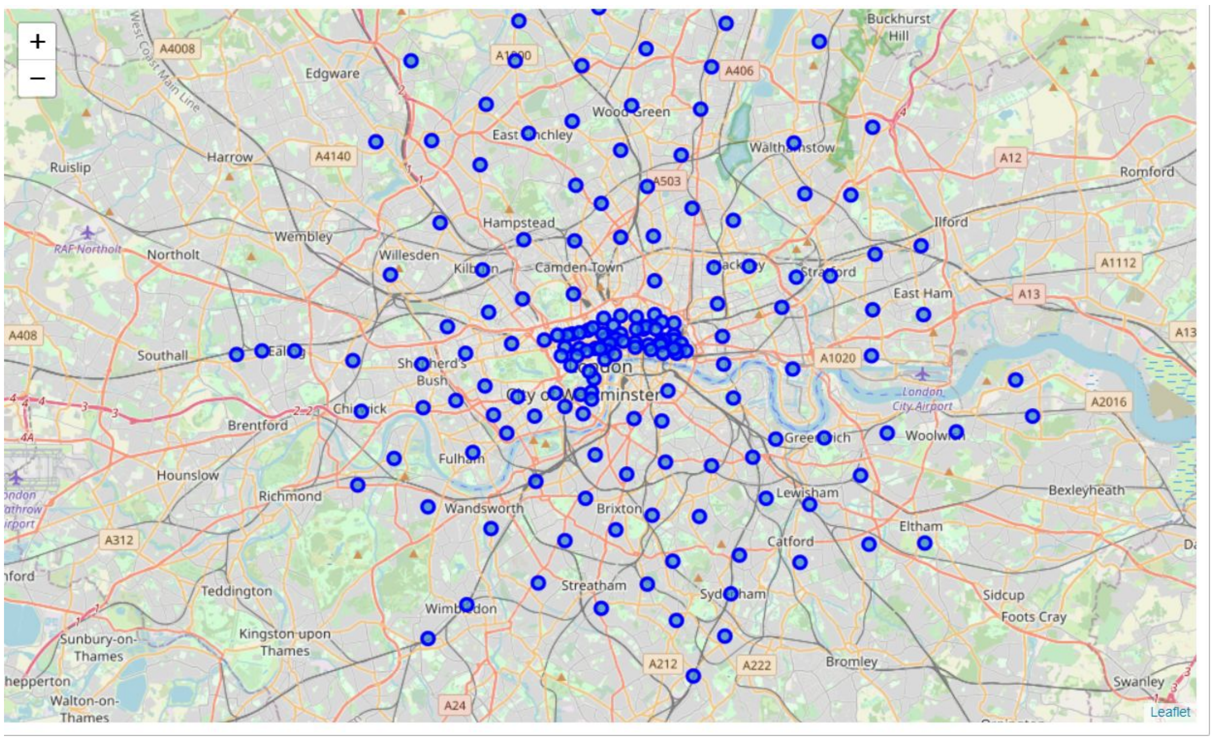


```python
map_London = folium.Map(location=[latitude, longitude], zoom_start=11)

# add markers to map
for lat, lng, borough, neighbourhood in zip(London_df['latitude'], London_df['longitude'],
                                            London_df['Borough'], London_df['Neighbourhood']):
    label = '{}, {}'.format(neighbourhood, borough)
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius=5,
        popup=label,
        color='blue',
        fill=True,
        fill_color='#3186cc',
        fill_opacity=0.7,
        parse_html=False).add_to(map_London)

map_London
```

This returned the following map:



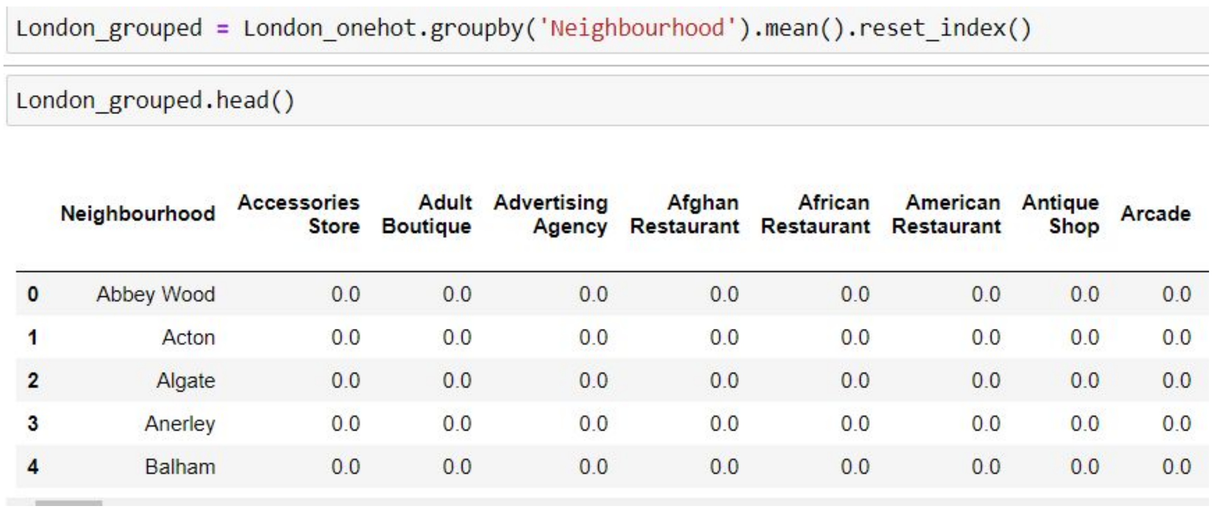
We can also see which are the most commons venues in London, thus:

```
print('The most common venues in London are:')
London_venues['Venue Category'].value_counts().head(20).to_frame()
```

```
The most common venues in London are:
```

	Venue Category
Coffee Shop	665
Pub	523
Café	497
Hotel	393
Italian Restaurant	365
Sandwich Place	232
Gym / Fitness Center	223
Grocery Store	218
Bakery	213
Restaurant	191
Cocktail Bar	183
Pizza Place	175
Clothing Store	175
Indian Restaurant	173
French Restaurant	171
Park	152
Bar	136
Fast Food Restaurant	128
Bus Stop	117
Burger Joint	116

We also carried out a clustering analysis to see in how many clusters we could categorise all the London Neighbourhoods. The first step was to perform a one hot encoding of all the venues per neighbourhood and then group by the neighbourhood aggregating by the mean over all venues. This returned the London_grouped dataframe as shown below:

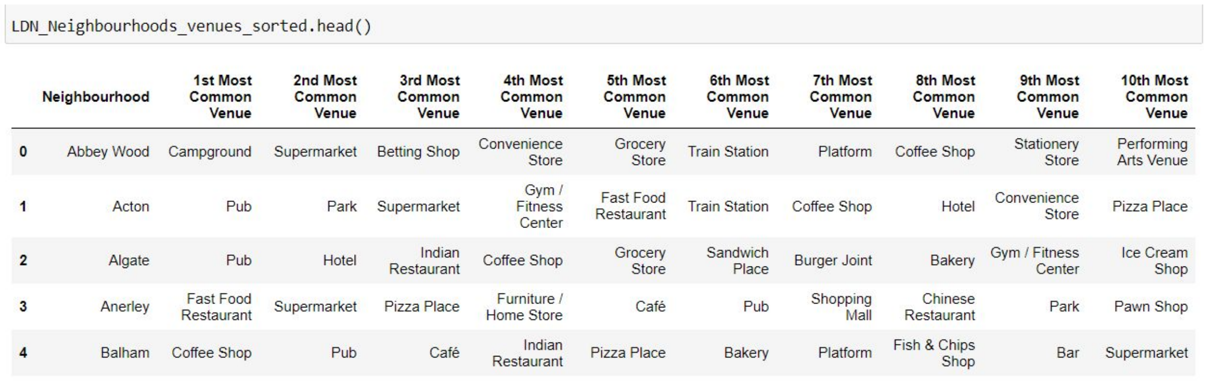


```
London_grouped = London_onehot.groupby('Neighbourhood').mean().reset_index()
```

```
London_grouped.head()
```

	Neighbourhood	Accessories Store	Adult Boutique	Advertising Agency	Afghan Restaurant	African Restaurant	American Restaurant	Antique Shop	Arcade
0	Abbey Wood	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	Acton	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	Algate	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	Anerley	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	Balham	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

With this information we were then able to create a function to return the 10 most common venues per neighbourhood. As the preview of the result shows pubs and pubs and coffee shops/cafés are very popular:



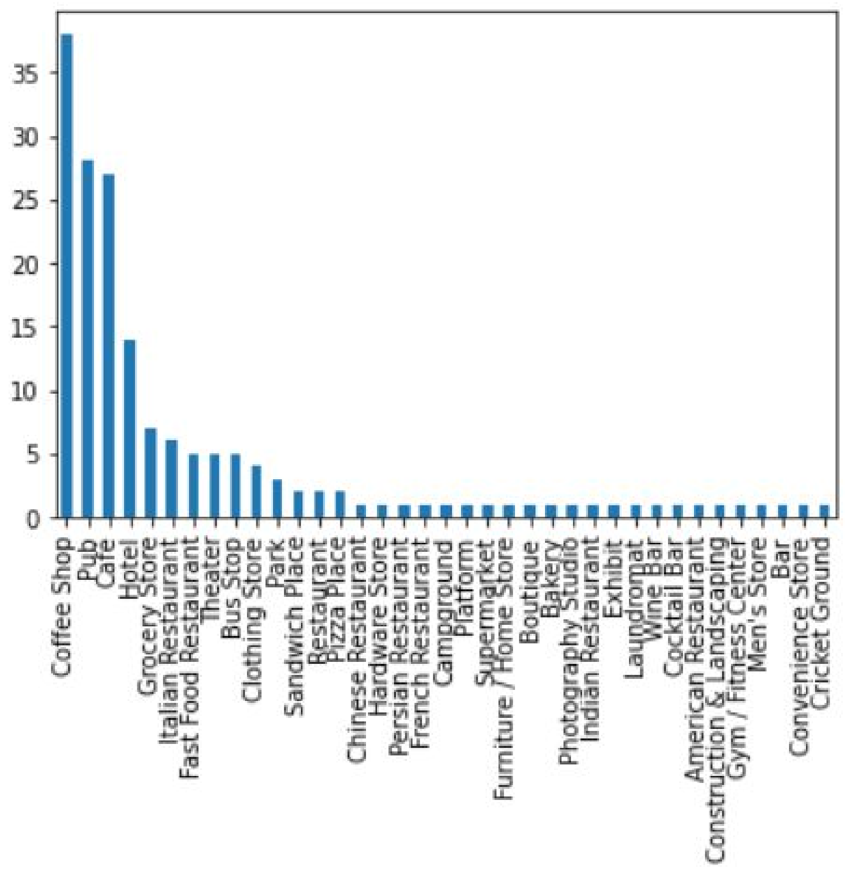
```
LDN_Neighbourhoods_venues_sorted.head()
```

	Neighbourhood	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	Abbey Wood	Campground	Supermarket	Betting Shop	Convenience Store	Grocery Store	Train Station	Platform	Coffee Shop	Stationery Store	Performing Arts Venue
1	Acton	Pub	Park	Supermarket	Gym / Fitness Center	Fast Food Restaurant	Train Station	Coffee Shop	Hotel	Convenience Store	Pizza Place
2	Algate	Pub	Hotel	Indian Restaurant	Coffee Shop	Grocery Store	Sandwich Place	Burger Joint	Bakery	Gym / Fitness Center	Ice Cream Shop
3	Anerley	Fast Food Restaurant	Supermarket	Pizza Place	Furniture / Home Store	Café	Pub	Shopping Mall	Chinese Restaurant	Park	Pawn Shop
4	Balham	Coffee Shop	Pub	Café	Indian Restaurant	Pizza Place	Bakery	Platform	Fish & Chips Shop	Bar	Supermarket

In fact when we plot the 1st most common venue we find again that coffee shops/cafés and pubs are very popular.

```
London_merged['1st Most Common Venue'].value_counts().plot.bar()
```

```
<AxesSubplot:>
```



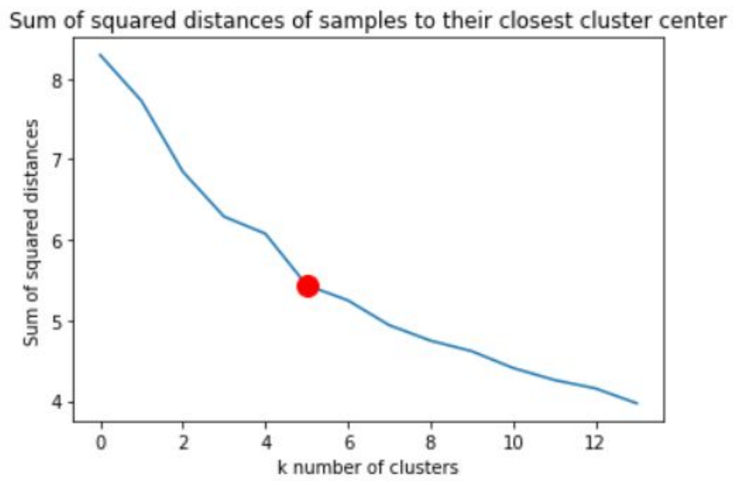
We then carried out clustering analysis for a range of clusters k, from k=1 to 15. We saved the sum of squared distances for all values of k to find the optimum number of clusters.

We chose k=5 as the optimum number of clusters as this is where we see a kink in the plot of k vs sum of squared distances as shown below:

```python
import matplotlib.pyplot as plt
plt.plot(Sum_of_squared_distances)
plt.xlabel('k number of clusters')
plt.ylabel('Sum of squared distances')
plt.title('Sum of squared distances of samples to their closest cluster center')

plt.plot( [5.0],[Sum_of_squared_distances[5]], marker='o', markersize=12, color = 'red')
print('We will use number of clusters k = 5')
```

```
We will use number of clusters k = 5
```



We then created a new dataframe showing all neighbourhoods along with the cluster label and the 10 most common venues:

```python
London_merged = London_df.join(LDN_Neighbourhoods_venues_sorted.set_index('Neighbourhood'), on='Neighbourhood')
```

```python
London_merged.head()
```

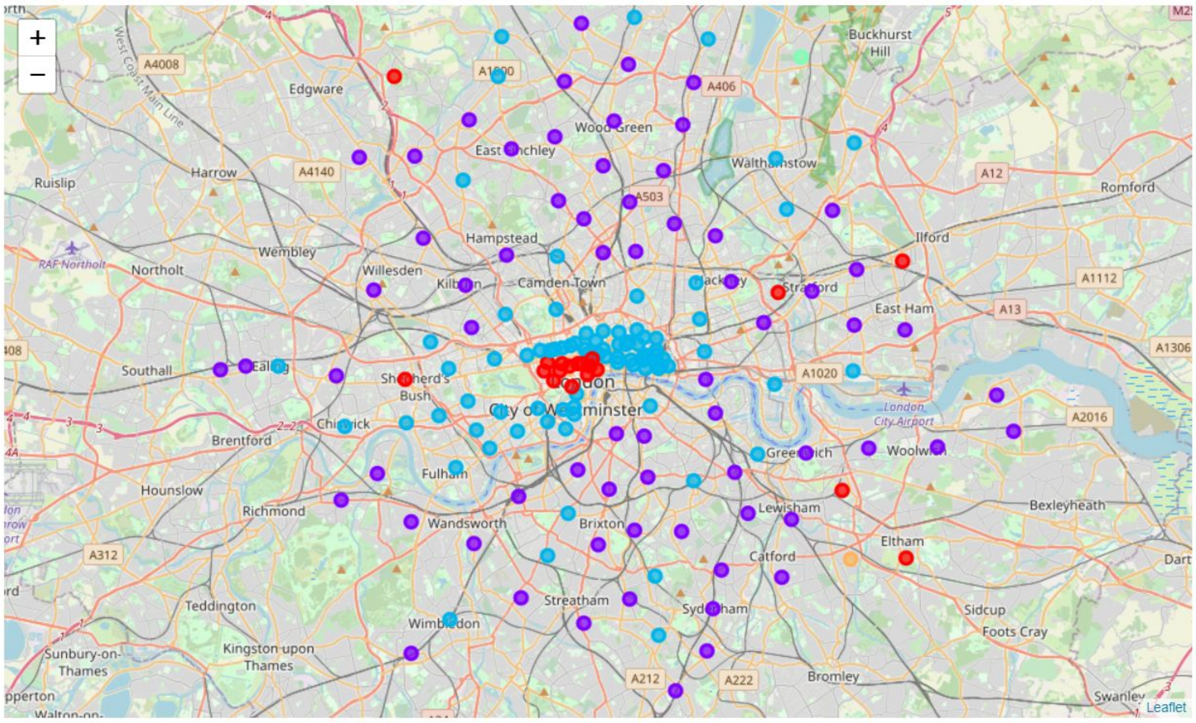
	Postcode	City	Neighbourhood	Borough	latitude	longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Mos Commor Venue
0	E1	LONDON	Algate	Tower Hamlets, Hackney, City of London	51.51766	-0.05841	2	Pub	Hotel	Indian Restaurant	Coffee Shop	Grocery Store	Sandwich Place	Burger Join
1	E1W	LONDON	Wapping	Tower Hamlets	51.50775	-0.05739	1	Pub	Coffee Shop	Grocery Store	Fast Food Restaurant	Gym / Fitness Center	Italian Restaurant	Convenience Store
2	E2	LONDON	Bethnal Green	Tower Hamlets, Hackney	51.52939	-0.06080	2	Coffee Shop	Café	Pub	Cocktail Bar	Grocery Store	Park	Hote
3	E3	LONDON	Bow	Tower Hamlets, Newham	51.52789	-0.02482	1	Grocery Store	Bus Stop	Hotel	Pub	Café	Locksmith	Coffee Shop
4	E4	LONDON	Chingford	Waltham Forest, Enfield, Epping Forest (Essex)	51.62196	-0.00339	3	American Restaurant	Adult Boutique	Pakistani Restaurant	Pet Store	Peruvian Restaurant	Persian Restaurant	Perfume Shop

We can also observe that there are three large clusters (with several neighbourhoods each) and two clusters having only one neighbourhood.

```
Neighbourhoods_per_cluster = London_merged['Cluster Labels'].value_counts().sort_index().to_frame()
Neighbourhoods_per_cluster.columns = ['No. of Neighbourhoods per Neighbourhood cluster']
Neighbourhoods_per_cluster
```

	No. of Neighbourhoods per Neighbourhood cluster
0	19
1	67
2	83
3	1
4	1

Finally we display a map of the neighbourhoods colour coded by clusters:

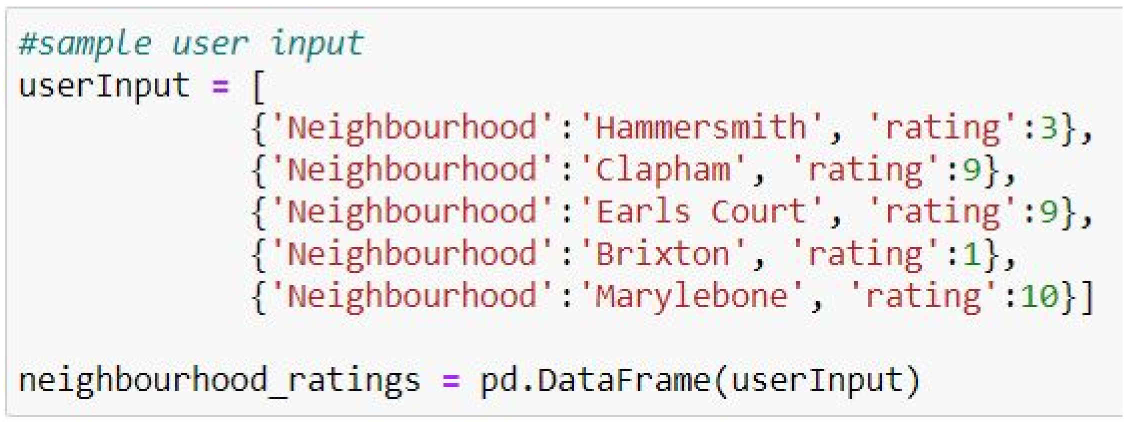


Methodology

We have already explored the k-means clustering methodology above including iterating over different values of k to find the optimum k.

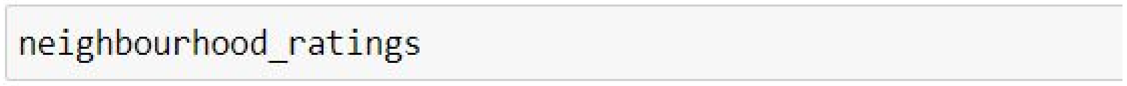
The way this user recommendation system works is very simple.

First the user will rate the neighbourhoods that he is familiar with according to his own preferences. We assume that his preferences on neighbourhoods are based on the number of venues available in those neighbourhoods.

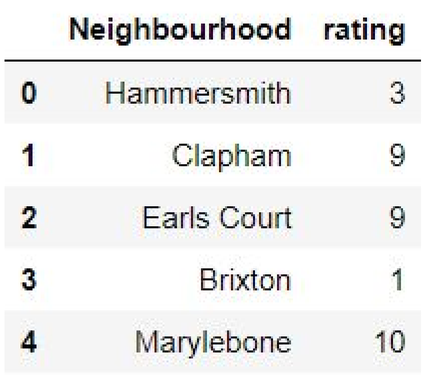


```
#sample user input
userInput = [
            {'Neighbourhood':'Hammersmith', 'rating':3},
            {'Neighbourhood':'Clapham', 'rating':9},
            {'Neighbourhood':'Earls Court', 'rating':9},
            {'Neighbourhood':'Brixton', 'rating':1},
            {'Neighbourhood':'Marylebone', 'rating':10}]

neighbourhood_ratings = pd.DataFrame(userInput)
```

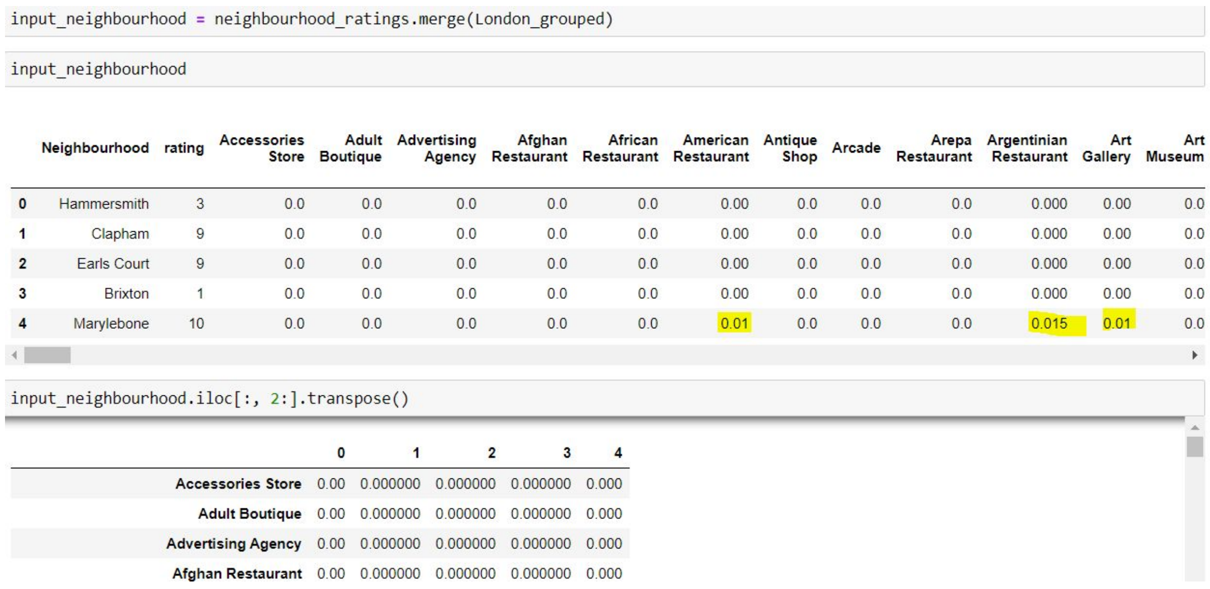


```
neighbourhood_ratings
```



	Neighbourhood	rating
0	Hammersmith	3
1	Clapham	9
2	Earls Court	9
3	Brixton	1
4	Marylebone	10

We then use these neighbourhood ratings to create a user profile of his preference. This is done by multiplying the ratings by the weightings of each venue category in those neighbourhoods. As an example here we see that the user rates Marylebone as 10. We also see that in Marylebone “American Restaurant” and “Art Gallery” have a relative weight of 0.01 while “Argentinian Restaurant” have a higher relative weight of 0.015.



```
input_neighbourhood = neighbourhood_ratings.merge(London_grouped)
```

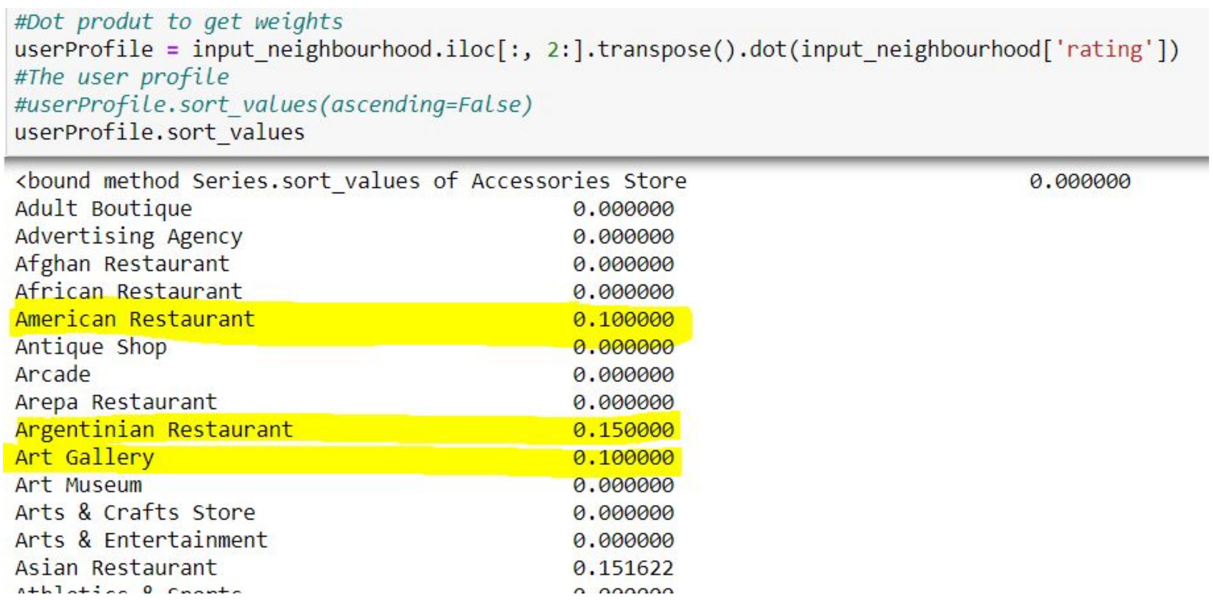
```
input_neighbourhood
```

	Neighbourhood	rating	Accessories Store	Adult Boutique	Advertising Agency	Afghan Restaurant	African Restaurant	American Restaurant	Antique Shop	Arcade	Arepa Restaurant	Argentinian Restaurant	Art Gallery	Art Museum
0	Hammersmith	3	0.0	0.0	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.000	0.00	0.0
1	Clapham	9	0.0	0.0	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.000	0.00	0.0
2	Earls Court	9	0.0	0.0	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.000	0.00	0.0
3	Brixton	1	0.0	0.0	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.000	0.00	0.0
4	Marylebone	10	0.0	0.0	0.0	0.0	0.0	0.01	0.0	0.0	0.0	0.015	0.01	0.0

```
input_neighbourhood.iloc[:, 2:].transpose()
```

	0	1	2	3	4
Accessories Store	0.00	0.000000	0.000000	0.000000	0.000
Adult Boutique	0.00	0.000000	0.000000	0.000000	0.000
Advertising Agency	0.00	0.000000	0.000000	0.000000	0.000
Afghan Restaurant	0.00	0.000000	0.000000	0.000000	0.000

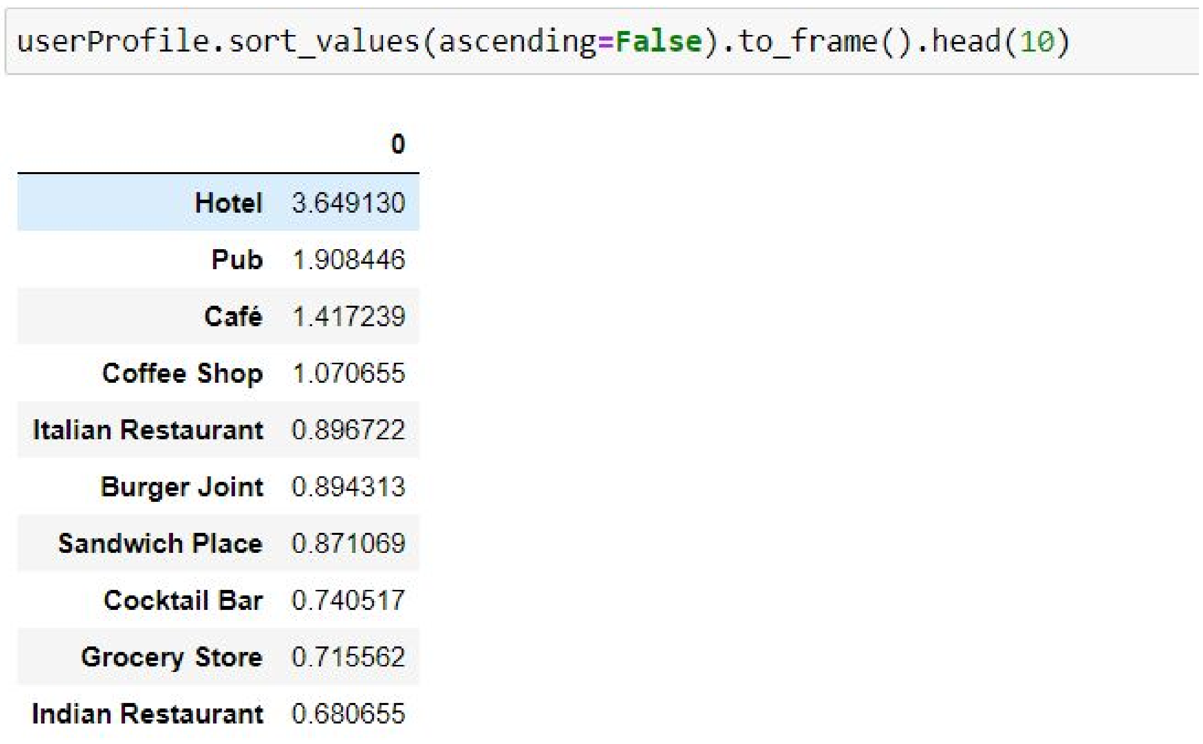
Therefore the final user profile preferences will give weightings of 0.1 to “American Restaurant” and “Art Gallery” (i.e. user rating for Marylebone 10*0.01 relative weights of venues in Marylebone) and 0.15 to “Argentinean Restaurant” (user rating for Marylebone 10*.015 relative weight of venue).



```
#Dot produt to get weights
userProfile = input_neighbourhood.iloc[:, 2:].transpose().dot(input_neighbourhood['rating'])
#The user profile
#userProfile.sort_values(ascending=False)
userProfile.sort_values
```

```
<bound method Series.sort_values of Accessories Store                                    0.000000
Adult Boutique                                  0.000000
Advertising Agency                              0.000000
Afghan Restaurant                               0.000000
African Restaurant                              0.000000
American Restaurant                             0.100000
Antique Shop                                    0.000000
Arcade                                          0.000000
Arepa Restaurant                                0.000000
Argentinian Restaurant                          0.150000
Art Gallery                                     0.100000
Art Museum                                      0.000000
Arts & Crafts Store                             0.000000
Arts & Entertainment                            0.000000
Asian Restaurant                                0.151622
Athletics & Sports                              0.000000
```

If we order the user preferences then we can see that the user likes neighbourhoods with lots of places to eat and a lot of hotels (hotels might be a proxy for upscale and desirable areas where tourists stay).



```
userProfile.sort_values(ascending=False).to_frame().head(10)
```

	0
Hotel	3.649130
Pub	1.908446
Café	1.417239
Coffee Shop	1.070655
Italian Restaurant	0.896722
Burger Joint	0.894313
Sandwich Place	0.871069
Cocktail Bar	0.740517
Grocery Store	0.715562
Indian Restaurant	0.680655

We then normalise the user profile preferences and multiply by the neighbourhood venue weightings of each neighbourhood. This creates a recommendation table.

```
normalised_userProfile = userProfile/userProfile.sum()
```

```
normalised_userProfile.shape
London_grouped.iloc[:,1:].shape
```

```
(170, 402)
```

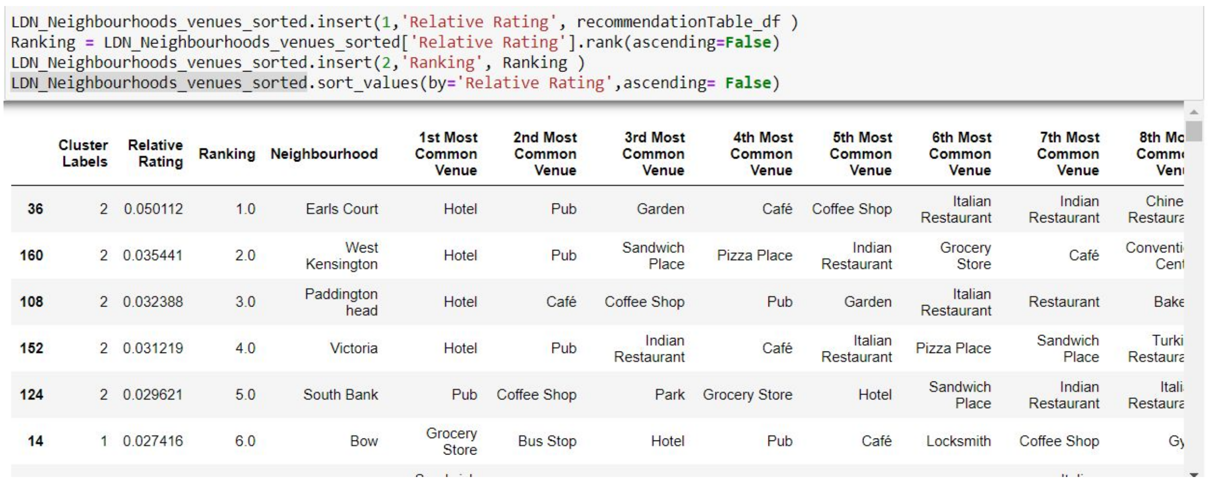
```
#Multiply the characteristics by the weights and then take the weighted average
recommendationTable_df = (London_grouped.iloc[:,1:]*normalised_userProfile).sum(axis=1)
recommendationTable_df.head()
```

```
0    0.008455
1    0.020917
2    0.023615
3    0.020321
4    0.020458
dtype: float64
```

This recommendation table has a relative rating value. Higher values of this relative rating indicate neighbourhoods that strongly match the preferences of the user. The rating is not in the same scale as the original rating given by the user so we rank them all in descending order and create a new column called ranking. The neighbourhood indexed by 0 (i.e. Abbey Wood) has a relative rating of 0.008455 whereas the neighbourhood indexed by 1 (i.e. Acton) has a relative rating of 0.020917...etc. The new column ranking will have a value of 1 for the highest relative weight, 2 for the second highest relative weight. We can explore the results in the next section.

Results

Joining the results table to the existing neighbourhoods table we observed that the neighbourhoods with the highest ranking (i.e. those neighbourhoods whose venues most closely match the user preferences) all have as the most common venue hotels. We said before that hotels might be a proxy for desirable areas favoured by tourists.



```
LDN_Neighbourhoods_venues_sorted.insert(1,'Relative Rating', recommendationTable_df )
Ranking = LDN_Neighbourhoods_venues_sorted['Relative Rating'].rank(ascending=False)
LDN_Neighbourhoods_venues_sorted.insert(2,'Ranking', Ranking )
LDN_Neighbourhoods_venues_sorted.sort_values(by='Relative Rating',ascending= False)
```

	Cluster Labels	Relative Rating	Ranking	Neighbourhood	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 Mo Comm Ven
36	2	0.050112	1.0	Earls Court	Hotel	Pub	Garden	Café	Coffee Shop	Italian Restaurant	Indian Restaurant	Chine Restaura
160	2	0.035441	2.0	West Kensington	Hotel	Pub	Sandwich Place	Pizza Place	Indian Restaurant	Grocery Store	Café	Conventi Cent
108	2	0.032388	3.0	Paddington head	Hotel	Café	Coffee Shop	Pub	Garden	Italian Restaurant	Restaurant	Bake
152	2	0.031219	4.0	Victoria	Hotel	Pub	Indian Restaurant	Café	Italian Restaurant	Pizza Place	Sandwich Place	Turki Restaura
124	2	0.029621	5.0	South Bank	Pub	Coffee Shop	Park	Grocery Store	Hotel	Sandwich Place	Indian Restaurant	Itali Restaura
14	1	0.027416	6.0	Bow	Grocery Store	Bus Stop	Hotel	Pub	Café	Locksmith	Coffee Shop	Gy

We can also see that the top 5 matches are in cluster 2, which confirms that those neighbourhoods are similar. We can see that the user prefers neighbourhoods with lots of hotels, pubs, cafés/coffee shops and different food places - all of these venues figuring among the most common in the neighbourhoods.

Discussion

The above recommender system for neighbourhood recommends Earls Court as the best neighbourhood and this was rated very highly by the user - however it was not rated the highest. The user does indeed prefer neighbourhoods with a high prevalence of hotels (if we assume hotels are a proxy for desirable neighbourhoods). The limitation of this approach is that the user might not necessarily be interested in neighbourhoods with lots of hotels per se, but in neighbourhoods with a different feature which is not captured by the data. Perhaps hotels generally are located in safe areas with low crime rates. Or perhaps hotels are generally located in very picturesque areas with lots of history and landmarks. However our model does not include either of these two measures so we cannot say for sure that the user prefers neighbourhoods with a high prevalence of hotels.

Indeed the recommendation only takes into account the venues/features present in the original rated neighbourhoods. If those originally rated neighbourhoods do not have lots of parks or green spaces then it will never suggest neighbourhoods with lots of parks or green spaces - not because the user does not like these features but because the user has not rated neighbourhoods with these features. So it will only suggest neighbourhoods that are very similar to what he has already tried. A better recommender system could give some weighting to the features (age, gender, education, income ...etc) of the user and suggest recommendations based on similar users.

We can also observe that the top 5 neighbourhoods recommended all belong to cluster 2. This tells us that those neighbourhoods are similar to each other - and match the preferences of our user.

Conclusion

We have collected data from different sources, we cleaned the data, and we manipulated the data such that we were able to divide London into different neighbourhoods (based on postcodes). Using the postcodes and coordinates data we then used the FOURSQUARES API to get the different types of venues in each neighbourhood.

We did exploratory data analysis including clustering of neighbourhoods using k-means clustering. We then developed a recommender system. This system used the user ratings as input to figure out a user preference profile. This user preference for distinct types of venues were then matched with the neighbourhoods containing those venues - and a recommendation made to the user.

We concluded that the recommendations made to the user matched his preferences but noted that the recommender system was limited to venues/features contained in the neighbourhoods he rated. The recommender system could be improved by using data from the user profile and suggesting neighbourhoods liked by similar users. We also noted that most of the recommended neighbourhoods belonged to the same cluster.

References

[1] Wikipedia contributors. (2021, February 26). London postal district. In *Wikipedia, The Free Encyclopedia*. Retrieved 17:21, March 2, 2021, from https://en.wikipedia.org/w/index.php?title=London_postal_district&oldid=1009140985

[2] Wikipedia contributors. (2021, March 1). E postcode area. In *Wikipedia, The Free Encyclopedia*. Retrieved 17:19, March 2, 2021, from https://en.wikipedia.org/w/index.php?title=E_postcode_area&oldid=1009531161

[3] Wikipedia contributors. (2020, December 18). EC postcode area. In *Wikipedia, The Free Encyclopedia*. Retrieved 17:19, March 2, 2021, from https://en.wikipedia.org/w/index.php?title=EC_postcode_area&oldid=994966749

[4] Wikipedia contributors. (2021, January 21). N postcode area. In *Wikipedia, The Free Encyclopedia*. Retrieved 17:19, March 2, 2021, from https://en.wikipedia.org/w/index.php?title=N_postcode_area&oldid=1001787686

[5] Wikipedia contributors. (2021, January 11). NW postcode area. In *Wikipedia, The Free Encyclopedia*. Retrieved 17:19, March 2, 2021, from https://en.wikipedia.org/w/index.php?title=NW_postcode_area&oldid=999732776

[6] Wikipedia contributors. (2021, February 26). SE postcode area. In *Wikipedia, The Free Encyclopedia*. Retrieved 17:19, March 2, 2021, from https://en.wikipedia.org/w/index.php?title=SE_postcode_area&oldid=1009068149

[7] Wikipedia contributors. (2021, February 28). SW postcode area. In *Wikipedia, The Free Encyclopedia*. Retrieved 17:19, March 2, 2021, from https://en.wikipedia.org/w/index.php?title=SW_postcode_area&oldid=1009466671

[8] Wikipedia contributors. (2021, February 11). W postcode area. In *Wikipedia, The Free Encyclopedia*. Retrieved 17:19, March 2, 2021, from https://en.wikipedia.org/w/index.php?title=W_postcode_area&oldid=1006095827

[9] Wikipedia contributors. (2021, January 7). WC postcode area. In *Wikipedia, The Free Encyclopedia*. Retrieved 17:19, March 2, 2021, from https://en.wikipedia.org/w/index.php?title=WC_postcode_area&oldid=998980977

[10] Freemaptools.com. (2021). Download UK Postcodes with Latitude and Longitude. Retrieved 17:19, March 2, 2021, from https://www.freemaptools.com/download-uk-postcode-lat-lng.htm

- Contains Ordnance Survey data © Crown copyright and database right 2020
- Contains Royal Mail data © Royal Mail copyright and database right 2020
- Source: Office for National Statistics licensed under the Open Government Licence v.3.0

[11] FOURSQUARE API Attribution | Usage Guidelines (foursquare.com)