Exploring Areas of London

Introduction

London is one of the largest and most important cities of Europe with a diverse population of close to 9 million people. It makes a considerable impact upon the arts, commerce, education, entertainment, fashion, finance, healthcare, media, professional services, research and development, tourism and transportation.

In this project we aim to understand the structure of London better: is there a clear distinction between some areas and what is it? For that we will analyse different locations around the city and venues close to them. This should tell us how people use them.

Aside from pure academic interest, our research can help the government of London in city development planning.

Data

We used free and open sources of data for the project, listed below.

Wikipedia

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

List of London locations and boroughs is provided in HTML format:

Location	London borough	Post town	Postcode district	Dial code	OS grid ref
Abbey Wood	Bexley, Greenwich [7]	LONDON	SE2	020	TQ465785
Acton	Ealing, Hammersmith and Fulham[8]	LONDON	W3, W4	020	TQ205805
Addington	Croydon[8]	CROYDON	CR0	020	TQ375645
Addiscombe	Croydon[8]	CROYDON	CR0	020	TQ345665
Albany Park	Bexley	BEXLEY, SIDCUP	DA5, DA14	020	TQ478728
Aldborough Hatch	Redbridge[9]	ILFORD	IG2	020	TQ455895
Aldgate	City[10]	LONDON	EC3	020	TQ334813
Aldwych	Westminster[10]	LONDON	WC2	020	TQ307810
Alperton	Brent[11]	WEMBLEY	HA0	020	TQ185835

Geocoder

Link: https://geocoder.readthedocs.io/

This is a geocoding library supporting many providers. Free Arcgis provider was used.

```
>>> import geocoder
>>> g = geocoder.google('Mountain View, CA')
>>> g.latlng
(37.3860517, -122.0838511)
```

Foursquare

Link: https://developer.foursquare.com/docs/places-api/

Foursquare provides an HTTP REST API for access to a list of venues at any coordinates. That API is free, but requires a registration to obtain an API key.

An example of data:


```
{'meta': {'code': 200, 'requestId': '5f0265e7efbbe10132e2f2d5'},
 'response': {'warning': {'text': "There aren't a lot of results near you. Try something more general, reset your fil
ters, or expand the search area."},
  'headerLocation': 'Bygrave',
  'headerFullLocation': 'Bygrave',
  'headerLocationGranularity': 'city',
  'totalResults': 1,
  'suggestedBounds': {'ne': {'lat': 52.00900000900001,
    'lng': -0.1354088564683746},
   'sw': {'lat': 51.99099999099999, 'lng': -0.16459114353162538}},
  'groups': [{'type': 'Recommended Places',
    'name': 'recommended',
    'items': [{'reasons': {'count': 0,
       'items': [{'summary': 'This spot is popular',
         'type': 'general',
         'reasonName': 'globalInteractionReason'}]},
      'venue': {'id': '514f51d67ab4081c42d8050a',
       'name': 'SG7.biz',
       'location': {'address': 'Royston Road',
        'lat': 51.997292929613515,
        'lng': -0.15892269822019148,
        'labeledLatLngs': [{'label': 'display',
          'lat': 51.997292929613515,
          'lng': -0.15892269822019148}],
        'distance': 681,
        'postalCode': 'SG7 6QY',
        'cc': 'GB',
        'city': 'Baldock',
        'state': 'Hertfordshire',
        'country': 'United Kingdom',
        'formattedAddress': ['Royston Road',
         'Baldock',
         'Hertfordshire',
         'SG7 6QY',
         'United Kingdom']},
       'categories': [{'id': '4bf58dd8d48988d1f4941735',
```


Methodology

Prepare London locations dataset

Downloading

First we download a list of all locations from Wikipedia with Pandas **read_html** method:

	Location	Borough	Town	Postcode
0	Abbey Wood	Bexley, Greenwich [7]	LONDON	SE2
1	Acton	Ealing, Hammersmith and Fulham[8]	LONDON	W3, W4
2	Addington	Croydon[8]	CROYDON	CR0
3	Addiscombe	Croydon[8]	CROYDON	CR0
4	Albany Park	Bexley	BEXLEY, SIDCUP	DA5, DA14
...	...	...	...	...

There are **533** locations.

Cleaning

Borough, Town and Postcode columns have multiple comma-separated values in many cases. We only want to take a single one for geocoding purposes. There are also numeric references present in some boroughs which we want to remove.

Geocoding

We don't have any coordinates in the dataset from Wiki. To get them we will use Geocoder python library with Arcgis provider which doesn't require any API key.

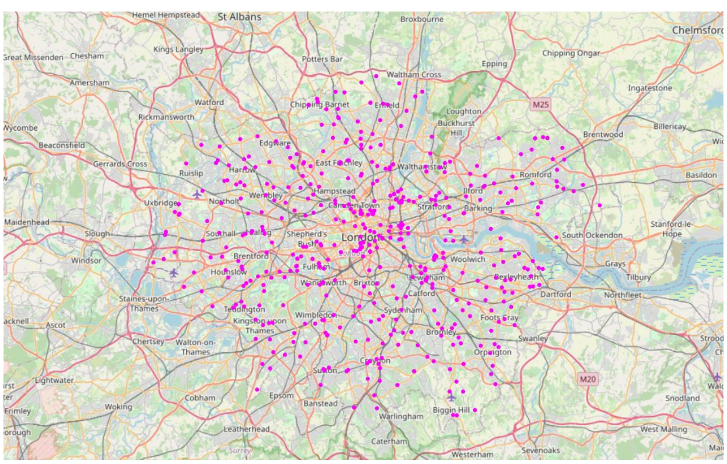
After this step we have a clean dataset with coordinates:

	Location	Borough	Town	Postcode	Lat	Lng
0	Abbey Wood	Bexley	LONDON	SE2	51.4925	0.12127
1	Acton	Ealing	LONDON	W3	51.5181	-0.301954
2	Addington	Croydon	CROYDON	CR0	51.3588	-0.0329062
3	Addiscombe	Croydon	CROYDON	CR0	51.3736	-0.0903331
4	Albany Park	Bexley	BEXLEY	DA5	51.4357	0.12588
...	...	...	...	...	...	...
528	Woolwich	Greenwich	LONDON	SE18	51.4855	0.00627
529	Worcester Park	Sutton	WORCESTER PARK	KT4	51.3751	-0.23489
530	Wormwood Scrubs	Hammersmith and Fulham	LONDON	W12	51.4777	-0.20145
531	Yeading	Hillingdon	HAYES	UB4	51.5244	-0.399389
532	Yiewsley	Hillingdon	WEST DRAYTON	UB7	51.5126	-0.47259

533 rows × 6 columns

Visualizing

Plotting them on the map we can see we covered the city pretty well:



Get venues data

Foursquare has venues organized into lots of hierarchical categories, with top level:

1. Arts & Entertainment

2. College & University
3. Event
4. Food
5. Nightlife Spot
6. Outdoors & Recreation
7. Professional & Other Places
8. Residence'
9. Shop & Service
10. Travel & Transport

Every category has a known ID, which can be obtained in the website or by listing the categories via REST API: https://developer.foursquare.com/docs/api-reference/venues/categories/.

The venues can be queried with another REST API:
https://developer.foursquare.com/docs/api-reference/venues/explore/
That API has the following useful parameters, among many:

- *ll* - coordinates
- *categoryId* - filter by category
- *radius* - radius in meters to look for venues around the coordinates

In our project we won't actually use venues data itself, but rather ***totalResults*** response field. We will make a query for **every location** for **every category** and get a total count of venues in that category at that location. Overall we will make 533*10 requests, so that will take a while. But in the end we are going to have this dataset:

	Location	Arts & Entertainment	College & University	Event	Food	Nightlife Spot	Outdoors & Recreation	Professional & Other Places	Residence	Shop & Service	Travel & Transport
0	Abbey Wood	1	0	0	6	6	5	3	2	7	4
1	Acton	4	5	0	61	14	12	38	6	66	30
2	Addington	0	0	0	6	1	5	4	0	3	3
3	Addiscombe	5	7	0	53	25	8	36	6	82	32
4	Albany Park	1	2	0	5	5	2	6	0	17	2
...	...	...	...	...	...	...	...	...	...	...	...
528	Woolwich	6	11	0	31	11	13	23	8	16	14
529	Worcester Park	2	0	0	12	4	2	7	1	19	4
530	Wormwood Scrubs	9	4	0	113	40	33	40	10	60	19
531	Yeading	0	2	0	5	3	2	5	0	16	3
532	Yiewsley	1	1	0	5	5	4	7	1	12	11

533 rows × 11 columns

Pickle

Both geocoding and querying Foursquare takes a long time, besides there is a limit on the number of calls per day. To avoid repeating these queries all over again when re-running the notebook we serialize results in Python binary format: **Pickle**.

Data Analysis

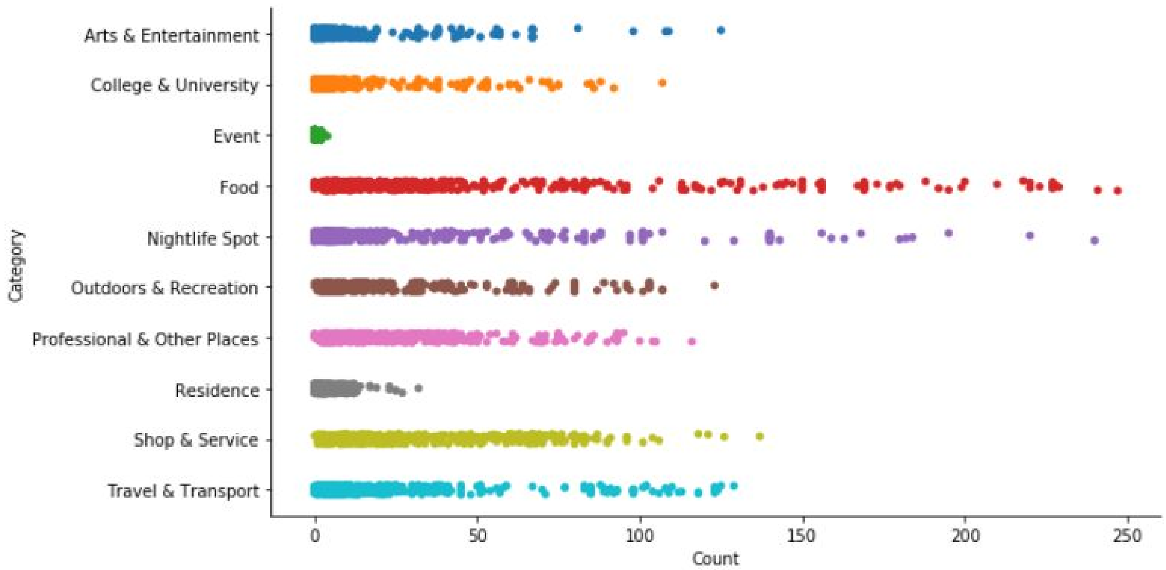
Analyse venues

Let's take a look at most popular venues in London, by their max presence in any location by using Pandas **describe** method:

	count	mean	std	min	25%	50%	75%	max
Food	533.0	44.917448	55.979829	0.0	7.0	22.0	57.0	247.0
Nightlife Spot	533.0	23.046904	36.504450	0.0	4.0	7.0	22.0	240.0
Shop & Service	533.0	32.204503	26.846427	0.0	10.0	21.0	53.0	137.0
Travel & Transport	533.0	21.090056	28.533573	0.0	4.0	8.0	23.0	129.0
Arts & Entertainment	533.0	9.410882	16.950448	0.0	1.0	3.0	8.0	125.0
Outdoors & Recreation	533.0	18.131332	23.700465	0.0	4.0	7.0	21.0	123.0
Professional & Other Places	533.0	23.945591	22.452767	0.0	7.0	16.0	35.0	116.0
College & University	533.0	10.170732	17.141136	0.0	2.0	4.0	7.0	107.0
Residence	533.0	4.170732	4.162007	0.0	1.0	3.0	6.0	32.0
Event	533.0	0.106942	0.431173	0.0	0.0	0.0	0.0	4.0

Or another nice representation with a boxplot graph:

```
places_stacked = places.set_index('Location').stack().reset_index()
places_stacked.columns = ['Location', 'Category', 'Count']
sns.catplot(x="Count", y="Category", orient="h", aspect=2, data=places_stacked);
```



We can see that different categories have different max counts. We, however, want to compare different areas of the city to each other, not categories. That's why we need to normalize the data.

Normalization

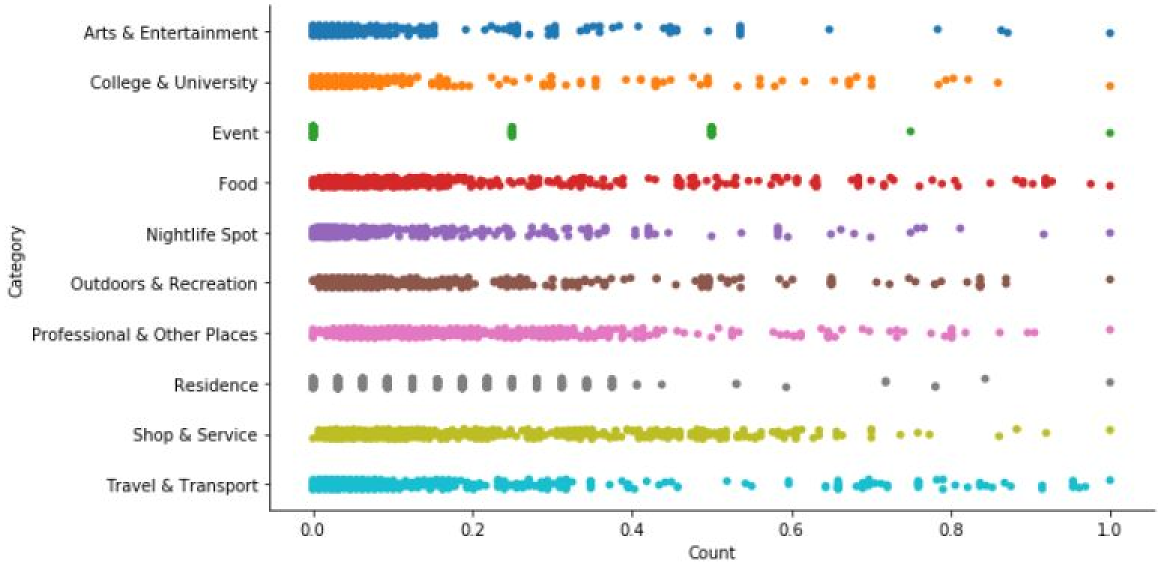
We will do this by dividing each category count by the max number of these categories.

```
places_normalized = places.set_index('Location')
places_normalized = places_normalized.div(places_normalized.max())
places_normalized = places_normalized.reset_index()
places_normalized
```

So we'll have every category count in range 0 to 1:

	Location	Arts & Entertainment	College & University	Event	Food	Nightlife Spot	Outdoors & Recreation	Professional & Other Places	Residence	Shop & Service	Travel & Transport
0	Abbey Wood	0.008	0.000000	0.0	0.024291	0.025000	0.040650	0.025862	0.06250	0.051095	0.031008
1	Acton	0.032	0.046729	0.0	0.246964	0.058333	0.097561	0.327586	0.18750	0.481752	0.232558
2	Addington	0.000	0.000000	0.0	0.024291	0.004167	0.040650	0.034483	0.00000	0.021898	0.023256
3	Addiscombe	0.040	0.065421	0.0	0.214575	0.104167	0.065041	0.310345	0.18750	0.598540	0.248062
4	Albany Park	0.008	0.018692	0.0	0.020243	0.020833	0.016260	0.051724	0.00000	0.124088	0.015504
...	...	...	...	...	...	...	...	...	...	...	...

```
places_stacked = places_normalized.set_index('Location').stack().reset_index()
places_stacked.columns = ['Location', 'Category', 'Count']
sns.catplot(x="Count", y="Category", orient="h", aspect=2, data=places_stacked);
```



That's much better and we are ready for machine learning.

Cleaning

We chose to use a predefined set of categories for our analysis. That saved us from the need to clean category data.

Machine Learning

We will use K-Means - unsupervised learning algorithm that will cluster London locations into several groups.

Optimal K

The most important part is to find the number of clusters we want to use. There can be multiple viable numbers all grouping data points in different ways (or none at all, in which case K-Mean doesn't fit here).

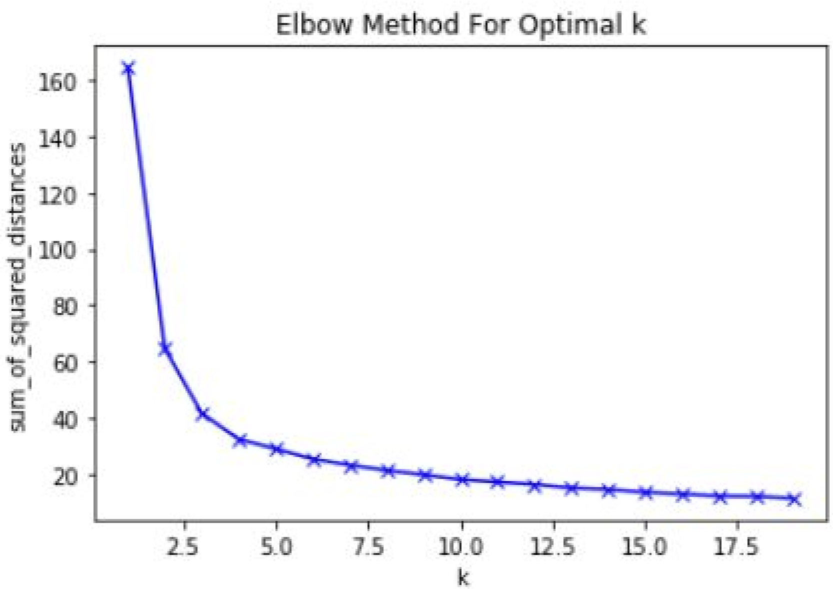
There are two methods we can use to help us: Elbow and Silhouette

The Elbow Method- calculate the sum of squared distances of samples to their closest cluster center for different values of k. The value of k after which there is no significant decrease in sum of squared distances is chosen.

```
sum_of_squared_distances = []
K = range(1,20)
for k in K:
    print(k, end=' ')
    kmeans = KMeans(n_clusters=k).fit(places_clustering)
    sum_of_squared_distances.append(kmeans.inertia_)
```

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
```

```
plt.plot(K, sum_of_squared_distances, 'bx-')
plt.xlabel('k')
plt.ylabel('sum_of_squared_distances')
plt.title('Elbow Method For Optimal k');
```



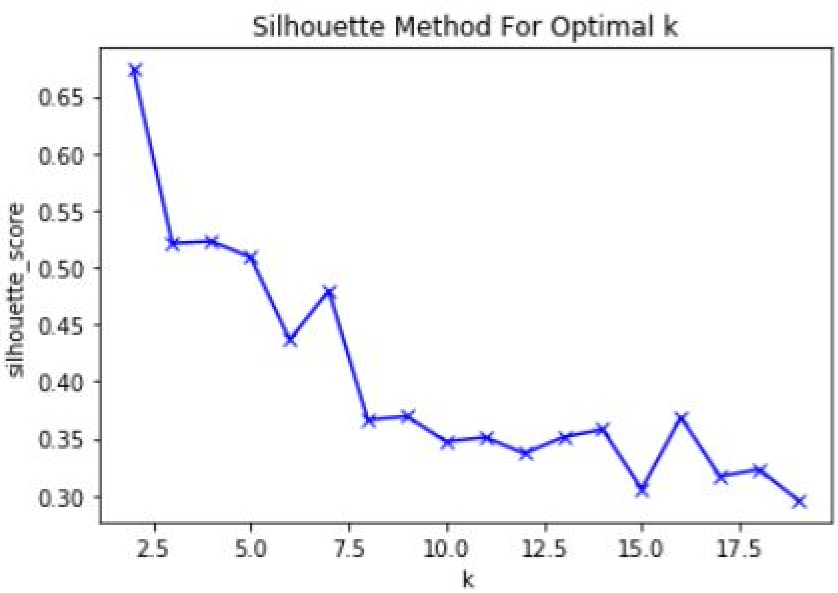
It looks like we have an elbow point at k = 4 or k = 5. But let's see if another method gives a better result.

The Silhouette Method - The silhouette value measures how similar a point is to its own cluster (cohesion) compared to other clusters (separation).

```
sil = []
K_sil = range(2,20)
# minimum 2 clusters required, to define dissimilarity
for k in K_sil:
    print(k, end=' ')
    kmeans = KMeans(n_clusters = k).fit(places_clustering
    labels = kmeans.labels_
    sil.append(silhouette_score(places_clustering, labels
```

```
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
```

```
plt.plot(K_sil, sil, 'bx-')
plt.xlabel('k')
plt.ylabel('silhouette_score')
plt.title('Silhouette Method For Optimal k')
plt.show()
```

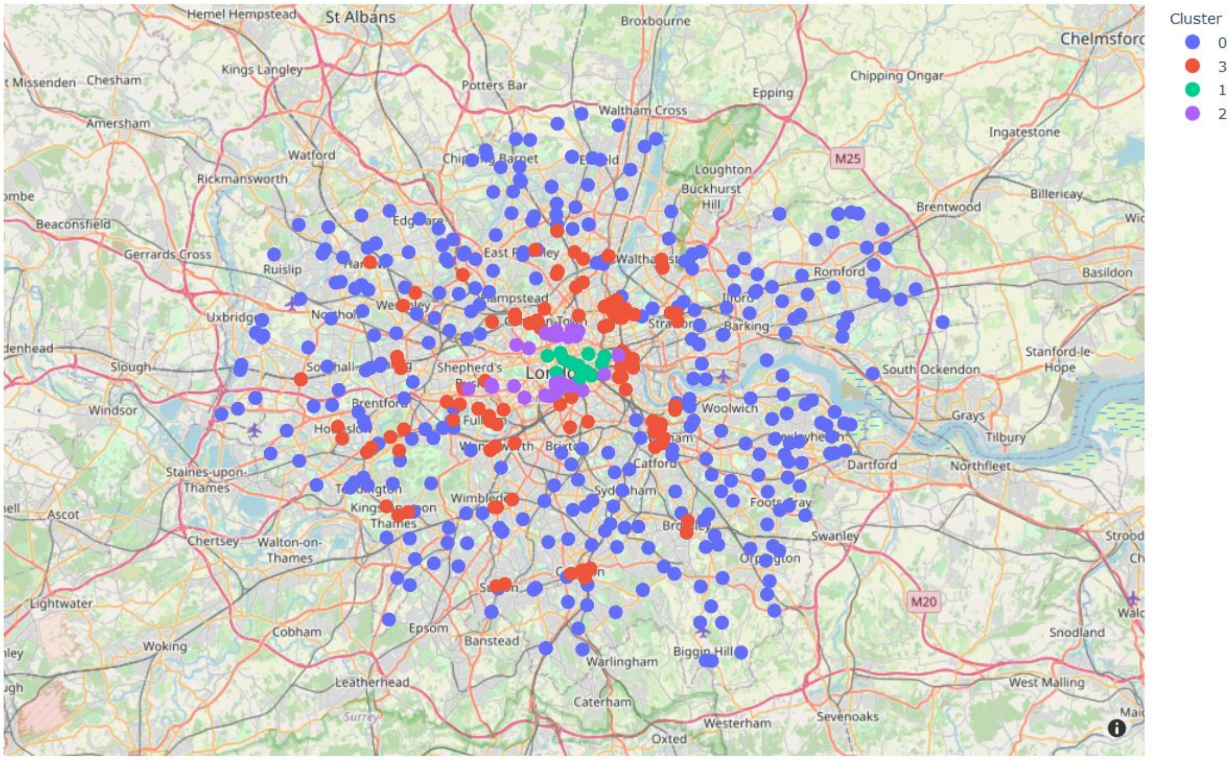


We have local maximums at: 4,7, 9.

Looking at both methods it seems we can try 4 clusters. This is a small enough number so we can analyse every cluster better.

Visualization

And here are our clusters on the map:



Results

Now let’s try to understand the clusters.

Cluster 0

```
for col in required_column:
    print(cluster_0[col].value_counts(ascending = False))
    print("---------------------------------------------")
```

```
Shop & Service                  138
Professional & Other Places      88
Residence                        65
Outdoors & Recreation            19
Travel & Transport               10
Event                             9
College & University              6
Food                              2
Name: 1st Most Common Venue, dtype: int64
---------------------------------------------
Professional & Other Places     115
Shop & Service                   76
Residence                        66
Outdoors & Recreation            31
Travel & Transport               17
Food                             16
College & University             13
Event                             2
Arts & Entertainment              1
Name: 2nd Most Common Venue, dtype: int64
---------------------------------------------
Barnet                     36
Bromley                    35
Bexley                     30
Havering                   21
Croydon                    18
Hillingdon                 17
Brent                      16
Redbridge                  16
Enfield                    16
Harrow                     13
Newham                     13
Hounslow                   12
Richmond upon Thames       12
Kingston upon Thames       11
Greenwich                  10
Lewisham                    8
Barking and Dagenham        8
Ealing                      8
Merton                      7
Haringey                    6
Sutton                      6
Lambeth                     5
Waltham Forest              4
Hackney                     3
Wandsworth                  1
Dartford                    1
Haringey and Barnet         1
Camden                      1
Kensington and Chelsea      1
Southwark                   1
Name: Borough, dtype: int64
---------------------------------------------
```

We can see that this cluster has Shops, Professional places and **Residences**.
Given that no other cluster has that many residences it looks like a distinction feature of that cluster.

Cluster 1

```
for col in required_column:
    print(cluster_1[col].value_counts(ascending = False))
    print("---------------------------------------------")
```

```
Shop & Service                101
Professional & Other Places    21
Food                            7
Travel & Transport              6
Residence                       3
College & University            1
Name: 1st Most Common Venue, dtype: int64
---------------------------------------------
Professional & Other Places    58
Food                           36
Shop & Service                 17
Residence                      13
Outdoors & Recreation           7
Travel & Transport              6
Event                           2
Name: 2nd Most Common Venue, dtype: int64
---------------------------------------------
Tower Hamlets                                   18
Hackney                                         15
Lewisham                                        13
Haringey                                        10
Wandsworth                                       9
Hammersmith and Fulham                           8
Richmond upon Thames                             7
Camden                                           6
Lambeth                                          6
Brent                                            5
Croydon                                          5
Greenwich                                        4
Hounslow                                         4
Waltham Forest                                   4
Kingston upon Thames                             3
Merton                                           3
Kensington and Chelsea                           2
Bromley                                          2
Newham                                           2
Sutton                                           2
Ealing                                           2
Southwark                                        2
Bexley                                           1
Harrow                                           1
Enfield                                          1
Hillingdon                                       1
Kensington and ChelseaHammersmith and Fulham     1
Barnet                                           1
Islington                                        1
Name: Borough, dtype: int64
---------------------------------------------
```

This one has mostly Shops and **Professional** places. We can assume it's a business area of London.

Cluster 2

```
for col in required_column:
    print(cluster_2[col].value_counts(ascending = False))
    print("---------------------------------------------")
```

```
Food                            10
Outdoors & Recreation            3
Shop & Service                   1
Professional & Other Places      1
Travel & Transport               1
College & University             1
Arts & Entertainment             1
Name: 1st Most Common Venue, dtype: int64
---------------------------------------------
Outdoors & Recreation           7
Food                            5
Professional & Other Places     3
Nightlife Spot                  1
Travel & Transport              1
College & University            1
Name: 2nd Most Common Venue, dtype: int64
---------------------------------------------
Southwark        9
City             4
Camden           3
Tower Hamlets    1
Westminster      1
Name: Borough, dtype: int64
---------------------------------------------
```

This one has mostly Food and **Recreation**. Looks like a usual city center for Londoners.

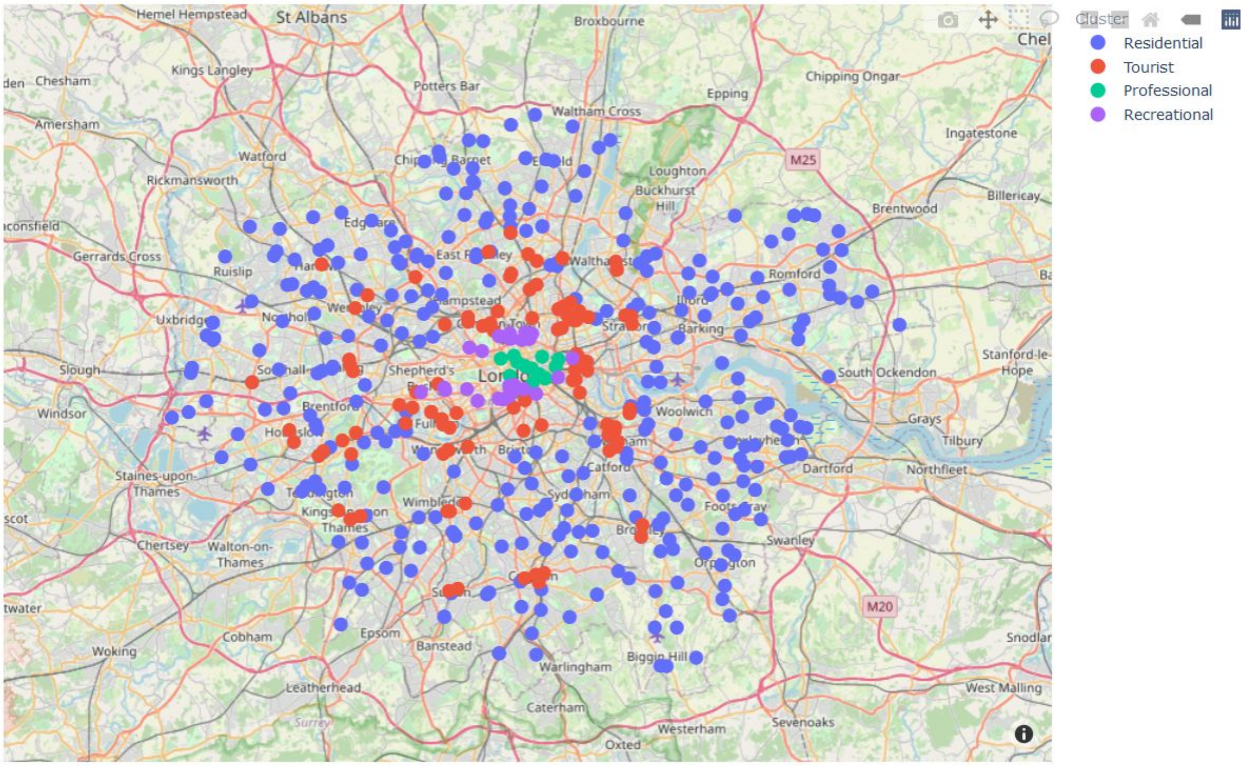
Cluster 3

```
for col in required_column:
    print(cluster_3[col].value_counts(ascending = False))
    print("---------------------------------------------")
```

```
Travel & Transport            16
Food                          16
Professional & Other Places   12
Residence                      3
Shop & Service                 2
Outdoors & Recreation          1
College & University           1
Name: 1st Most Common Venue, dtype: int64
---------------------------------------------
Food                          16
Outdoors & Recreation         12
Shop & Service                 6
Travel & Transport             6
College & University           6
Professional & Other Places    5
Name: 2nd Most Common Venue, dtype: int64
---------------------------------------------
Westminster               18
Islington                 14
Camden                     7
Kensington and Chelsea     5
Tower Hamlets              2
Camden and Islington       1
Islington                  1
Hammersmith and Fulham     1
Lambeth                    1
Southwark                  1
Name: Borough, dtype: int64
---------------------------------------------
```

And this one is Food and Travel. Guess **Tourism**.

Labeled map



6. Discussion

We discovered 4 clusters:

1. Touristic
2. Recreational
3. Professional
4. Residential

We only used 10 basic categories for our analysis. However a careful selection of subcategories should be tried to improve it further.

Also the size of places is not taken into account, because Foursquare doesn't have such a concept. But other metrics like visits count can be used to get more insight.

7. Conclusion

Foursquare data together with unsupervised learning can give useful insights into a city structure. More data from other sources can be used to improve results.