The London NYC: The Comparison of Neighborhoods in London and New York city

Zohre kianni

February 15,2020

1.Introduction

1.1Background

London and New York. The Big Smoke and the Big Apple. Two of the biggest, most vibrant cities in the world. They are the most influential megacities of the anglosphere and of the world. London and New York City are also known as twin cities, therefore these two cities are alike in that they're both densely populated cosmopolitan metropolises, but they also have their differences. Assume that we've been asked to find the better city for launching a new branch of coffee shop or cafe. We have conducted an experiment where we compare the two cities in terms of their public venues, including coffee shops, cafes, restaurants, and etc to get a good measure of each city's residents towards their choices of venue.

1.2Problem

Nowadays there is competition in every sector of the market. This is particularly an issue when it comes to the food industry. Therefore, it is necessary to analyze different factors when opening up a new food catering place, e.g. a cafe or a restaurant. Of crucial importance among these factors is the location. Here we try to classify London and New York City in terms of different location-related factors to find out which one is more desirable to open up a cafe.

1.3Stakeholders

This quantifiable analysis can be used to understand the distribution of different commonplaces or venues within two megacities of the world, New York City and London. Also, it can be utilized by a new food vendor who is willing to open his or her cafe, coffee shop, bar and etc.

2.Data

To examine the above said, the following data sources are used:

2.1London Dataset

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

Neighborhoods were not readily available in .csv format so we scrapped Wikipedia to gather that data. The data was subsequently cleaned. According to the data cleaning, the columns that we didn't need were dropped and the column co-ordinates (51°37′31″N 0°09′06″W / 51.6252°N 0.1517°W) first were separated and then their strings, such as N, W, E, and degree sign were removed. Then we multiplied the data that has W(West) string by minus one because when a longitude is located in the west the result is negative. After exploring the data with foursquare, they were clustered using k-Means algorithms and labeled. So, without further ado, here is the exciting tourist-version competition between these two cities.

2.2New York City Dataset

Link: https://geo.nyu.edu/catalog/nyu_2451_34572

This New York City Neighborhood Names point file was created as a guide to New York City's neighborhoods that appear on the web resource, 'New York: A City of Neighborhoods.' Best estimates of label centroids were established at a 1:1,000 scale, but are ideally viewed at a 1:50,000 scale. This dataset will provide the addresses of neighborhoods of NYC in JSON format

3.Foursquare API

Link: https://developer.foursquare.com/docs

Foursquare API, a location data provider, will be used to make RESTful API calls to retrieve data about venues in different neighborhoods. This is the link to Foursquare Venue Category Hierarchy. Venues retrieved from all the neighborhoods are categorized broadly into 'Arts & Entertainment', 'College & University', 'Event', 'Food', 'Nightlife Spot', 'Outdoors & Recreation', etc.

4.Methodology

4.1Download and Explore London and New York City Dataset

In order to segment the neighborhoods of two Cities, the datasets are required that contain the 5 boroughs and the neighborhoods, that exist in each borough, with respective latitude and longitude coordinates. These datasets are prepared as mentioned before. For the London dataset first,

we scraped its Wikipedia page and then cleaning the dataset and for the NYC dataset, Once the .json file is downloaded, it is analyzed to understand the structure of the file. A python dictionary is returned by the URL and all the relevant data is found to be in the features key, which is basically a list of the neighborhoods. The dictionary is transformed, into a pandas data frame, by looping through the data and filling the data frame rows one at a time.

As a result, data frames are created with Borough, Neighborhood, Latitude and Longitude details of London and NYC's neighborhood.

	Borough	Neighborhood	Latitude	Longitude
0	Bronx	Wakefield	40.894705	-73.847201
1	Bronx	Co-op City	40.874294	-73.829939
2	Bronx	Eastchester	40.887556	-73.827806
3	Bronx	Fieldston	40.895437	-73.905643
4	Bronx	Riverdale	40.890834	-73.912585

NYC data farme

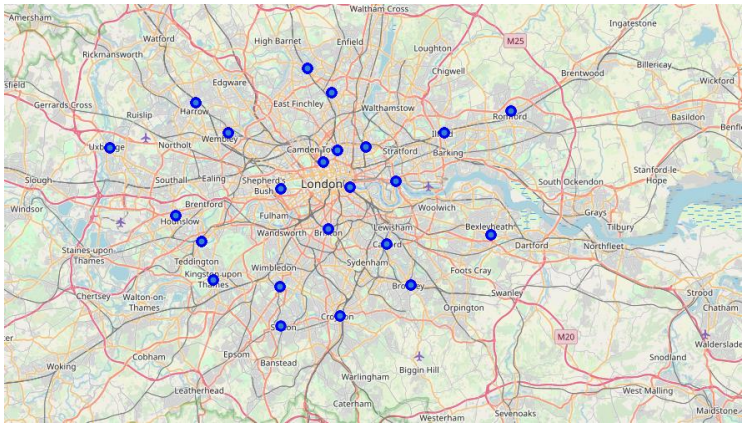
	Borough	latitude	longitude	Neighborhood
0	Barnet	51.6252	-0.1517	Barnet Gate
1	Barnet	51.6252	-0.1517	Brent Cross
2	Barnet	51.6252	-0.1517	Childs Hill
3	Barnet	51.6252	-0.1517	Church End
4	Barnet	51.6252	-0.1517	Colindale

London Dataframe

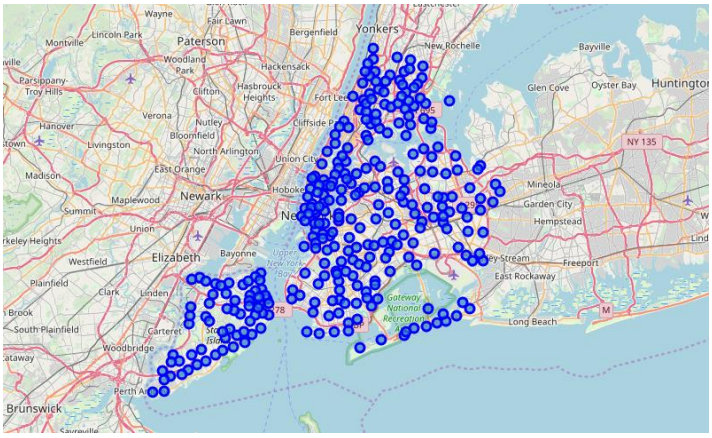
Upon analysis, it is found that the London data frame consists of 23 boroughs and 336 neighborhoods and the NYC data frame consists of 5 boroughs and 306 neighborhoods.

Further, ‘geopy’ library is used to get the latitude and longitude values of two cities, which was returned to be Latitude_NYC: 40.71, Longitude_NYC: -74.01 for NYC and Latitude_London:51.5073219, Longitude_London:-0.1276474 for London.

The curated data frames are then used to visualize by creating a map of London and New York City with their neighborhoods superimposed on top. The following depictions are the maps generated using python ‘folium’ library.



London Map



NYC Map

5.RESTful API Calls to Foursquare

The Foursquare API is used to explore the neighborhoods and segment them. To access the API, ‘CLIENT_ID’, ‘CLIENT_SECRET’ and ‘VERSION’ are defined.

To overcome the redundancy of the process followed above, a function ‘getNearbyVenues’ is created. This functions loop through all the neighborhoods of London and New York City and creates an API request URL with radius = 500, LIMIT = 100. By limit, it is defined that the maximum of 100 nearby venues should be returned. Further, the GET request is made to Foursquare API and only relevant information for each nearby venue is extracted from it. The data is then appended to a python ‘list’. Lastly the python ‘list’ is unfolded or flattened to append it to the data frame being returned by the function.

```
LIMIT=100
def getNearbyVenues(names, latitudes, longitudes, radius=500):

    venues_list=[]
    for name, lat, lng in zip(names, latitudes, longitudes):
        print(name)

        # create the API request URL
        url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={},{}&radius={}&limit={}'.fo
mat( CLIENT_ID,

CLIENT_SECRET,

VERSION,

lat,

lng,

radius,

LIMIT)


        # make the GET request
        results = requests.get(url).json()["response"]['groups'][0]['items']

        # return only relevant information for each nearby venue
        venues_list.append([(
            name,
            lat,
            lng,
            v['venue']['name'],
            v['venue']['location']['lat'],
            v['venue']['location']['lng'],
            v['venue']['categories'][0]['name']) for v in results])

    nearby_venues = pd.DataFrame([item for venue_list in venues_list for item in venue_list])
    nearby_venues.columns = ['Neighbourhood',
                  'Neighbourhood Latitude',
                  'Neighbourhood Longitude',
                  'Venue',
                  'Venue Latitude',
                  'Venue Longitude',
                  'Venue Category']

    return(nearby_venues)
```

```
nyc_venues = getNearbyVenues(names=nyc_neighborhoods['Neighborhood'],
                                    latitudes=nyc_neighborhoods['Latitude'],
                                    longitudes=nyc_neighborhoods['Longitude']
                                   )
```

NYC Venues

```
london_venues = getNearbyVenues(names=final_london_df['Neighborhood'],
                                    latitudes=final_london_df['latitude'],
                                    longitudes=final_london_df['longitude']
                                   )
```

London Venues

Then the size of the resulting two data frames is checked. The returned data frames are as follows:

	Neighbourhood	Neighbourhood Latitude	Neighbourhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Wakefield	40.894705	-73.847201	Lollipops Gelato	40.894123	-73.845892	Dessert Shop
1	Wakefield	40.894705	-73.847201	Rite Aid	40.896649	-73.844846	Pharmacy
2	Wakefield	40.894705	-73.847201	Carvel Ice Cream	40.890487	-73.848568	Ice Cream Shop
3	Wakefield	40.894705	-73.847201	Shell	40.894187	-73.845862	Gas Station
4	Wakefield	40.894705	-73.847201	Dunkin'	40.890459	-73.849089	Donut Shop

NYC_Venues

	Neighbourhood	Neighbourhood Latitude	Neighbourhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Barnet Gate	51.6252	-0.1517	The Atrium	51.624726	-0.151933	Café
1	Barnet Gate	51.6252	-0.1517	Beaconsfield Road (BF)	51.622827	-0.151466	Bus Stop
2	Barnet Gate	51.6252	-0.1517	Oakleigh Cafe	51.623412	-0.154899	Café
3	Brent Cross	51.6252	-0.1517	The Atrium	51.624726	-0.151933	Café
4	Brent Cross	51.6252	-0.1517	Beaconsfield Road (BF)	51.622827	-0.151466	Bus Stop

London_Venues

As of now, two Python ‘data frame’ is created for each city:

1. ‘final_london_df’ and ‘nyc_neighborhoods’ which contain the Borough, Neighborhood, Latitude and Longitude details of the two cities.

2.‘london_venues’ and ‘nyc_venues’ which is a merger between ‘neighborhoods’ data frame and its ‘Venue’ category venues searched with ‘Radius’ = 500 meters and ‘Limit’ = 100. Also, each venue has its own Latitude, Longitude, and Category.

6.Exploratory Data Analysis

The merged data frames ‘london_venues’ and ‘nyc_venues’ have all the required information. The size of these two data frames is determined, and it is found that there is a total of 10280 venues for NYC and 14921 venues For London.

```
print(nyc_venues.shape)
nyc_venues.head()
```

```
(10280, 7)
```

```
print(london_venues.shape)
london_venues.head()
```

```
(14921, 7)
```

7.Feature Engineering

The above process is taken forth by using the ‘one-hot encoding’ function of python ‘pandas’ library. One hot encoding converts the categorical variables (which are ‘Venue Category’) into a form that could be provided to ML algorithms to do a better job in prediction.

```
# one hot encoding
nyc_onehot = pd.get_dummies(nyc_venues[['Venue Category']], prefix="", prefix_sep="")

# add neighborhood column back to dataframe
nyc_onehot['Neighbourhood'] = nyc_venues['Neighbourhood']

# move neighborhood column to the first column
fixed_columns = [nyc_onehot.columns[-1]] + list(nyc_onehot.columns[:-1])
nyc_onehot = nyc_onehot[fixed_columns]

nyc_onehot.head()
```

]:

	Neighbourhood	Accessories Store	Adult Boutique	Afghan Restaurant	African Restaurant	Airport Terminal	American Restaurant	Antique Shop	Arcade	Arepa Restaurant	Argentinian Restaurant	Art Gallery
0	Wakefield	0	0	0	0	0	0	0	0	0	0	0
1	Wakefield	0	0	0	0	0	0	0	0	0	0	0
2	Wakefield	0	0	0	0	0	0	0	0	0	0	0
3	Wakefield	0	0	0	0	0	0	0	0	0	0	0
4	Wakefield	0	0	0	0	0	0	0	0	0	0	0

```
london_onehot = pd.get_dummies(london_venues[['Venue Category']], prefix="", prefix_sep="")

london_onehot['Neighbourhood'] = london_venues['Neighbourhood']

# move neighborhood column to the first column
fixed_columns = [london_onehot.columns[-1]] + list(london_onehot.columns[:-1])
london_onehot = london_onehot[fixed_columns]

london_onehot.head()
```

]:

	Neighbourhood	African Restaurant	American Restaurant	Argentinian Restaurant	Art Gallery	Art Museum	Arts & Crafts Store	Asian Restaurant	BBQ Joint	Bakery	Bar	Bed & Breakfast	Beer Bar	Beer Garden	Beer Store
0	Barnet Gate	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	Barnet Gate	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	Barnet Gate	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	Brent Cross	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	Brent Cross	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Upon converting the categorical variables, as shown above, the ‘Neighborhood’ column is added back. The size of the new data frame ‘nyc_onehot’ is examined and it is found that there are around 10280 data points altogether and for ‘london_onehot’ also there are around 14921, data points altogether. Further, the number of venues in each category in each neighborhood is counted. The top 5 ‘Venue Categories’ can also be found by counting their occurrences.

```
venue_nyc_counts =nyc_onehot.groupby('Neighbourhood').sum()
venue_nyc_counts.head()
```

```
venue_nyc_counts_described=venue_nyc_counts.describe().transpose()
```

```
venue_nyc_top10=venue_nyc_counts_described.sort_values('max', ascending=False)[0:10]
venue_nyc_top10
```

	count	mean	std	min	25%	50%	75%	max
Korean Restaurant	302.0	0.231788	1.628009	0.0	0.0	0.0	0.0	23.0
Italian Restaurant	302.0	1.006623	1.976596	0.0	0.0	0.0	1.0	18.0
Clothing Store	302.0	0.261589	0.882338	0.0	0.0	0.0	0.0	9.0
Pizza Place	302.0	1.473510	1.573544	0.0	0.0	1.0	2.0	9.0
Chinese Restaurant	302.0	0.754967	1.075038	0.0	0.0	0.0	1.0	9.0
Latin American Restaurant	302.0	0.271523	0.789750	0.0	0.0	0.0	0.0	9.0
Bar	302.0	0.741722	1.524827	0.0	0.0	0.0	1.0	9.0
Coffee Shop	302.0	0.980132	1.666437	0.0	0.0	0.0	1.0	9.0
Theater	302.0	0.132450	0.688544	0.0	0.0	0.0	0.0	8.0
Deli / Bodega	302.0	0.844371	1.108238	0.0	0.0	1.0	1.0	8.0

NYC

```
venue_london_counts =london_onehot.groupby('Neighbourhood').sum()
venue_london_counts_described=venue_london_counts.describe().transpose()
venue_london_top10=venue_london_counts_described.sort_values('max', ascending=False)[0:10]
venue_london_top10
```

]:

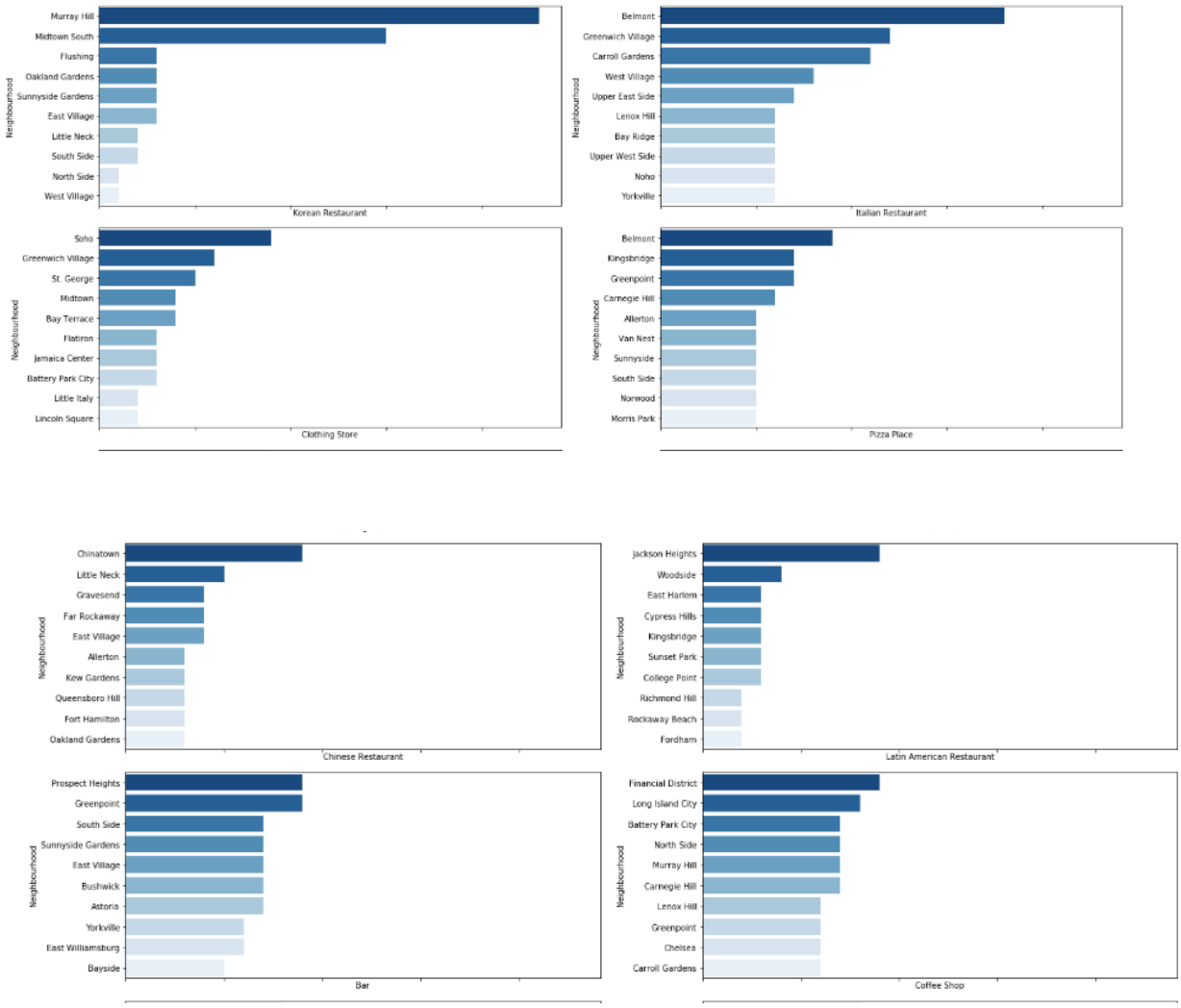
	count	mean	std	min	25%	50%	75%	max
Coffee Shop	332.0	3.768072	3.001078	0.0	2.0	3.0	6.0	13.0
Clothing Store	332.0	2.135542	2.438922	0.0	0.0	1.0	5.0	11.0
Pub	332.0	2.689759	2.854500	0.0	0.0	1.0	5.0	9.0
Café	332.0	2.009036	1.869595	0.0	1.0	1.0	3.0	9.0
Hotel	332.0	1.460843	2.065755	0.0	0.0	1.0	2.0	8.0
Caribbean Restaurant	332.0	0.222892	0.994708	0.0	0.0	0.0	0.0	6.0
Italian Restaurant	332.0	1.563253	1.587024	0.0	0.0	1.0	2.0	6.0
Restaurant	332.0	0.626506	1.101615	0.0	0.0	0.0	1.0	5.0
Mediterranean Restaurant	332.0	0.325301	0.907950	0.0	0.0	0.0	0.0	5.0
Bar	332.0	0.990964	1.322559	0.0	0.0	1.0	1.0	5.0

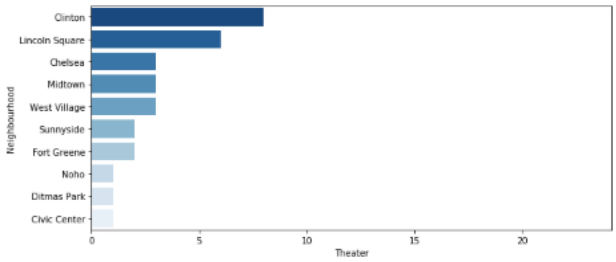
London

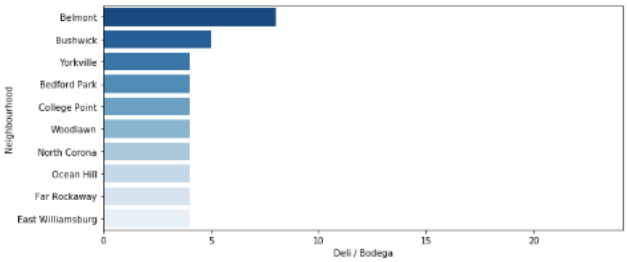
8.Data Visualization

These top 10 categories are further plotted individually on bar graph using python ‘seaborn’ library. A code block is created which loops and plots the graph of top 10 neighborhoods for a category.

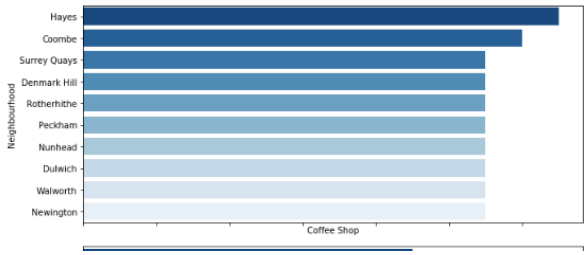
NYC graph:

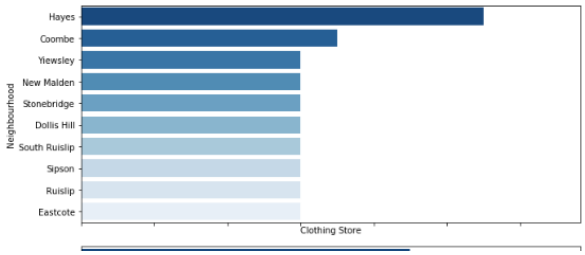


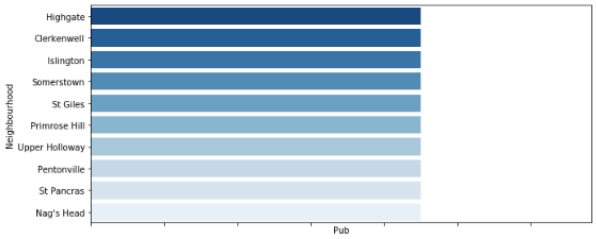


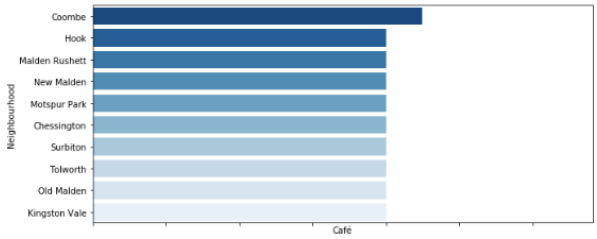


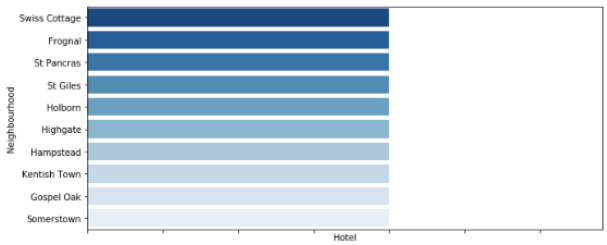
London graph:

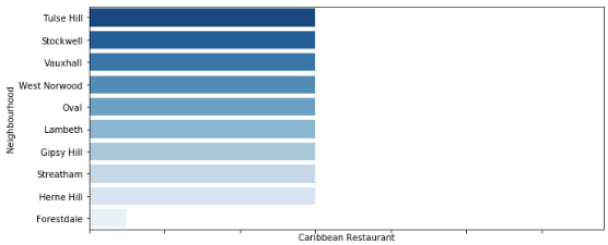


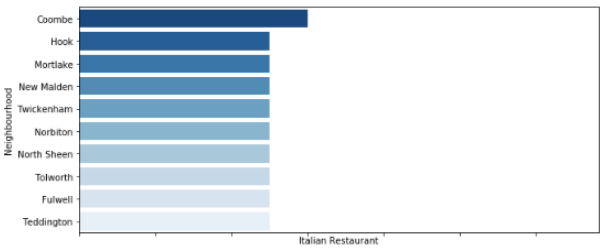


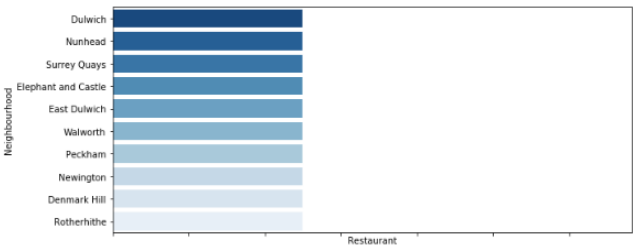


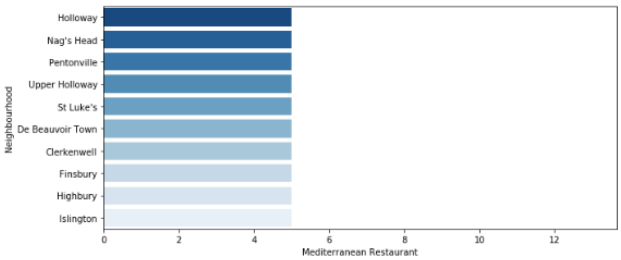


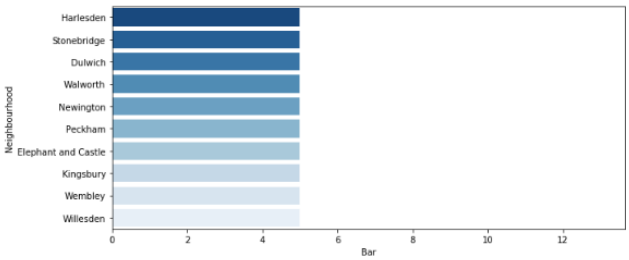












Next, the rows of the neighborhoods are grouped together and the frequency of occurrence of each category is calculated by taking the mean.

As the limit is set to be 100, there will be many venues being returned by the Foursquare API. The created data frame is then fed with the top 5 most common venues categories in the respective neighborhoods.

```
num_top_venues = 5

indicators = ['st', 'nd', 'rd']

# create columns according to number of top venues
columns = ['Neighbourhood']
for ind in np.arange(num_top_venues):
    try:
        columns.append('{}{} Most Common Venue'.format(ind+1, indicators[ind]))
    except:
        columns.append('{}th Most Common Venue'.format(ind+1))

# create a new dataframe
nyc_venues_sorted = pd.DataFrame(columns=columns)
nyc_venues_sorted['Neighbourhood'] = nyc_grouped['Neighbourhood']

for ind in np.arange(nyc_grouped.shape[0]):
    nyc_venues_sorted.iloc[ind, 1:] = return_most_common_venues(nyc_grouped.iloc[ind, :], num_top_venues)

)% _venues_sorted.head()
```

	Neighbourhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue
0	Allerton	Pizza Place	Bakery	Chinese Restaurant	Deli / Bodega	Department Store
1	Annadale	Pizza Place	American Restaurant	Dance Studio	Sports Bar	Train Station
2	Arden Heights	Pizza Place	Deli / Bodega	Pharmacy	Bus Stop	Business Service
3	Arlington	Grocery Store	Deli / Bodega	American Restaurant	Intersection	Bus Stop
4	Arrochar	Italian Restaurant	Bus Stop	Deli / Bodega	Pizza Place	Cosmetics Shop

NYC

```
num_top_venues = 5

indicators = ['st', 'nd', 'rd']

# create columns according to number of top venues
columns = ['Neighbourhood']
for ind in np.arange(num_top_venues):
    try:
        columns.append('{}{} Most Common Venue'.format(ind+1, indicators[ind]))
    except:
        columns.append('{}th Most Common Venue'.format(ind+1))

# create a new dataframe
london_venues_sorted = pd.DataFrame(columns=columns)
london_venues_sorted['Neighbourhood'] = london_grouped['Neighbourhood']

for ind in np.arange(london_grouped.shape[0]):
    london_venues_sorted.iloc[ind, 1:] = return_most_common_venues(london_grouped.iloc[ind, :], num_top_venues)

london_venues_sorted.head()
```

	Neighbourhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue
0	Albany Park	Pub	Clothing Store	Coffee Shop	Fast Food Restaurant	Italian Restaurant
1	Barnet Gate	Café	Bus Stop	Turkish Restaurant	Yoga Studio	Fast Food Restaurant
2	Blendon	Pub	Clothing Store	Coffee Shop	Fast Food Restaurant	Italian Restaurant
3	Brent Cross	Café	Bus Stop	Turkish Restaurant	Yoga Studio	Fast Food Restaurant
4	Brent Park	Coffee Shop	Hotel	Clothing Store	Bar	Sporting Goods Shop

London

9.Machine Learning

'k-means' is an unsupervised machine learning algorithm that creates clusters of data points aggregated together because of certain similarities. This algorithm will be used to count neighborhoods for each cluster label for variable cluster size.

To implement this algorithm, it is very important to determine the optimal number of clusters (i.e. k). There are 2 most popular methods for the same, namely 'The Elbow Method' and 'The Silhouette Method'.

9.1The Elbow Method

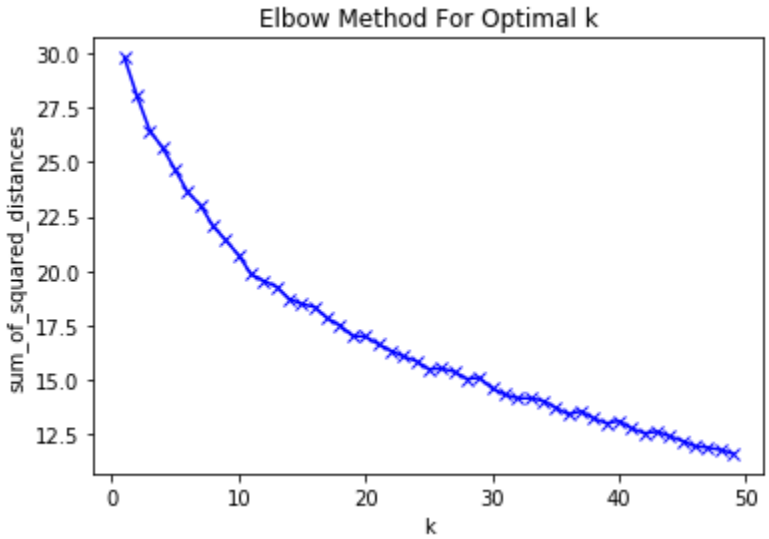
The Elbow Method calculates the sum of squared distances of samples to their closest cluster center for different values of ‘k’. The optimal number of clusters is the value after which there is no significant decrease in the sum of squared distances.

Following is an implementation of this method (with varying number of clusters from 1 to 49):

```
sum_of_squared_distances = []
K = range(1,50)
for k in K:
    print(k, end=' ')
    kmeans = KMeans(n_clusters=k).fit(nyc_grouped_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 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49
```

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



Sometimes, the Elbow method does not give the required result, which happened in this case. As there is a gradual decrease in the sum of squared distances, the optimal number of clusters can not be determined. To counter this, another method can be implemented, as discussed below. Because of this result for the NYC data frame, we ignored implementing this method for the London data set.

9.2The Silhouette Method

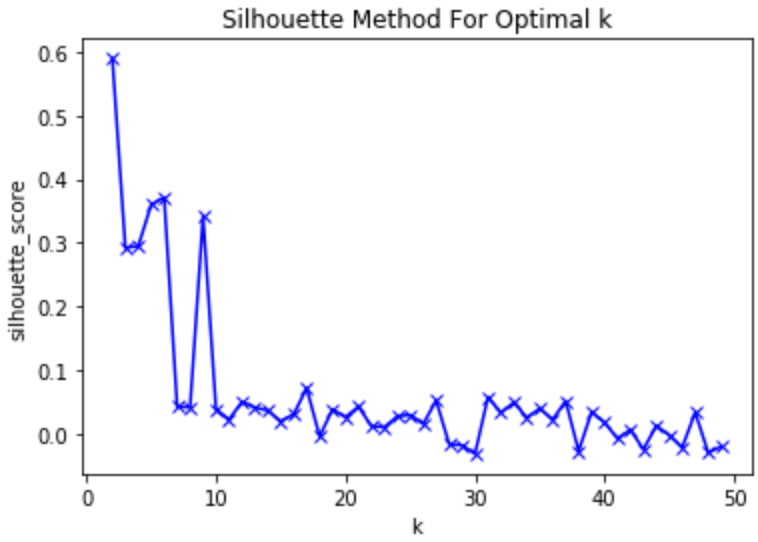
As quoted in Wikipedia — “The Silhouette Method measures how similar a point is to its own cluster (cohesion) compared to other clusters (separation).”

The following is an implementation of this method will vary from 2 to 49:

```
from sklearn.metrics import silhouette_score

sil = []
K_sil = range(2,50)
# minimum 2 clusters required, to define dissimilarity
for k in K_sil:
    print(k, end=' ')
    kmeans = KMeans(n_clusters = k).fit(nyc_grouped_clustering)
    labels = kmeans.labels_
    sil.append(silhouette_score(nyc_grouped_clustering, labels, metric = 'euclidean'))
```

```
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49
```

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



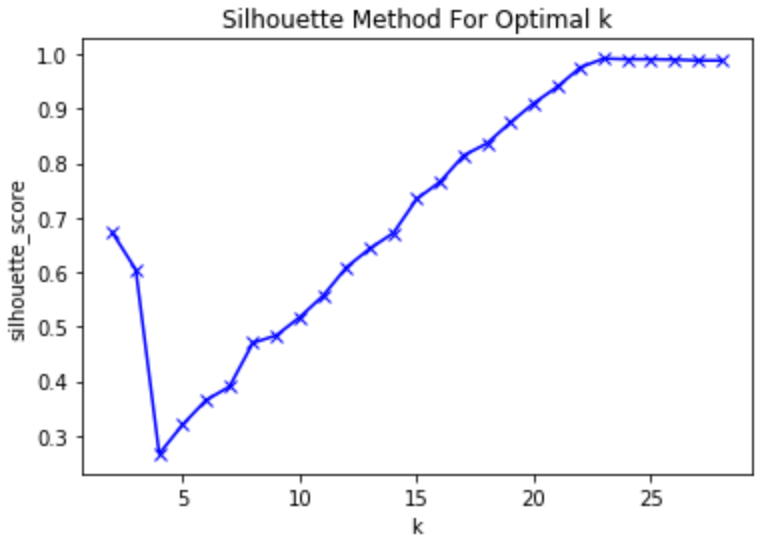
NYC

There is a peak at k = 2, k =5 and k = 6. Two clusters will give a very broad classification of the venues. Therefore, the number of New York clusters (i.e. ‘k’) is chosen to be 5.

The following is an implementation of this method for the London data set that will vary from 2 to 30 because of duplicate points.

```
from sklearn.metrics import silhouette_score

sil = []
K_sil = range(2,29)
# minimum 2 clusters required, to define dissimilarity
for k in K_sil:
    print(k, end=' ')
    kmeans = KMeans(n_clusters = k).fit(london_grouped_clustering)
    labels = kmeans.labels_
    sil.append(silhouette_score(london_grouped_clustering, labels,metric = 'euclidean'))
```

```
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
```



London

There is a peak at k = 2, k = 3 and k = 5. 2 and 3 clusters will give a very broad classification of the venues **the number for London clustering will be 5**.

9.3k-Means

Following code blocks run the k-Means algorithm with the numbers of clusters =5 and print the counts of neighborhoods assigned to different clusters:

```
kclusters =5

nyc_grouped_clustering = nyc_grouped.drop('Neighbourhood', 1)

# run k-means clustering
kmeans = KMeans(n_clusters=kclusters, random_state=0).fit(nyc_grouped_clustering)

# check cluster labels generated for each row in the dataframe
kmeans.labels_[0:10]
```

```
]: array([0, 0, 3, 3, 3, 0, 0, 0, 0, 0], dtype=int32)
```

NYC

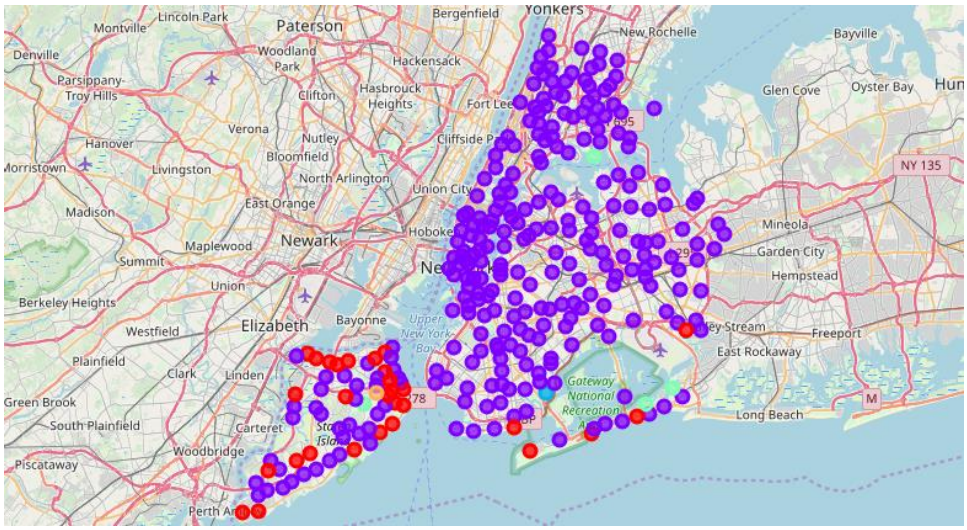
```
kclusters = 5

london_grouped_clustering =london_grouped.drop('Neighbourhood', 1)

# run k-means clustering
kmeans = KMeans(n_clusters=kclusters, random_state=0).fit(london_grouped_clustering)

# check cluster labels generated for each row in the dataframe
kmeans.labels_[0:10]
```

```
]: array([3, 1, 3, 1, 4, 4, 4, 3, 3, 4], dtype=int32)
```

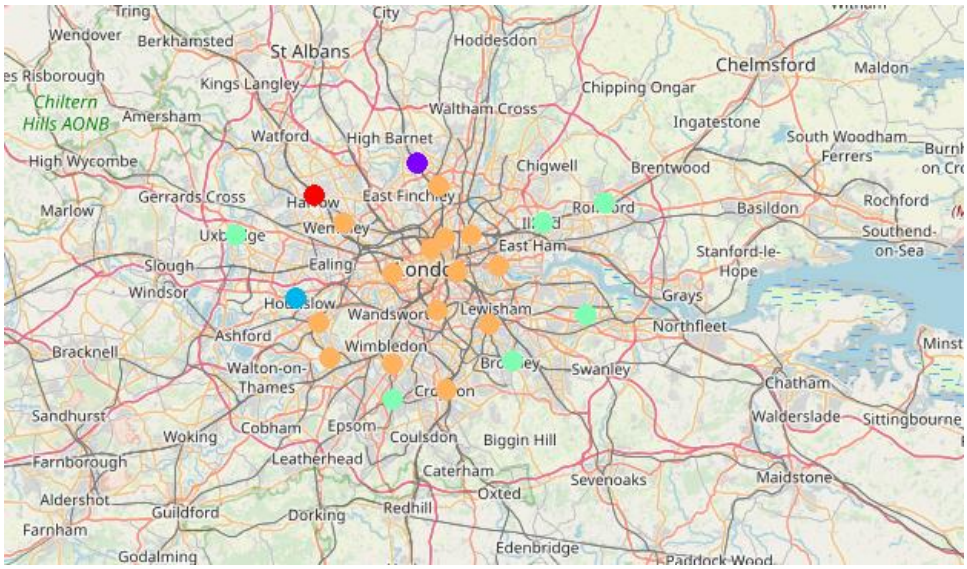
London

Further, the cluster labels curated are added to the data frame to get the desired results of segmenting the neighborhood based upon the most common venues in its vicinity.

Again, London and New York City's neighborhoods are visualized utilizing the python 'folium' library. Following the maps are generated which show the desired segmentation of two cities' neighborhoods:



NYC's Clustering map



London’s clustering map

10.Results

10.1NYC:

Following are the results of 8 Clusters of New York City data frame:

Cluster 1

```
nyc_cluster_1=nyc_merged.loc[nyc_merged['Cluster Labels'] == 0, nyc_merged.columns[[0]+[1]+ list(range(5, nyc_merged.shape[1]))]]
nyc_cluster_1
```

|:

	Borough	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue
0	Bronx	Wakefield	Donut Shop	Ice Cream Shop	Pharmacy	Laundromat	Gas Station
1	Bronx	Co-op City	Bus Station	Pizza Place	Fast Food Restaurant	Grocery Store	Park
2	Bronx	Eastchester	Bus Station	Caribbean Restaurant	Diner	Convenience Store	Chinese Restaurant
5	Bronx	Kingsbridge	Pizza Place	Sandwich Place	Bar	Latin American Restaurant	Mexican Restaurant
6	Manhattan	Marble Hill	Coffee Shop	Sandwich Place	Yoga Studio	Steakhouse	Tennis Stadium
7	Bronx	Woodlawn	Deli / Bodega	Pizza Place	Pub	Food & Drink Shop	Playground
8	Bronx	Norwood	Pizza Place	Park	Bank	Coffee Shop	American Restaurant

```
for col in nyc_required_column:
    print(nyc_cluster_1[col].value_counts(ascending=False))
    print('-------------------------------------')
```

```
Queens           40
Bronx            40
Brooklyn         35
Staten Island    24
Manhattan         2
Name: Borough, dtype: int64
-------------------------------------
Pizza Place                   32
Deli / Bodega                 15
Pharmacy                      10
Chinese Restaurant             8
Caribbean Restaurant           8
Donut Shop                     7
Bank                           7
Bus Station                    5
Fried Chicken Joint            4
Grocery Store                  4
Bakery                         4
Mexican Restaurant             2
```

‘Bus Stop’ holds massive accountability for this cluster with 7 occurrences in ‘1st Most Common Venue’ across different neighborhoods followed by ‘Italian Restaurant ’ with 5 occurrences. Also,‘Deli / Bodega’ and ‘Bus Stop’ occurs 4 times in ‘2nd Most Common Venue’. To add on, it is inquisitive to know that the majority of these neighborhoods are in ‘Staten Island’ borough of New York City. So,nyc_cluster_1 is a combination of ‘Bus Stop’ and ‘Italian Restaurant’.

Cluster 2

```
nyc_cluster_2=nyc_merged.loc[nyc_merged['Cluster Labels'] == 1, nyc_merged.columns[[0]+[1] + list(range(5, nyc_merged.shape[1]))]]
nyc_cluster_2
```

	Borough	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue
3	Bronx	Fieldston	Bus Station	Music Venue	River	Plaza	Yoga Studio
4	Bronx	Riverdale	Park	Bus Station	Gym	Baseball Field	Food Truck
9	Bronx	Williamsbridge	Dance Studio	Nightclub	Bar	Caribbean Restaurant	Spa
12	Bronx	City Island	Harbor / Marina	Seafood Restaurant	Thrift / Vintage Store	Boat or Ferry	Park
22	Bronx	Port Morris	Furniture / Home Store	Restaurant	Donut Shop	Latin American Restaurant	Spanish Restaurant
27	Bronx	Clason Point	Park	Bus Stop	Pool	Grocery Store	Boat or Ferry
28	Bronx	Throgs Neck	Deli / Bodega	Italian Restaurant	Sports Bar	Asian Restaurant	Coffee Shop
29	Bronx	Country Club	Sandwich Place	Bar	Playground	Yoga Studio	Entertainment Service
34	Bronx	Belmont	Italian Restaurant	Pizza Place	Deli / Bodega	Bakery	Donut Shop
35	Bronx	Souvten Duvvil	Park	Bus Line	Bank	Pharmacy	Thai Restaurant

```
for col in nyc_required_column:
    print(nyc_cluster_2[col].value_counts(ascending=False))
    print('---------------------------------------')
```

```
Manhattan        38
Queens           38
Brooklyn         33
Staten Island    25
Bronx            12
Name: Borough, dtype: int64
---------------------------------------
Italian Restaurant            27
Bar                           13
Coffee Shop                   10
Beach                          9
Park                           6
Hotel                          5
Gym                            4
Clothing Store                 4
Café                           3
Sandwich Place                 3
Dance Studio                   3
```

As we can see on the map_nyc_clusters,cluster1 or nyc_cluster_2 is the biggest cluster‘Pizza Place’ holds massive accountability for this cluster with 32 occurrences‘1st Most Common Venue’ across different neighborhoods followed by ‘Italian Restaurant’ with 29 occurrences. Also, ‘Deli / Bodega’ occurs 22 times in the ‘2nd Most Common Venue’. To add on, it is inquisitive to know that the majority of these neighborhoods are in ‘Queens’ and ‘Brooklyn’ borough of New York City. So,nyc_cluster_2 is a combination of ‘pizza Place’ and ‘Italian Restaurant’.

Cluster 3

```
nyc_cluster_3=nyc_merged.loc[nyc_merged['Cluster Labels'] == 2, nyc_merged.columns[[0]+[1] + list(range(5, nyc_merged.shape[1]))]]
nyc_cluster_3
```

]:

	Borough	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue
193	Queens	Brookville	Deli / Bodega	Yoga Studio	Flea Market	Event Space	Exhibit

```
for col in nyc_required_column:
    print(nyc_cluster_3[col].value_counts(ascending=False))
    print('-------------------------------------')
```

```
Queens    1
Name: Borough, dtype: int64
-------------------------------------
Deli / Bodega    1
Name: 1st Most Common Venue, dtype: int64
-------------------------------------
Yoga Studio    1
Name: 2nd Most Common Venue, dtype: int64
-------------------------------------
```

'Pool' holds massive accountability for this cluster with 1 occurrence in the '1st Most Common Venue'. Also, the 'Zoo Exhibit' occurs 1 time in the '2nd Most Common Venue'. To add on, it is inquisitive to know that the majority of these neighborhoods are in 'Brooklyn' borough of New York City. So,nyc_cluster_3 is a combination of 'Pool' and 'Zoo EXhibit'.

Cluster 4

```
nyc_cluster_4=nyc_merged.loc[nyc_merged['Cluster Labels'] ==3, nyc_merged.columns[[0]+[1] + list(range(5, nyc_merged.shape[1]))]]
nyc_cluster_4
```

]:

	Borough	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue
192	Queens	Somerville	Park	Yoga Studio	Ethiopian Restaurant	Event Space	Exhibit
203	Staten Island	Todt Hill	Park	Yoga Studio	Ethiopian Restaurant	Event Space	Exhibit

```
for col in nyc_required_column:
    print(nyc_cluster_4[col].value_counts(ascending=False))
    print('-------------------------------------')
```

```
Staten Island    1
Queens           1
Name: Borough, dtype: int64
-------------------------------------
Park    2
Name: 1st Most Common Venue, dtype: int64
-------------------------------------
Yoga Studio    2
Name: 2nd Most Common Venue, dtype: int64
-------------------------------------
```

It's obvious that 'Park' occurs 4 times in '1st Most Common Venue'.Also 'Zoo Exhibit' occurs 2 times followed by a mix of different places with 1 occurrence in'2nd Most Common Venue'.To add on, it is inquisitive to know that the majority of these neighborhoods are in 'Queens', 'Staten Island' and 'Bronx' borough of New York City. So,nyc_cluster_4 is a dominant 'Park' cluster.

Cluster 5

```
nyc_cluster_5=nyc_merged.loc[nyc_merged['Cluster Labels'] == 4, nyc_merged.columns[[0]+[1] + list(range(5, nyc_merged.shape[1]))]]
nyc_cluster_5
```

]:

	Borough	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue
207	Staten Island	Port Ivory	Bar	Fish Market	Event Service	Event Space	Exhibit

```
for col in nyc_required_column:
    print(nyc_cluster_5[col].value_counts(ascending=False))
    print('--------------------------------------')
```

```
Staten Island    1
Name: Borough, dtype: int64
--------------------------------------
Bar    1
Name: 1st Most Common Venue, dtype: int64
--------------------------------------
Fish Market    1
Name: 2nd Most Common Venue, dtype: int64
--------------------------------------
```

It is clear, that only one neighborhood 'Emerson Hill' is curated under this cluster. So,nyc_cluster_5 is a combination of 'Construction & Landscaping ', 'Zoo Exhibit'.

10.1.1 Discussion

To understand the clusters, three analyses were done, namely:

1. Count of 'Borough'
2. Count of '1st Most Common Venue'
3. Count of '2nd Most Common Venue'

The above information speaks a lot about the ground reality of clustering based on the similarity metrics between the neighborhoods.

Tabulating the results of the k-Mean unsupervised machine learning algorithm:

Cluster	1st Most Common Venu e	2nd Most Common Venue	Borough
1	Bus Stop	Deli / Bodega,Bus Stop	Staten Island
2	Pizza Place	Deli/Bodega,Pizza place	Queens,Brooklyn
3	Pool	Zoo Exhibit	Brooklyn
4	Park	Zoo Exhibit	Queens,Staten Island,Bronx
5	Construction & Landscaping	Zoo Exhibit	Staten Island

The following could be the name of the clusters segmented and curated by k-Means unsupervised machine learning algorithm:

* nyc_cluster_1 — Bus Stop
* nyc_cluster_2 — Pizza Place
* nyc_cluster_3 — Pool
* nyc_cluster_4 — Park
* nyc_cluster_5 — Construction & Landscaping

10.2 London:

Following are the results of 8 Clusters of London data frame:

Cluster 1

```
london_cluster_1=london_merged.loc[london_merged['London Cluster Labels'] == 0, london_merged.columns[[0]+[3] + list(range(5, london_merged.shape[1]))]]
london_cluster_1
```

]:

	Borough	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue
291	Richmond upon Thames	East Sheen	Pub	Coffee Shop	Italian Restaurant	Indian Restaurant	Café
292	Richmond upon Thames	Eel Pie Island	Pub	Coffee Shop	Italian Restaurant	Indian Restaurant	Café
293	Richmond upon Thames	Fulwell	Pub	Coffee Shop	Italian Restaurant	Indian Restaurant	Café
294	Richmond upon Thames	Ham	Pub	Coffee Shop	Italian Restaurant	Indian Restaurant	Café
295	Richmond upon Thames	Hampton	Pub	Coffee Shop	Italian Restaurant	Indian Restaurant	Café
296	Richmond upon Thames	Hampton Hill	Pub	Coffee Shop	Italian Restaurant	Indian Restaurant	Café
297	Richmond upon Thames	Hampton Wick	Pub	Coffee Shop	Italian Restaurant	Indian Restaurant	Café
298	Richmond upon Thames	Kew	Pub	Coffee Shop	Italian Restaurant	Indian Restaurant	Café
299	Richmond upon Thames	Mortlake	Pub	Coffee Shop	Italian Restaurant	Indian Restaurant	Café

```
for col in london_required_column:
    print(london_cluster_1[col].value_counts(ascending=False))
    print('-------------------------------------')
```

```
Richmond upon Thames    17
Name: Borough, dtype: int64
-------------------------------------
Pub    17
Name: 1st Most Common Venue, dtype: int64
-------------------------------------
Coffee Shop    17
Name: 2nd Most Common Venue, dtype: int64
-------------------------------------
```

‘Bus Stop ’ holds massive accountability for this cluster with 27 occurrences in ‘1st Most Common Venue’. Also, ‘Café’ occurs whopping 27 times in ‘2nd Most Common Venue’. To add on, it is inquisitive to know that the majority of these neighborhoods are in ‘Barnet’ borough of London City. So, Cluster 1 is a combination of ‘Bus Stop’ and ‘Café’.

Cluster 2

```
london_cluster_2=london_merged.loc[london_merged['London Cluster Labels'] == 1, london_merged.columns[[0]+[3] + list(range(5, london_merged.shape[1]))]]
london_cluster_2
```

]:

	Borough	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue
0	Barnet	Barnet Gate	Café	Bus Stop	Turkish Restaurant	Yoga Studio	Fast Food Restaurant
1	Barnet	Brent Cross	Café	Bus Stop	Turkish Restaurant	Yoga Studio	Fast Food Restaurant
2	Barnet	Childs Hill	Café	Bus Stop	Turkish Restaurant	Yoga Studio	Fast Food Restaurant
4	Barnet	Colindale	Café	Bus Stop	Turkish Restaurant	Yoga Studio	Fast Food Restaurant
5	Barnet	Colney Hatch	Café	Bus Stop	Turkish Restaurant	Yoga Studio	Fast Food Restaurant
6	Barnet	East Barnet	Café	Bus Stop	Turkish Restaurant	Yoga Studio	Fast Food Restaurant
7	Barnet	East Finchley	Café	Bus Stop	Turkish Restaurant	Yoga Studio	Fast Food Restaurant
8	Barnet	Edgware	Café	Bus Stop	Turkish Restaurant	Yoga Studio	Fast Food Restaurant
9	Barnet	Finchley	Café	Bus Stop	Turkish Restaurant	Yoga Studio	Fast Food Restaurant
10	Barnet	Friern Barnet	Café	Bus Stop	Turkish Restaurant	Yoga Studio	Fast Food Restaurant

```
for col in london_required_column:
    print(london_cluster_2[col].value_counts(ascending=False))
    print('------------------------------------')
```

```
Barnet    27
Name: Borough, dtype: int64
------------------------------------
Café    27
Name: 1st Most Common Venue, dtype: int64
------------------------------------
Bus Stop    27
Name: 2nd Most Common Venue, dtype: int64
------------------------------------
```

'Pub' holds massive accountability for this cluster with 78 occurrences in '1st Most Common Venue' across different neighborhoods followed by 'Coffee Shop' with 23 occurrences. Also,'Coffee Shop' occurs 51 times in '2nd Most Common Venue'. To add on, it is inquisitive to know that the majority of these neighborhoods are in 'Croydon', 'Hackney' and 'Richmond upon Thames' borough of New London. So,london_cluster_2 is a dominant 'Pub' cluster.

Cluster 3

```
london_cluster_3=london_merged.loc[london_merged['London Cluster Labels'] == 2, london_merged.columns[[0]+[3] + list(range(5, london_merged.shape[1]))]]
london_cluster_3
```

]:

	Borough	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue
28	Bexley	Albany Park	Pub	Clothing Store	Coffee Shop	Fast Food Restaurant	Italian Restaurant
29	Bexley	Blendon	Pub	Clothing Store	Coffee Shop	Fast Food Restaurant	Italian Restaurant
30	Bexley	Colyers	Pub	Clothing Store	Coffee Shop	Fast Food Restaurant	Italian Restaurant
31	Bexley	Crayford	Pub	Clothing Store	Coffee Shop	Fast Food Restaurant	Italian Restaurant
32	Bexley	Crook Log	Pub	Clothing Store	Coffee Shop	Fast Food Restaurant	Italian Restaurant
33	Bexley	Crossness	Pub	Clothing Store	Coffee Shop	Fast Food Restaurant	Italian Restaurant
34	Bexley	East Wickham	Pub	Clothing Store	Coffee Shop	Fast Food Restaurant	Italian Restaurant
35	Bexley	Erith	Pub	Clothing Store	Coffee Shop	Fast Food Restaurant	Italian Restaurant
36	Bexley	Foots Cray	Pub	Clothing Store	Coffee Shop	Fast Food Restaurant	Italian Restaurant

```
for col in london_required_column:
    print(london_cluster_3[col].value_counts(ascending=False))
    print('-------------------------------------')
```

```
Bromley       28
Havering      20
Bexley        20
Hillingdon    17
Redbridge     11
Sutton         3
Name: Borough, dtype: int64
-------------------------------------
Coffee Shop       45
Clothing Store    31
Pub               23
Name: 1st Most Common Venue, dtype: int64
-------------------------------------
Clothing Store          68
Coffee Shop             20
Fast Food Restaurant    11
Name: 2nd Most Common Venue, dtype: int64
```

'Bed & Breakfast' holds massive accountability for this cluster with 12 occurrences in '1st Most Common Venue'. Also, 'Café ' occurs 27 times in '2nd Most Common Venue'. To add on, and the majority of these neighborhoods are in 'Hounslow' borough of London City. So, Cluster3 is a combination of 'Café' and 'Bus stop'.

Cluster 4

```
london_cluster_4=london_merged.loc[london_merged['London Cluster Labels'] == 3, london_merged.columns[[0]+[3] + list(range(5, london_merged.shape[1]))]]
london_cluster_4
```

|:

	Borough	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue
203	Hounslow	Cranford	Bed & Breakfast	Café	Park	Chinese Restaurant	Fast Food Restaurant
204	Hounslow	East Bedfont	Bed & Breakfast	Café	Park	Chinese Restaurant	Fast Food Restaurant
205	Hounslow	Feltham	Bed & Breakfast	Café	Park	Chinese Restaurant	Fast Food Restaurant
207	Hounslow	Gunnersbury	Bed & Breakfast	Café	Park	Chinese Restaurant	Fast Food Restaurant
208	Hounslow	Hanworth	Bed & Breakfast	Café	Park	Chinese Restaurant	Fast Food Restaurant
209	Hounslow	Hatton	Bed & Breakfast	Café	Park	Chinese Restaurant	Fast Food Restaurant
210	Hounslow	Heston	Bed & Breakfast	Café	Park	Chinese Restaurant	Fast Food Restaurant
211	Hounslow	Hounslow	Bed & Breakfast	Café	Park	Chinese Restaurant	Fast Food Restaurant
212	Hounslow	Isleworth	Bed & Breakfast	Café	Park	Chinese Restaurant	Fast Food Restaurant
213	Hounslow	Lampton	Bed & Breakfast	Café	Park	Chinese Restaurant	Fast Food Restaurant
214	Hounslow	Osterley	Bed & Breakfast	Café	Park	Chinese Restaurant	Fast Food Restaurant

```
for col in london_required_column:
    print(london_cluster_4[col].value_counts(ascending=False))
    print('--------------------------------------')
```

```
Hounslow    12
Name: Borough, dtype: int64
--------------------------------------
Bed & Breakfast    12
Name: 1st Most Common Venue, dtype: int64
--------------------------------------
Café    12
Name: 2nd Most Common Venue, dtype: int64
--------------------------------------
```

‘Clothing Store’ holds massive accountability for this cluster with 78 occurrences in ‘1st Most Common Venue’ across different neighborhoods. Also, ‘Coffee Shop ’ occurs 67 times in ‘2nd Most Common Venue’. To add on, and the majority of these neighborhoods are in ‘Bromley’ borough of London City. So, Cluster4 is a combination of ‘Clothing Store’ and ‘Coffee Shop’.

Cluster 5

```
london_cluster_5=london_merged.loc[london_merged['London Cluster Labels'] == 4, london_merged.columns[[0]+[3] + list(range(5, london_merged.shape[1]))]]
london_cluster_5
```

]:

	Borough	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue
3	Barnet	Church End	Coffee Shop	Hotel	Clothing Store	Bar	Sporting Goods Shop
48	Brent	Brent Park	Coffee Shop	Hotel	Clothing Store	Bar	Sporting Goods Shop
49	Brent	Church End	Coffee Shop	Hotel	Clothing Store	Bar	Sporting Goods Shop
50	Brent	Dollis Hill	Coffee Shop	Hotel	Clothing Store	Bar	Sporting Goods Shop
51	Brent	Harlesden	Coffee Shop	Hotel	Clothing Store	Bar	Sporting Goods Shop
52	Brent	Kensal Green	Coffee Shop	Hotel	Clothing Store	Bar	Sporting Goods Shop
53	Brent	Kingsbury	Coffee Shop	Hotel	Clothing Store	Bar	Sporting Goods Shop
54	Brent	Neasden	Coffee Shop	Hotel	Clothing Store	Bar	Sporting Goods Shop
55	Brent	Preston	Coffee Shop	Hotel	Clothing Store	Bar	Sporting Goods Shop
56	Brent	Queen's Park	Coffee Shop	Hotel	Clothing Store	Bar	Sporting Goods Shop
57	Brent	Stonebridge	Coffee Shop	Hotel	Clothing Store	Bar	Sporting Goods Shop

```
for col in london_required_column:
    print(london_cluster_5[col].value_counts(ascending=False))
    print('------------------------------------')
```

```
Croydon                   20
Hackney                   18
Camden                    14
Brent                     14
Kingston upon Thames      12
Islington                 10
Southwark                 10
Lambeth                    9
Kensington and Chelsea     8
Barnet                     1
Name: Borough, dtype: int64
------------------------------------
Pub                     61
Coffee Shop             27
Café                    19
Caribbean Restaurant     9
Name: 1st Most Common Venue, dtype: int64
------------------------------------
```

'Supermarket' holds massive accountability for this cluster with 16 occurrences in '1st Most Common Venue' across different neighborhoods. Also,'Grocery Store' occurs 16 times in the '2nd Most Common Venue'. To add on, it is inquisitive to know that the majority of these neighborhoods are in 'Lewisham' borough of London City. So, Cluster5 can be termed as 'Supermarket' and 'Grocery Store' dominant cluster.

10.2.1 Discussion

To understand the clusters, three analyses were done, namely:

1. Count of 'Borough'
2. Count of '1st Most Common Venue'
3. Count of '2nd Most Common Venue'

The above information speaks a lot about the ground reality of clustering based on the similarity metrics between the neighborhoods.

Tabulating the results of the k-Mean unsupervised machine learning algorithm:

Cluster	1st Most Common Venue	2nd Most Common Venue	Borough
1	Bus Stop	Café	Barnet
2	Pub	Coffee Shop	Croydon,Hackney
3	Bed & Breakfast	Café	Hounslow
4	Clothing Store	Coffee Shop	Bromley
5	Supermarket	Grocery Store	Lewisham

The following could be the name of the clusters segmented and curated by k-Means unsupervised machine learning algorithm:

* london_cluster_1 — Café
* london_cluster_2 — Pub
* london_cluster_3 — Bed & Breakfast
* london_cluster_4 — Coffee Shop
* london_cluster_5 — Market

11.Conclusion

One application of the Clustering Algorithm, k-Means or others, to a multi-dimensional dataset, is that it yields a very inquisitive result that can be curated which helps to understand and visualize the data. The neighborhoods of New York City and London were very briefly segmented into 5 clusters and upon analysis, it was possible to rename them upon the basis of the categories of venues in and around that neighborhood. Compared to NYC, it seems that boroughs and neighborhoods of London are ideal places for having a brunch or a cup of tea and taking a break for a little while as we can see ‘Coffeeshop’ and ‘Café’ are the most common venues in this city. The results of this project can be improved and made more inquisitive by using current London’s dataset and New York City’s dataset along with API platforms which are more interested in Cafes and Coffee shops as Venues. The scope of this project can be expanded

further to understand the dynamics of each neighborhood and suggest a new vendor a profitable location to open up his or her cafe.

References

Notebook created by Alex Aklson and Polong Lin for the ‘Applied Data Science Capstone’ course on Coursera