Battle of Neighbourhoods (week 2)

The Travelling Student

Clustering the neighbourhoods of London and Toronto

1. Introduction

Studying abroad is an essential part of a lot of students University experience. This is because they not only want to study at a great establishment but also want to explore the new city that they are going to be living in for the next few years. So, you could say that they are tourists who are also studying. We will be focusing on students that want to study in an English-speaking country as they want to develop their knowledge of the language but are having difficulties deciding which country they want to explore.

London and Toronto are quite the popular tourist destinations for people all around the world. They are diverse and multicultural and offer a wide variety of experiences that is widely sought after. We try to group the neighbourhoods of London and Toronto respectively and draw insights to what they look like now.

2. Business problem

The aim is to help international students choose their destinations depending on the experiences that the neighbourhoods have to offer and what they would want to have. This also helps people make decisions if they are thinking about living in London or Toronto. Our findings will help the international students make informed decisions and address any concerns they have including the different kinds of cuisines; provision stores and what the city has to offer.

3. Data Description

We require geolocation data for both London and Toronto. Postal codes in each city serve as a starting point. Using Postal codes, we use can find out the neighbourhoods, boroughs, venues and their most popular venue categories.

3.1 London

To derive our solution, we scrape our data
from: https://en.wikipedia.org/wiki/List_of_areas_of_London

This Wikipedia page has information about all the neighbourhoods, we limit it London.

1. borough: Name of Neighbourhood
2. town: Name of borough
3. postcode: Postal codes for London.

This Wikipedia page lacks information about the geographical locations. To solve this problem, we use ArcGIS API

3.2 ArcGIS API

ArcGIS Online enables you to connect people, locations, and data using interactive maps. Work with smart, data-driven styles and intuitive analysis tools that deliver location intelligence. Share your insights with the world or specific groups.

More specifically, we use ArcGIS to get the geo locations of the neighbourhoods of London. The following columns are added to our initial dataset which prepares our data.

1. latitude: Latitude for Neighbourhood
2. longitude: Longitude for Neighbourhood

3.3 Toronto

To derive our solution, we scrape our data
from: https://en.wikipedia.org/wiki/List_of_postal_codes_of_Canada:_M

1. Neighborhood: Name of Neighbourhood
2. Borough: Name of borough
3. PostalCode: Postal codes for Toronto.

This Wikipedia page lacks information about the geographical locations, so we will be using the link provided for us in week 3 of this course: https://cocl.us/Geospatial_data

3.4 Foursquare API

We will need data about different venues in different neighbourhoods of that specific borough. In order to gain that information, we will use "Foursquare" locational information. Foursquare is a location data provider with information about all manner of venues and events within an area of interest. Such information includes venue names, locations, menus and even photos. As such, the foursquare location platform will be used as the sole data source since all the stated required information can be obtained through the API.

After finding the list of neighbourhoods, we then connect to the Foursquare API to gather information about venues inside each and every neighbourhood. For each neighbourhood, we have chosen the radius to be 500 meters.

The data retrieved from Foursquare contained information of venues within a specified distance of the longitude and latitude of the postcodes. The information obtained per venue as follows:

1. Neighbourhood: Name of the Neighbourhood
2. Neighbourhood Latitude: Latitude of the Neighbourhood
3. Neighbourhood Longitude: Longitude of the Neighbourhood
4. Venue: Name of the Venue
5. Venue Latitude: Latitude of Venue
6. Venue Longitude: Longitude of Venue
7. Venue Category: Category of Venue

Based on all the information collected for both London and Toronto, we have sufficient data to build our model. We cluster the neighbourhoods together based on similar venue categories. We then present our observations and findings. Using this data, the international students can take the necessary decision.

4. Methodology

We will be creating our model with the help of Python so we start off by importing all the required packages.

```
import pandas as pd
import requests
import numpy as np
from bs4 import BeautifulSoup
!pip install geopy
from geopy.geocoders import Nominatim # convert an address into latitude and longitude values
from pandas.io.json import json_normalize  # tranform JSON file into a pandas dataframe
!pip install folium==0.5.0
import folium # map rendering library
# import k-means from clustering stage
from sklearn.cluster import KMeans
# Matplotlib and associated plotting modules
import matplotlib.cm as cm
import matplotlib.colors as colors
```

Figure 1

From *Figure 1* we can see the packages that I used in the assignment. The packages are as followed:

Pandas: To collect and manipulate data in JSON and HTML to use data analysis
requests: Handle http requests
Numpy: To use mathematical operations
Beautiful soup: to import our data set from Wikipedia
Nominatim: To convert addresses into latitude and longitude values
matplotlib: Detailing the generated maps
folium: Generating maps of London and Toronto

sklearn: To import Kmeans which is the machine learning model that we are using.

Some of these packages may not be required for our analysis, but imported them just in case I do need them to save time and makes things easier.
The approach taken here is to explore each of the cities individually, plot the map to show the neighbourhoods being considered and then build our model by clustering all of the similar neighbourhoods together and finally, to plot the new map with the clustered neighbourhoods. We draw insights and then compare and discuss our findings.

4.1 Data collection

In the data collection stage, we begin with collecting the required data for the cities of London and Toronto. We need data that has the postal codes, neighbourhoods and boroughs specific to each of the cities.

To collect data for London, we scrape the List of areas of London Wikipedia page to take the second table using the following code as seen in *Figure 2.1*:

```
url_london="https://en.wikipedia.org/wiki/List_of_areas_of_London"
wiki_london_url=requests.get(url_london)
wiki_london_url
wiki_london_data=pd.read_html(wiki_london_url.text)
wiki_london_data
```

Figure 2.1

The result of this code then resulted in our table seen in *Figure 2.2*:

	Location	London borough	Post town	Postcode district	Dial code	OS grid ref
0	Abbey Wood	Bexley, Greenwich [7]	LONDON	SE2	020	TQ465785
1	Acton	Ealing, Hammersmith and Fulham[8]	LONDON	W3, W4	020	TQ205805
2	Addington	Croydon[8]	CROYDON	CR0	020	TQ375645
3	Addiscombe	Croydon[8]	CROYDON	CR0	020	TQ345665
4	Albany Park	Bexley	BEXLEY, SIDCUP	DA5, DA14	020	TQ478728
...	...	...	...	...	...	...
526	Woolwich	Greenwich	LONDON	SE18	020	TQ435795
527	Worcester Park	Sutton, Kingston upon Thames	WORCESTER PARK	KT4	020	TQ225655
528	Wormwood Scrubs	Hammersmith and Fulham	LONDON	W12	020	TQ225815
529	Yeading	Hillingdon	HAYES	UB4	020	TQ115825
530	Yiewsley	Hillingdon	WEST DRAYTON	UB7	020	TQ063804

531 rows × 6 columns

Figure 2.2

To collect the data from Toronto, we had to scrape the data from the List of postal codes of Canada: M Wikipedia page, which we used the package Beautiful soup to extract the data from the table on the Wikipedia page shown in *Figure 2.3*:

```
source=requests.get("https://en.wikipedia.org/wiki/List_of_postal_codes_of_Canada:_M").text
soup=BeautifulSoup(source,"lxml")
table=soup.find("table")
table_rows=table.tbody.find_all("tr")
res=[]
for tr in table_rows:
    td=tr.find_all("td")
    row=[tr.text for tr in td]
    # Only process the cells that have an assigned borough. Ignore cells with a borough that is Not assigned.
    if row != [] and row[1] != "Not assigned\n":
        # If a cell has a borough but a "Not assigned" neighborhood, then the neighborhood will be the same as the borough.
        if "Not assigned\n" in row[2]:
            row[2]=row[1]
        res.append(row)
# Dataframe with 3 columns
df=pd.DataFrame(res,columns=["PostalCode","Borough","Neighborhood"])
df.head()
```

Figure 2.3

And this then resulted in the following table shown in *Figure 2.4*:

	PostalCode	Borough	Neighborhood
0	M3A\n	North York\n	Parkwoods\n
1	M4A\n	North York\n	Victoria Village\n
2	M5A\n	Downtown Toronto\n	Regent Park, Harbourfront\n
3	M6A\n	North York\n	Lawrence Manor, Lawrence Heights\n
4	M7A\n	Downtown Toronto\n	Queen's Park, Ontario Provincial Government\n

Figure 2.4

4.2 Data pre-processing

In the London data frame, we had to replace the spaces with underscores in the column titles and also had to remove the square brackets with numbers in them ("[7]" shown in *Figure 2.2*). we used the following codes seen below in *Figure 3.1*:

```
wiki_london_data.rename(columns=lambda x: x.strip().replace(" ", "_"), inplace=True)
df1["borough"]=df1["borough"].map(lambda x: x.rstrip("]").rstrip("0123456789").rstrip("["))
```

Figure 3.1

For Toronto, we had to remove the "\n" from the table (shown in *Figure 2.4*) in order for us to work with and analyse the data inside of the table. The codes that we used are shown in *Figure 3.2*:

```
df["Neighborhood"]=df["Neighborhood"].str.replace("\n","")
df["PostalCode"]=df["PostalCode"].str.replace("\n","")
df["Borough"]=df["Borough"].str.replace("\n","")
df.head()
```

Figure 3.2

4.3 Feature Selection

For the London data, we only needed the columns “boroughs”, “Postal codes” and “Post town” for further steps. Therefore, we can remove the columns “Location”, “Dial codes” and “OS grid ref”. We did this by using the following code shown below in *Figure 4*:

```
df1=wiki_london_data.drop([wiki_london_data.columns[0],
wiki_london_data.columns[4],wiki_london_data.columns[5]],axis=1)
```

Figure 4

For the Toronto data, we already had the table in the format that we need with, “PostalCode”, “Borough” and “Neighbourhood”, resulting in us not needing to drop any of the columns.

4.4 Feature Engineering

The data from the London table contains information about cities in the UK as well as London. Because we are only focussing on London, we had to then select only the neighbourhoods in London as seen by the following code below in *Figure 5.1*:

```
df1=df1[df1["town"].str.contains("LONDON")]
```

Figure 5.1

In comparison to the London data, the Toronto data was already how we wanted it with the data just being from the city of Toronto.

Looking over our London dataset, we can see that we don't have the geolocation data. We need to extrapolate the missing data for our neighbourhoods. We perform this by leveraging the ArcGIS API to get the latitude and longitude of our London neighbourhood data. We do this by using the codes shown below in *Figure 5.2*:

```
from arcgis.geocoding import geocode
from arcgis.gis import GIS
g=GIS()
def get_x_y_uk(address1):
    lat_coords=0
    lng_coords=0
    g=geocode(address="{},London,England,GBR".format(address1))[0]
    lng_coords=g["location"]["x"]
    lat_coords=g["location"]["y"]
    return str(lat_coords) +","+ str(lng_coords)
```

Figure 5.2

We then have to pass the through the postal codes of London to get the geographical coordinates as shown in *Figure 5.3*:

```
geo_coordinates_uk=df1["post_code"]
coordinates_latlng_uk=geo_coordinates_uk.apply(lambda x: get_x_y_uk(x))
```

Figure 5.3

We then proceed to merging the London data with the geographical coordinates in order for us to continue with our analysis as shown in *Figure 5.4*:

```
london_merged=pd.concat([df1,lat_uk.astype(float),lng_uk.astype(float)],axis=1)
london_merged.columns=["borough","town","post_code","latitude","longitude"]
london_merged
```

	borough	town	post_code	latitude	longitude
0	Bexley, Greenwich	LONDON	SE2	51.49245	0.12127
1	Ealing, Hammersmith and Fulham	LONDON	W3, W4	51.51324	-0.26746
6	City	LONDON	EC3	51.51200	-0.08058
7	Westminster	LONDON	WC2	51.51651	-0.11968
9	Bromley	LONDON	SE20	51.41009	-0.05683
...	...	...	...	...	...
521	Redbridge	LONDON	IG8, E18	51.58977	0.03052
522	Redbridge, Waltham Forest	LONDON, WOODFORD GREEN	IG8	51.50642	-0.12721
525	Barnet	LONDON	N12	51.61592	-0.17674
526	Greenwich	LONDON	SE18	51.48207	0.07143
528	Hammersmith and Fulham	LONDON	W12	51.50645	-0.23691

308 rows × 5 columns

Figure 5.4

As we can see from the table in *Figure 5.4*, we have the geographical locations in the same table as our London data.

For the Toronto data, we are going to get the Geolocations of the Toronto neighbourhoods from the link that was provided for us in the assignment in week 3 of this module. This makes things easier for us compared with the London data. This is shown in *Figure 5.5*:

```
df_geo_coor=pd.read_csv("https://cocl.us/Geospatial_data")
```

Figure 5.5

We now have the geographical co-ordinates of the Toronto Neighbourhoods. We proceed with Merging our source data with the geographical co-ordinates to make our dataset ready for the next stage. This is shown in *Figure 5.6*:

```
df_toronto=pd.merge(df,df_geo_coor,how="left",left_on="PostalCode",right_on="Postal Code")
# remove the "Postal Code" column
df_toronto.drop("Postal Code",axis=1,inplace=True)
```

	PostalCode	Borough	Neighborhood	Latitude	Longitude
0	M3A	North York	Parkwoods	43.753259	-79.329656
1	M4A	North York	Victoria Village	43.725882	-79.315572
2	M5A	Downtown Toronto	Regent Park, Harbourfront	43.654260	-79.360636
3	M6A	North York	Lawrence Manor, Lawrence Heights	43.718518	-79.464763
4	M7A	Downtown Toronto	Queen's Park, Ontario Provincial Government	43.662301	-79.389494

Figure 5.6

As we can see from the table in *Figure 5.6*, we have the geographical locations in the same table as our Toronto data.

Finally, we get the geocodes of Toronto to help us with the plotting of the maps in the next section. Shown in *Figure 5.7*:

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

Figure 5.7

4.5 Visualising the neighbourhoods in London and Toronto

Now that our datasets are ready, using the “Folium” package, we can visualise the maps of London and Toronto respectively, with the neighbourhoods that we collected.

Shown below in *Figure 6.1* is the neighbourhood map of London:

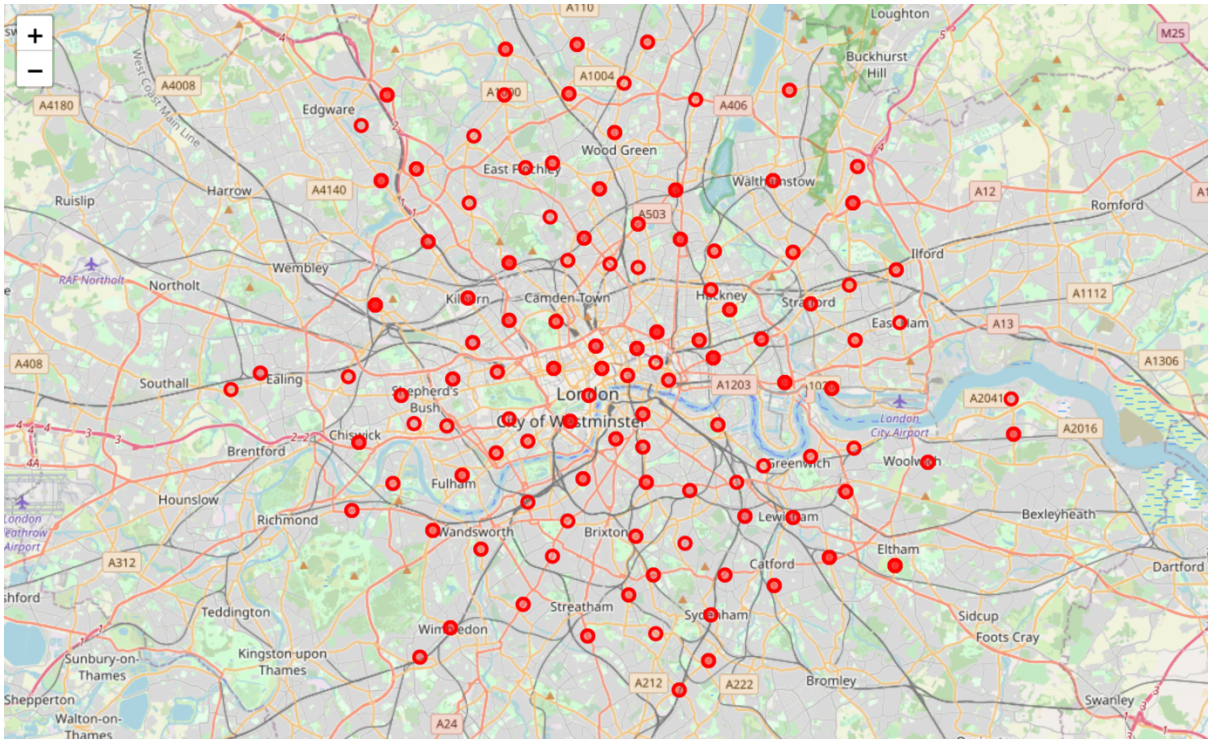

Figure 6.1

Shown below in *Figure 6.2* is the neighbourhood map of Toronto:

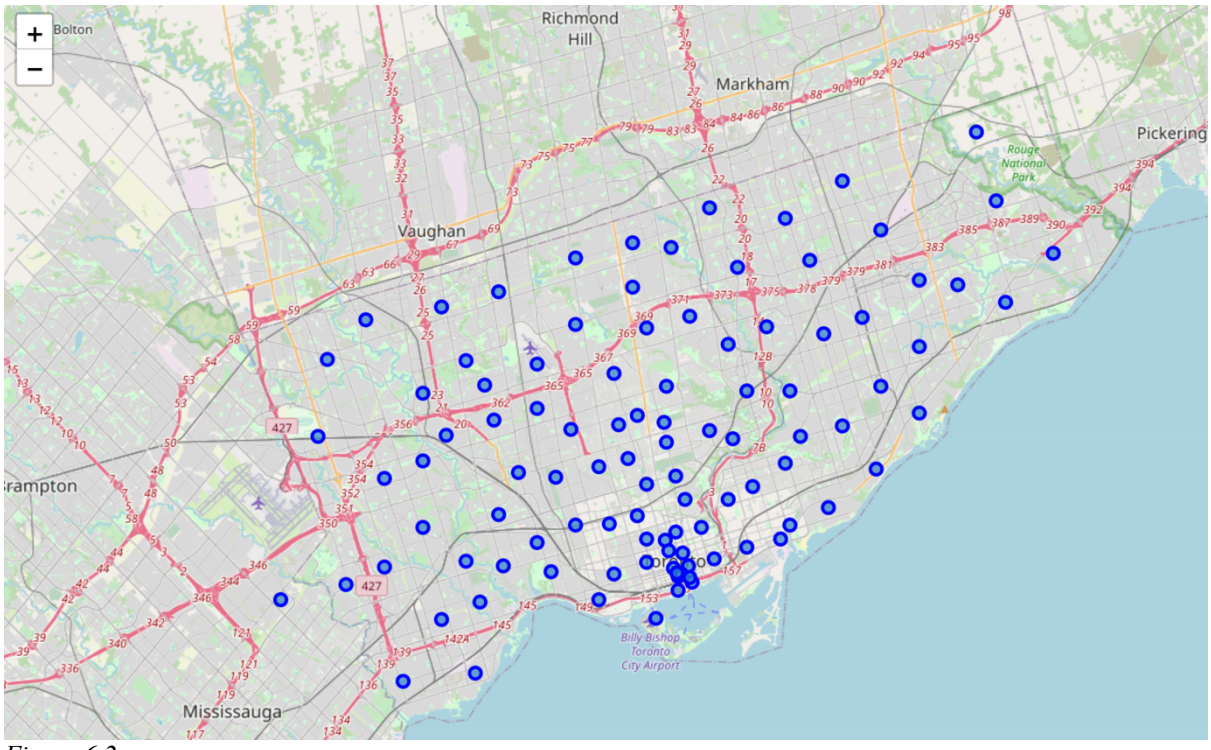

Figure 6.2

Now that we have visualised the neighbourhoods, we need to find out what each neighbourhood is like and what are the common venues and the venue categories within a 500m radius.

This is where “Foursquare” comes into play. With the help of Foursquare we define a function which collects information pertaining to each neighbourhood including that of the name of the neighbourhood, geo-coordinates, venue and the venue categories, which we need in order to do our analysis. In *Figure 6.3* we can see the codes that I used in order to do this, with the resulting table for London.

```
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={}".format(
            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"]["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 Category"]
    return(nearby_venues)
```

	Neighbourhood	Neighbourhood Latitude	Neighbourhood Longitude	Venue	Venue Category
0	Bexley, Greenwich	51.49245	0.12127	Lesnes Abbey	Historic Site
1	Bexley, Greenwich	51.49245	0.12127	Sainsbury's	Supermarket
2	Bexley, Greenwich	51.49245	0.12127	Lidl	Supermarket
3	Bexley, Greenwich	51.49245	0.12127	Abbey Wood Railway Station (ABW)	Train Station
4	Bexley, Greenwich	51.49245	0.12127	Bean @ Work	Coffee Shop

Figure 6.3

Similarly, we done the same for Toronto, shown below in *Figure 6.4*:

```
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={}".format(
            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=["Neighborhood",
                  "Neighborhood Latitude",
                  "Neighborhood Longitude",
                  "Venue",
                  "Venue Latitude",
                  "Venue Longitude",
                  "Venue Category"]
    return(nearby_venues)
```

Venue Category	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude
Accessories Store	Lawrence Manor, Lawrence Heights	43.778517	-79.346556	Sunglass Hut	43.777661	-79.344692
Airport	Downsview	43.737473	-79.394420	Toronto Downsview Airport (YZD)	43.738883	-79.396033
Airport Food Court	CN Tower, King and Spadina, Railway Lands, Har...	43.628947	-79.394420	Billy Bishop Café	43.631132	-79.396139
Airport Gate	CN Tower, King and Spadina, Railway Lands, Har...	43.628947	-79.394420	Gate 8	43.631536	-79.394570
Airport Lounge	CN Tower, King and Spadina, Railway Lands, Har...	43.628947	-79.394420	Porter Lounge	43.631360	-79.395756
...	...	...	...	...	...	...
Wine Bar	Toronto Dominion Centre, Design Exchange	43.657952	-79.375418	The National Club	43.659128	-79.380574
Wine Shop	Regent Park, Harbourfront	43.654260	-79.360636	Wine Rack	43.656573	-79.356928
Wings Joint	Mimico NW, The Queensway West, South of Bloor,...	43.628841	-79.520999	Wingporium	43.630275	-79.518169
Women's Store	Fairview, Henry Farm, Oriole	43.778517	-79.346556	Want Boutique	43.778111	-79.343660
Yoga Studio	University of Toronto, Harbord	43.715383	-79.321558	YogaSpace	43.714070	-79.324335

275 rows × 6 columns

Figure 6.4

4.6 One Hot Encoding

Since we are trying to find out what the different kinds of venue categories that are present in each neighbourhood and also calculating the top 10 common venues to base our similarity on, we use the “One Hot Encoding” to work with our categorical datatypes of the venue categories. This helps to convert the categorical data into numeric data.

We won't be using label encoding in this situation since label encoding might cause our machine learning model to have a bias or a sort of ranking which we are trying to avoid by using One Hot Encoding.

We perform one hot encoding and then calculate the mean of the grouped venue categories for each of the neighbourhoods as we can see below for London in *Figure 7.1*:

```
London_venue_cat=pd.get_dummies(venues_in_London[["Venue Category"]],prefix="",prefix_sep="")
London_venue_cat["Neighbourhood"]=venues_in_London["Neighbourhood"]
# moving neighborhood column to the first column
fixed_columns=[London_venue_cat.columns[-1]]+list(London_venue_cat.columns[:-1])
London_venue_cat=London_venue_cat[fixed_columns]
London_grouped=London_venue_cat.groupby("Neighbourhood").mean().reset_index()
```

	Neighbourhood	Accessories Store	Adult Boutique	African Restaurant	American Restaurant	Antique Shop	Arcade	Arepa Restaurant	Argentinian Restaurant	Art Gallery	...	Vietnamese Restaurant	Warehouse Store	Whisky Bar	Wine Bar	Wine Shop	Wings Joint
0	Barnet	0.0	0.0	0.0	0.001795	0.0	0.0	0.0	0.007181	0.0	...	0.007181	0.0	0.0	0.0	0.0	0.0
1	Barnet, Brent, Camden	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.000000	0.0	...	0.000000	0.0	0.0	0.0	0.0	0.0
2	Bexley	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.000000	0.0	...	0.000000	0.0	0.0	0.0	0.0	0.0
3	Bexley, Greenwich	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.000000	0.0	...	0.000000	0.0	0.0	0.0	0.0	0.0
4	Bexley, Greenwich	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.000000	0.0	...	0.000000	0.0	0.0	0.0	0.0	0.0

5 rows × 303 columns

Figure 7.1

Similarly, we can see that we have done the same for Toronto as shown in *Figure 7.2*:

```
# one hot encoding
toronto_onehot=pd.get_dummies(toronto_venues[["Venue Category"]],prefix="",prefix_sep="")
# add neighborhood column back to dataframe
toronto_onehot["Neighborhood"]=toronto_venues["Neighborhood"]
# move neighborhood column to the first column
fixed_columns=[toronto_onehot.columns[-1]]+list(toronto_onehot.columns[:-1])
toronto_onehot=toronto_onehot[fixed_columns]
toronto_grouped=toronto_onehot.groupby("Neighborhood").mean().reset_index()
```

	Neighborhood	Yoga Studio	Accessories Store	Airport	Airport Food Court	Airport Gate	Airport Lounge	Airport Service	Airport Terminal	American Restaurant	...	Train Station	Truck Stop	Vegetarian / Vegan Restaurant	Video Game Store	Vietnamese Restaurant	Warehouse Store	Wine Bar
0	Agincourt	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	Alderwood, Long Branch	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	Bathurst Manor, Wilson Heights, Downsview North	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	Bayview Village	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	Bedford Park, Lawrence Manor East	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.038462	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 275 columns

Figure 7.2

4.7 Top Venues in each Neighbourhoods

In our next step, we need to rank and label the top venue categories in our neighbourhood. Let's define a function to get the top venue categories in the neighbourhood. There are many categories, we will consider top 10 categories to avoid data skew, and then defining a function to label them accurately as seen below for London in *Figure 8.1*:

```
def return_most_common_venues(row,num_top_venues):
    row_categories=row.iloc[1:]
    row_categories_sorted=row_categories.sort_values(ascending=False)
     return row_categories_sorted.index.values[0:num_top_venues]
num_top_venues=10
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 for London
neighborhoods_venues_sorted_london=pd.DataFrame(columns=columns)
neighborhoods_venues_sorted_london["Neighbourhood"]=London_grouped["Neighbourhood"]
for ind in np.arange(London_grouped.shape[0]):
    neighborhoods_venues_sorted_london.iloc[ind,1:]=return_most_common_venues(London_grouped.iloc[ind,:],num_top_venues)
neighborhoods_venues_sorted_london.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	Barnet	Coffee Shop	Café	Grocery Store	Pub	Supermarket	Pharmacy	Italian Restaurant	Bus Stop	Sushi Restaurant	Turkish Restaurant
1	Barnet, Brent, Camden	Bus Station	Clothing Store	Gym / Fitness Center	Hardware Store	Supermarket	Fish & Chips Shop	Falafel Restaurant	Farmers Market	Fast Food Restaurant	Filipino Restaurant
2	Bexley	Supermarket	Historic Site	Convenience Store	Coffee Shop	Train Station	Platform	Bus Stop	Golf Course	Park	Fish Market
3	Bexley, Greenwich	Daycare	Construction & Landscaping	Bus Stop	Park	Golf Course	Historic Site	Home Service	Sports Club	Discount Store	Diner
4	Bexley, Greenwich	Supermarket	Train Station	Coffee Shop	Convenience Store	Platform	Historic Site	Film Studio	Exhibit	Falafel Restaurant	Farmers Market

Figure 8.1

And also, for Toronto, as seen in *Figure 8.2*:

```
def return_most_common_venues(row,num_top_venues):
    row_categories=row.iloc[1:]
    row_categories_sorted=row_categories.sort_values(ascending=False)
    return row_categories_sorted.index.values[0:num_top_venues]
num_top_venues=10
indicators=["st","nd","rd"]
# create columns according to number of top venues
columns=["Neighborhood"]
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
neighborhoods_venues_sorted=pd.DataFrame(columns=columns)
neighborhoods_venues_sorted["Neighborhood"]=toronto_grouped["Neighborhood"]
for ind in np.arange(toronto_grouped.shape[0]):
    neighborhoods_venues_sorted.iloc[ind, 1:]=return_most_common_venues(toronto_grouped.iloc[ind,:],num_top_venues)
neighborhoods_venues_sorted.head()
```

	Neighborhood	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	Agincourt	Latin American Restaurant	Lounge	Breakfast Spot	Women's Store	Dumpling Restaurant	Distribution Center	Dog Run	Doner Restaurant	Donut Shop	Drugstore
1	Alderwood, Long Branch	Pizza Place	Gym	Dance Studio	Pharmacy	Coffee Shop	Athletics & Sports	Pub	Dog Run	Dim Sum Restaurant	Diner
2	Bathurst Manor, Wilson Heights, Downsview North	Coffee Shop	Bank	Fried Chicken Joint	Pizza Place	Intersection	Supermarket	Ice Cream Shop	Sushi Restaurant	Restaurant	Shopping Mall
3	Bayview Village	Café	Bank	Chinese Restaurant	Japanese Restaurant	Women's Store	Doner Restaurant	Discount Store	Distribution Center	Dog Run	Donut Shop
4	Bedford Park, Lawrence Manor East	Sandwich Place	Coffee Shop	Italian Restaurant	Women's Store	Indian Restaurant	Juice Bar	Breakfast Spot	Liquor Store	Locksmith	Restaurant

Figure 8.2

4.8 Model Building (K means Clustering)

We will now be using K Means Clustering Machine learning algorithm to cluster similar neighbourhoods together. We will be going with the number of clusters as 5. We add our labels to the data and then merge with our neighbourhood venues that is sorted to add latitude

and longitude for each of the neighbourhoods. Firstly, we will see the K means clustering for the London data as see in *Figure 9.1*:

```
# set number of clusters
k_num_clusters=5
London_grouped_clustering=London_grouped.drop("Neighbourhood",1)
# run k-means clustering
kmeans_london=KMeans(n_clusters=k_num_clusters,random_state=0).fit(London_grouped_clustering)
neighborhoods_venues_sorted_london.insert(0,"Cluster Labels",kmeans_london.labels_+1)
london_data=london_merged
london_data=london_data.join(neighborhoods_venues_sorted_london.set_index("Neighbourhood"),on="borough")
```

borough	town	post_code	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 Most Common Venue	8th Most Common Venue	9th Most Common Venue	10th Most Common Venue
Bexley, Greenwich	LONDON	SE2	51.49245	0.12127	4	Supermarket	Train Station	Coffee Shop	Convenience Store	Platform	Historic Site	Film Studio	Exhibit	Falafel Restaurant	Farmers Market
Ealing, nmersmith nd Fulham	LONDON	W3, W4	51.51324	-0.26746	2	Grocery Store	Indian Restaurant	Park	Breakfast Spot	Train Station	Zoo Exhibit	Film Studio	Exhibit	Falafel Restaurant	Farmers Market
City	LONDON	EC3	51.51200	-0.08058	1	Hotel	Italian Restaurant	Coffee Shop	Gym / Fitness Center	Pub	Restaurant	Wine Bar	Sandwich Place	Salad Place	Scenic Lookout
estminster	LONDON	WC2	51.51651	-0.11968	1	Hotel	Coffee Shop	Pub	Café	Sandwich Place	Italian Restaurant	Theater	Restaurant	Hotel Bar	Sushi Restaurant
Bromley	LONDON	SE20	51.41009	-0.05683	1	Supermarket	Grocery Store	Convenience Store	Fast Food Restaurant	Hotel	Park	Café	Historic Site	Gym / Fitness Center	Italian Restaurant

Figure 9.1

Likewise, with the Toronto data as seen below in *Figure 9.2:*

```
# set number of clusters
kclusters=5
toronto_grouped_clustering=toronto_grouped.drop("Neighborhood",1)
# run k-means clustering
kmeans_toronto=KMeans(n_clusters=kclusters,random_state=0).fit(toronto_grouped_clustering)
neighborhoods_venues_sorted.insert(0,"Cluster Labels",kmeans_toronto.labels_+1)
toronto_merged=df_toronto
# merge toronto_merged with df_toronto to add latitude/longitude for each neighborhood
toronto_merged=toronto_merged.join(neighborhoods_venues_sorted.set_index("Neighborhood"),on="Neighborhood")
```

Borough	Neighborhood	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 Most Common Venue	8th Most Common Venue	9th Most Common Venue	10th Most Common Venue
North York	Parkwoods	43.753259	-79.329656	2.0	Park	Fireworks Store	Food & Drink Shop	Donut Shop	Diner	Discount Store	Distribution Center	Dog Run	Doner Restaurant	Drugstore
North York	Victoria Village	43.725882	-79.315572	1.0	Hockey Arena	Pizza Place	Coffee Shop	Portuguese Restaurant	Women's Store	Dim Sum Restaurant	Diner	Discount Store	Distribution Center	Dog Run
Downtown Toronto	Regent Park, Harbourfront	43.654260	-79.360636	1.0	Coffee Shop	Pub	Bakery	Park	Breakfast Spot	Café	Theater	Gym / Fitness Center	Electronics Store	Restaurant
North York	Lawrence Manor, Lawrence Heights	43.718518	-79.464763	1.0	Clothing Store	Furniture / Home Store	Vietnamese Restaurant	Athletics & Sports	Coffee Shop	Miscellaneous Shop	Event Space	Boutique	Accessories Store	Ethiopian Restaurant
Downtown Toronto	Queen's Park, Ontario Provincial Government	43.662301	-79.389494	1.0	Coffee Shop	Sushi Restaurant	College Cafeteria	Beer Bar	Bank	Bar	Portuguese Restaurant	Café	Diner	Yoga Studio

Figure 9.2

4.9 Visualisation of the Clustered Neighbourhoods

We then had to change the column “Cluster Labels” into integers, removing any values that were not numeric and getting rid of any missing data so that we can prepare it for visualisation. We then proceeded to plotting the maps of the clustered neighbourhoods as shown below for London in *Figure 10.1*:

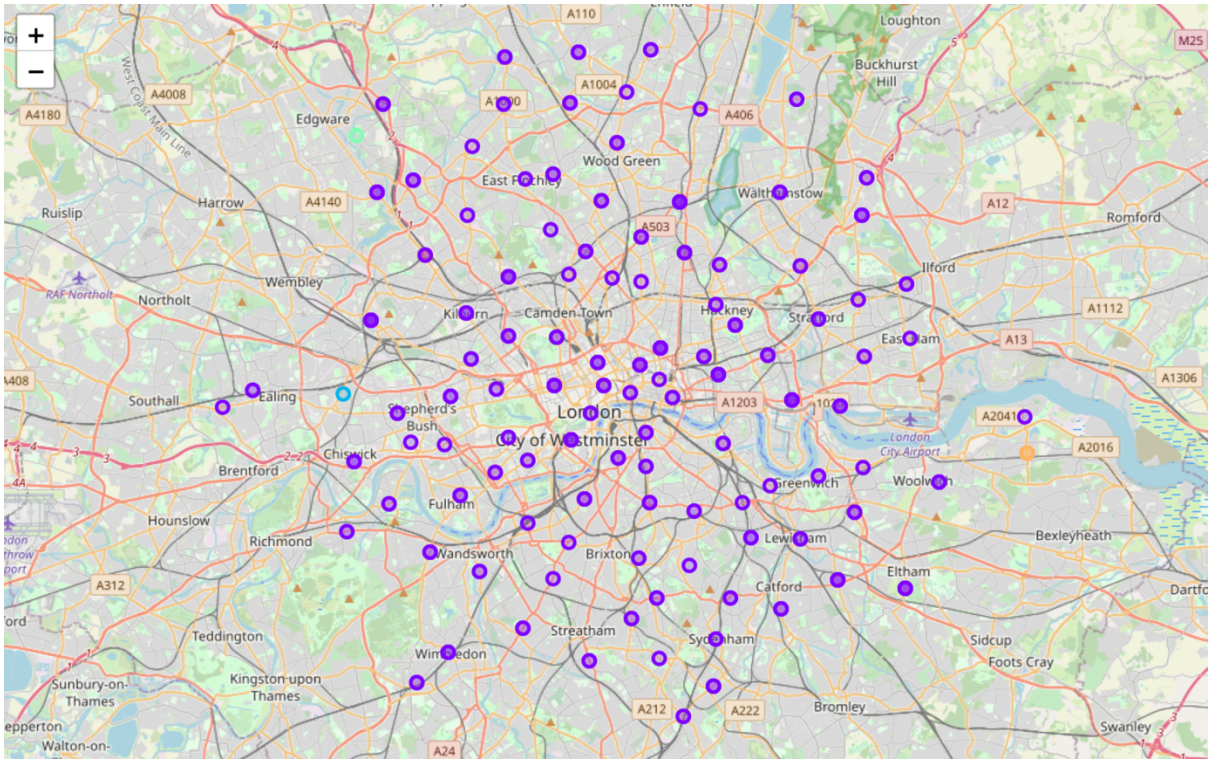

Figure 10.1

And also, the map of clustered neighbourhoods of Toronto shown in *Figure 10.2*:

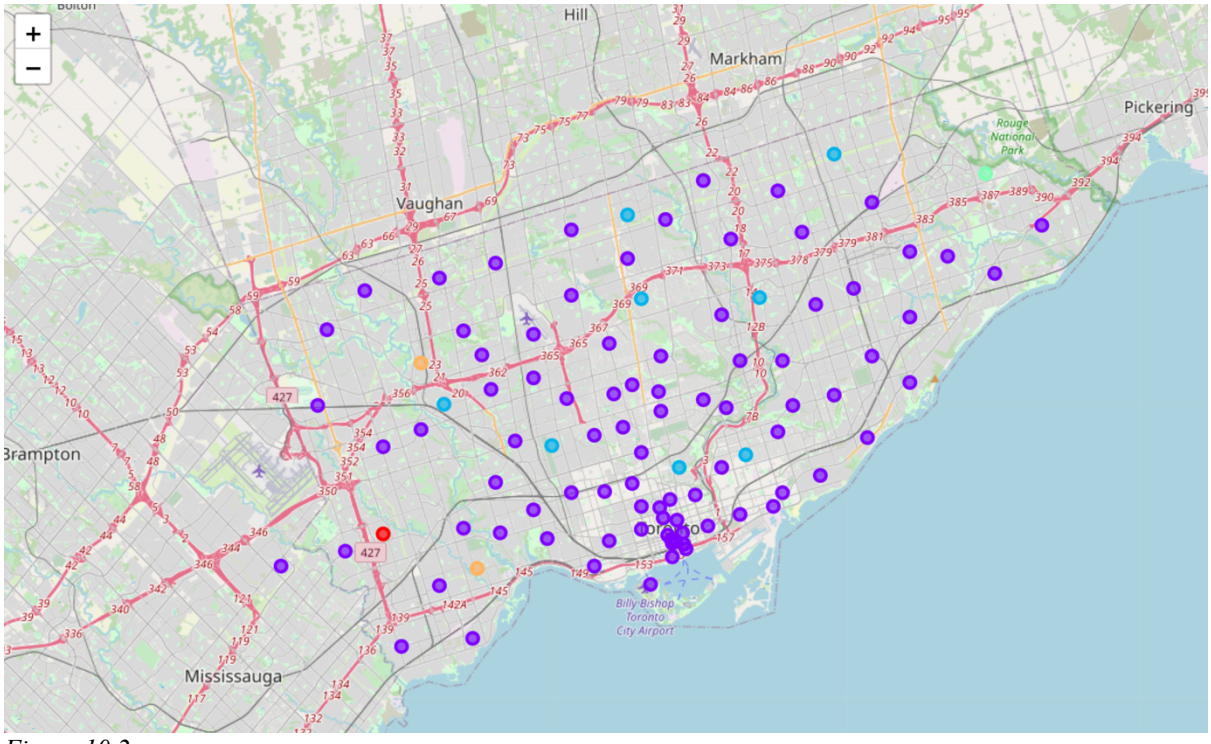

Figure 10.2

4.10 Examining Clusters

We then examined our clusters for each city, London and Toronto respectively, using the following codes that are shown below in *Figure 11*:

```
london_data_nonan.loc[london_data_nonan["Cluster Labels"]==1,london_data_nonan.columns[[1]+
                                                                                       list(range(5,london_data_nonan.shape[1]))]]
london_data_nonan.loc[london_data_nonan["Cluster Labels"]==2,london_data_nonan.columns[[1]+
                                                                                       list(range(5,london_data_nonan.shape[1]))]]
london_data_nonan.loc[london_data_nonan["Cluster Labels"]==3,london_data_nonan.columns[[1]+
                                                                                       list(range(5,london_data_nonan.shape[1]))]]
london_data_nonan.loc[london_data_nonan["Cluster Labels"]==4,london_data_nonan.columns[[1]
                                                                                       +list(range(5,london_data_nonan.shape[1]))]]
london_data_nonan.loc[london_data_nonan["Cluster Labels"]==5,london_data_nonan.columns[[1]
                                                                                       +list(range(5,london_data_nonan.shape[1]))]]

toronto_merged_nonan.loc[toronto_merged_nonan["Cluster Labels"]==1,toronto_merged_nonan.columns[[1]+
                                                                                               list(range(5,toronto_merged_nonan.shape[1]))]]
toronto_merged_nonan.loc[toronto_merged_nonan["Cluster Labels"]==2,toronto_merged_nonan.columns[[1]+
                                                                                               list(range(5,toronto_merged_nonan.shape[1]))]]
toronto_merged_nonan.loc[toronto_merged_nonan["Cluster Labels"]==3,toronto_merged_nonan.columns[[1]
                                                                                              +list(range(5,toronto_merged_nonan.shape[1]))]]
toronto_merged_nonan.loc[toronto_merged_nonan["Cluster Labels"]==4,toronto_merged_nonan.columns[[1]+
                                                                                               list(range(5,toronto_merged_nonan.shape[1]))]]
toronto_merged_nonan.loc[toronto_merged_nonan["Cluster Labels"]==5,toronto_merged_nonan.columns[[1]+
                                                                                               list(range(5,toronto_merged_nonan.shape[1]))]]
```

Figure 11

5. Results and Discussion

The neighbourhoods of London are very multicultural. There are a lot of different cuisines including Indian, Italian, Turkish and Chinese. London seems to take a step further in this direction by having a lot of Restaurants, bars, juice bars, coffee shops, Fish and Chips shop and Breakfast spots. It has a lot of shopping options too with that of the Flea markets, flower shops, fish markets, Fishing stores, clothing stores. The main modes of transport seem to be Buses and trains. For leisure, the neighbourhoods are set up to have lots of parks, golf courses, zoo, gyms and Historic sites. Overall, the city of London offers a multicultural, diverse and certainly an entertaining experience.

Toronto is relatively the same as London. It has a wide variety of cuisines and eateries including Japanese, Vietnamese, Ethiopian, Colombian, Bakeries and several others. There are a lot of places to relax including cafe's, beer bars and coffee shops, including more. People also like to go to places such as farmers markets and visiting the harbour. Toronto has a lot of Diners which London does not have. Different means of public transport in Toronto which includes the metro station and light rail station. For leisure, there a lot of dog runs, gyms, spas and people like to also play hockey baseball. Overall, much like London, Toronto is multicultural, diverse and offers a great experience to anyone who visits or lives in Toronto

6. Conclusion

The purpose of this project was to explore the cities of London and Toronto and see how attractive it is to potential tourists and international students who are planning on studying in one of these cities. We explored both the cities based on their postal codes and then extrapolated the common venues present in each of the neighbourhoods and finally concluding with clustering similar neighbourhoods together.

We could see that each of the neighbourhoods in both the cities have a wide variety of experiences to offer which is unique in its own way. The cultural diversity is quite evident which also gives the feeling of a sense of inclusion.

Both London and Toronto seem to offer a vast number of things to do while studying with a lot of places to explore, beautiful landscapes and a wide variety of culture. Overall, it's up to the international students to decide which city they would most like to study in as both of these cities offers a lot of the same thing which will make them feel included, so whichever city they decide to study in they will not be disappointed. To decide which city to study in they would probably need to do research on the university itself, but as for the city that they are staying in, you can't go wrong with either of these two wonderful cities

7. References

1. https://en.wikipedia.org/wiki/List_of_areas_of_London
2. https://en.wikipedia.org/wiki/List_of_postal_codes_of_Canada:_M
3. Clustering Neighborhoods of London and Paris using Machine Learning
4. The Battle of Neighbourhood — My London's Perspective by Dayo John
5. https://labs.cognitiveclass.ai/tools/jupyterlab/lab/tree/labs/DS0701EN/DS0701EN-2-2-1-Foursquare-API-py-v2.0.ipynb?lti=true