Priscilla Chan

Capstone Project - The Battle of Neighborhoods

The Business Problem:

Section Description: Introduction where you discuss the business problem and who would be interested in this project.

The purpose of this study is to help a Chinese investor find a neighbourhood in which to open up a bubble tea shop. The investor is based in Toronto and personally enjoys bubble tea so he thought he would open up his own franchise.

Criteria:

- Few Coffee shops or other bubble tea competitors but they should be in the area
- Is near a gym or a school
- is near restaurants (and fast food)
- is near stores where there is heavier foot traffic

The Data:

Section Description: Data where you describe the data that will be used to solve the problem and the source of the data.

The data consists of a web scrapped list of neighbourhoods and their latitudes and longitudes in Canada from Wikipedia using BeautifulSoup. Specifically all of the lostal codes that start with M are the ones for the City of Toronto in Ontario. This is the link to the table on Wikipedia: https://en.wikipedia.org/wiki/List_of_postal_codes_of_Canada:_M

This dataset is combined with data on nearby venues and amenities pulled using the FourSquare API. The data from the FourSquare API will look for venues within a radius of 500 meters which is about a 5 minute walking distance which is an important thing to consider for downtown Toronto especially.

These are the Python libraries that are used.

```
#Before we get the data and start exploring it, let's download all the dependencies that we will need.

import numpy as np # library to handle data in a vectorized manner
import pandas as pd # library for data analsysis

import json # library to handle JSON files

!conda install -c conda-forge geopy --yes
from geopy.geocoders import Nominatim # convert an address into latitude and longitude values

import requests # library to handle requests
from pandas.io.json import json_normalize # tranform JSON file into a pandas dataframe

# Matplotlib and associated plotting modules
import matplotlib.cm as cm
import matplotlib.colors as colors

# import k-means from clustering stage
from sklearn.cluster import KMeans

!conda install -c conda-forge folium=0.5.0 --yes # uncomment this line if you haven't completed the Foursquare API lab
import folium # map rendering library

print('Libraries imported.')
```

This is how the neighbourhoods are webscrapped from Wikipedia:

```
#webscrape data from wikipedia
from bs4 import BeautifulSoup

#Access url and needed table
website_text = requests.get('https://en.wikipedia.org/wiki/List_of_postal_codes_of_Canada:_M').text
soup = BeautifulSoup(website_text,'xml')
table = soup.find('table',{'class':'wikitable sortable'})

#Putting data into dataframe
table_rows = table.find_all('tr')
data = []
for row in table_rows:
    data.append([t.text.strip() for t in row.find_all('td')])
df = pd.DataFrame(data, columns=['PostalCode', 'Borough', 'Neighborhood'])

#Delete empty rows
df = df[~df['Borough'].isnull()]

#Dropping rows without assigned Boroughs
df.drop(df[df.Borough == 'Not assigned'].index, inplace=True)
df.reset_index(drop=True, inplace=True)
df = df.groupby(['PostalCode','Borough'])['Neighborhood'].apply(lambda x: ','.join(x)).reset_index()

#Set unassigned neighbourhoods to be the same as Borough
df['Neighborhood'].replace('Not assigned',df['Borough'],inplace=True)

df.head()
```

2]:

	PostalCode	Borough	Neighborhood
0	M1B	Scarborough	Rouge,Malvern
1	M1C	Scarborough	Highland Creek,Rouge Hill,Port Union
2	M1E	Scarborough	Guildwood,Morningside,West Hill
3	M1G	Scarborough	Woburn
4	M1H	Scarborough	Cedarbrae

```
df.shape
```

3]: (103, 3)

The latitude and longitudes are obtained from this file 'Toronto_long_lat_data.csv' at the url http://cocl.us/Geospatial_data .

They are then added to the dataframe of neighbourhoods.

```
#Using the CSV file from http://cocl.us/Geospatial_data to get coordinate data
!wget -q -O 'Toronto_long_lat_data.csv'  http://cocl.us/Geospatial_data
df_lon_lat = pd.read_csv('Toronto_long_lat_data.csv')
df_lon_lat.head()
```

]:

	Postal Code	Latitude	Longitude
0	M1B	43.806686	-79.194353
1	M1C	43.784535	-79.160497
2	M1E	43.763573	-79.188711
3	M1G	43.770992	-79.216917
4	M1H	43.773136	-79.239476

```
#Changing df column name to merge
df_lon_lat.columns=['PostalCode','Latitude','Longitude']

#Merging dfs
dfm = pd.merge(df,
                df_lon_lat[['PostalCode','Latitude', 'Longitude']],
                on='PostalCode')
dfm.head()
```

]:

	PostalCode	Borough	Neighborhood	Latitude	Longitude
0	M1B	Scarborough	Rouge,Malvern	43.806686	-79.194353
1	M1C	Scarborough	Highland Creek,Rouge Hill,Port Union	43.784535	-79.160497
2	M1E	Scarborough	Guildwood,Morningside,West Hill	43.763573	-79.188711
3	M1G	Scarborough	Woburn	43.770992	-79.216917
4	M1H	Scarborough	Cedarbrae	43.773136	-79.239476

```
dfm.shape
```

]: (103, 5)

Then the boroughs are filtered for only those containing "Toronto" in the name. The table continues to row index 38.

```
dfm.drop(dfm[dfm.Borough.str.contains("Toronto")==False].index, inplace=True)
dfm = dfm.reset_index(drop=True)
dfm
```

	PostalCode	Borough	Neighborhood	Latitude	Longitude
0	M4E	East Toronto	The Beaches	43.676357	-79.293031
1	M4K	East Toronto	The Danforth West,Riverdale	43.679557	-79.352188
2	M4L	East Toronto	The Beaches West,India Bazaar	43.668999	-79.315572
3	M4M	East Toronto	Studio District	43.659526	-79.340923
4	M4N	Central Toronto	Lawrence Park	43.728020	-79.388790

We can create a map for visual purposes. The starting point of the map is the spatial center of all of the borough latitudes and longitudes.

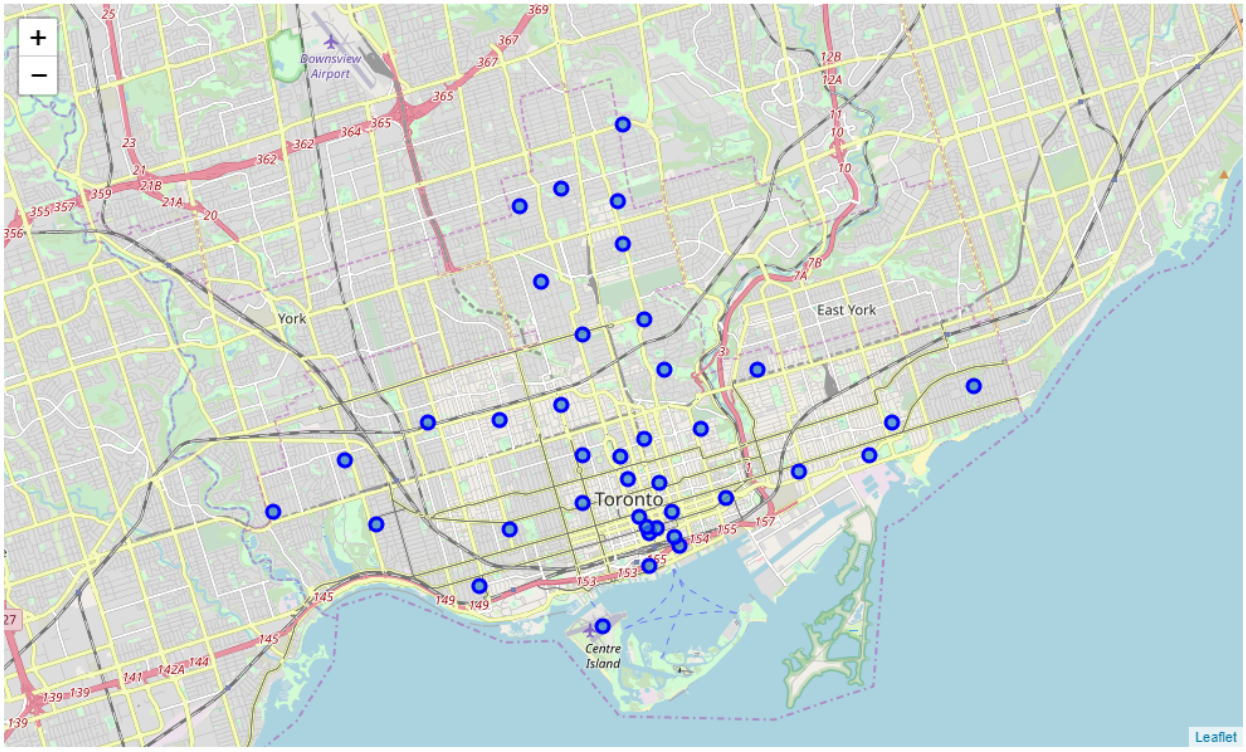
```
latitude = (dfm['Latitude'].max() + dfm['Latitude'].min())/2
longitude = (dfm['Longitude'].max() + dfm['Longitude'].min())/2
latitude, longitude
```

```
(43.6784836, -79.38874055)
```

```
# create map of Toronto using latitude and longitude values
map_toronto = folium.Map(location=[latitude, longitude], zoom_start=10)

# add markers to map
for lat, lng, borough, neighborhood in zip(dfm['Latitude'], dfm['Longitude'], dfm['Bor
ough'], dfm['Neighborhood']):
    label = '{}, {}'.format(neighborhood, borough)
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius=5,
        popup=label,
        color='blue',
        fill=True,
        fill_color='#3186cc',
        fill_opacity=0.7,
        parse_html=False).add_to(map_toronto)

map_toronto
```



Then we get, store, and print the latitude, longitude and name of each neighbourhood.

```
dfm.loc[0, 'Neighborhood']
```

```
'The Beaches'
```

```
neighborhood_latitude = dfm.loc[0, 'Latitude'] # neighborhood latitude value
neighborhood_longitude = dfm.loc[0, 'Longitude'] # neighborhood longitude value

neighborhood_name = dfm.loc[0, 'Neighborhood'] # neighborhood name

print('Latitude and longitude values of {} are {}, {}.'.format(neighborhood_name,
                                                               neighborhood_latitude,
                                                               neighborhood_longitude))
```

```
Latitude and longitude values of The Beaches are 43.67635739999999, -79.2930312.
```

After defining our client id, client secret, and the version of the Foursquare API (VERSION = '20180605') we can call the api for the top 100 venues in a 500 meter radius from the neighbourhood at index 0.

```
LIMIT = 100 # limit of number of venues returned by Foursquare API
radius = 500 # define radius
url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll=
{},{}&radius={}&limit={}'.format(
    CLIENT_ID,
    CLIENT_SECRET,
    VERSION,
    neighborhood_latitude,
    neighborhood_longitude,
    radius,
    LIMIT)
url
```

We get the results as a json file.

```
results = requests.get(url).json()
results
```

From these results we pull the venue categories as well and put everything into a pandas dataframe.

```
# function that extracts the category of the venue
def get_category_type(row):
    try:
        categories_list = row['categories']
    except:
        categories_list = row['venue.categories']

    if len(categories_list) == 0:
        return None
    else:
        return categories_list[0]['name']
```

```
venues = results['response']['groups'][0]['items']

nearby_venues = json_normalize(venues) # flatten JSON

# filter columns
filtered_columns = ['venue.name', 'venue.categories', 'venue.location.lat', 'venue.locatio
n.lng']
nearby_venues =nearby_venues.loc[:, filtered_columns]

# filter the category for each row
nearby_venues['venue.categories'] = nearby_venues.apply(get_category_type, axis=1)

# clean columns
nearby_venues.columns = [col.split(".")[-1] for col in nearby_venues.columns]

nearby_venues.head()
```

	name	categories	lat	lng
0	Glen Manor Ravine	Trail	43.676821	-79.293942
1	The Big Carrot Natural Food Market	Health Food Store	43.678879	-79.297734
2	Grover Pub and Grub	Pub	43.679181	-79.297215
3	Upper Beaches	Neighborhood	43.680563	-79.292869

This is then done to all of the neighbourhoods.

```
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)
```

This is all of the venues in Toronto of the relevant boroughs.

```
toronto_venues = getNearbyVenues(names=dfm['Neighborhood'],
                                 latitudes=dfm['Latitude'],
                                 longitudes=dfm['Longitude']
                                )
```

Checking the dataframe.

```
print(toronto_venues.shape)
toronto_venues.head()
```

```
(1714, 7)
```

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	The Beaches	43.676357	-79.293031	Glen Manor Ravine	43.676821	-79.293942	Trail
1	The Beaches	43.676357	-79.293031	The Big Carrot Natural Food Market	43.678879	-79.297734	Health Food Store
2	The Beaches	43.676357	-79.293031	Grover Pub and Grub	43.679181	-79.297215	Pub
3	The Beaches	43.676357	-79.293031	Upper Beaches	43.680563	-79.292869	Neighborhood
4	The Danforth West,Riverdale	43.679557	-79.352188	Pantheon	43.677621	-79.351434	Greek Restaurant

Methodology:

Section description: Methodology section which represents the main component of the report where you discuss and describe any exploratory data analysis that you did, any inferential statistical testing that you performed, and what machine learnings were used and why.

As mentioned in the Data section, the data of venues and boroughs of toronto are combined into one data set. From here, the combined dataset is refined to just boroughs containing "Toronto " in the name and the neighbourhoods will be further sorted to ones that only match the required criteria and then ranked.

The criteria is set to filter for high traffic areas with interests that are relevant to bubble tea drinkers. For instance, bubble tea is usually a go to hangout spot for students, and a treat for people who have just gone to the gym. Additionally bubble tea is often regarded as a dessert stop for post-meal consumers or a good drink alternative to those getting fast food (as opposed to a soft drink).

The boroughs are also filtered to just those containing the word Toronto to narrow it down to mostly the core downtown areas for simplicity, and then clusters are used to sort the neighbourhoods based on which relevant venue groups are the most frequent to help us narrow down a particular neighbourhood to suggest to open a bubble tea shop in.

We start by checking how many venues are returned for each neighbourhood. Only the first 5 rows are listed below.

```
toronto_venues.groupby('Neighborhood').count()
```

	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
Neighborhood						
Adelaide,King,Richmond	100	100	100	100	100	100
Berczy Park	56	56	56	56	56	56
Brockton,Exhibition Place,Parkdale Village	25	25	25	25	25	25
Business Reply Mail Processing Centre 969 Eastern	18	18	18	18	18	18

```
toronto_venues.groupby('Neighborhood')
```

```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x7f210a6f9f28>
```

```
#Find how many unique categories can be curated from all the returned venues
print('There are {} uniques categories.'.format(len(toronto_venues['Venue Category'].unique())))
```

```
There are 233 uniques categories.
```

Each neighbourhood is analyzed for now many counts of each category there are.

```
# 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_onehot.head()
```

	Yoga Studio	Afghan Restaurant	Airport	Airport Food Court	Airport Gate	Airport Lounge	Airport Service	Airport Terminal	American Restaurant	Antique Shop	...	Toy / Game Store
0	0	0	0	0	0	0	0	0	0	0	...	0
1	0	0	0	0	0	0	0	0	0	0	...	0
2	0	0	0	0	0	0	0	0	0	0	...	0
3	0	0	0	0	0	0	0	0	0	0	...	0
4	0	0	0	0	0	0	0	0	0	0	...	0

5 rows × 233 columns

The dataframe is checked and the dataframe is changed into mean frequency of occurrence for analysis. The toronto_grouped table goes to index 38.

```
#examine the new dataframe size
toronto_onehot.shape
```

```
(1714, 233)
```

```
#group rows by neighborhood and take the mean of the frequency of occurrence of each category
toronto_grouped = toronto_onehot.groupby('Neighborhood').mean().reset_index()
toronto_grouped
```

	Neighborhood	Yoga Studio	Afghan Restaurant	Airport	Airport Food Court	Airport Gate	Airport Lounge	Airport Service	Airport Terminal	Amer Resta
0	Adelaide,King,Richmond	0.000000	0.000000	0.0000	0.0000	0.0000	0.000	0.0000	0.0000	0.020
1	Berczy Park	0.000000	0.000000	0.0000	0.0000	0.0000	0.000	0.0000	0.0000	0.000
2	Brockton,Exhibition Place,Parkdale Village	0.040000	0.000000	0.0000	0.0000	0.0000	0.000	0.0000	0.0000	0.000
3	Business Reply Mail Processing Centre 969 Eastern	0.000000	0.000000	0.0000	0.0000	0.0000	0.000	0.0000	0.0000	0.000
4	CN Tower,Bathurst Quay,Island airport,Harbourf...	0.000000	0.000000	0.0625	0.0625	0.0625	0.125	0.1875	0.0625	0.000

```
#confirm new size
toronto_grouped.shape
```

```
(39, 233)
```

Now these categories are filtered and grouped for relevancy to the business problem at hand. Anything that contains "Gym" or "Studio" is a gym item, anything that contains "Restaurant" is a food location and anything that contains "Shop" or "Store" is a shopping location. The last two categories are "Bubble Tea" and it's competing good "Coffee" shops and "Café"s. There are other physical activity centres and locations that serve food where the category did not contain the terms "Gym", "Studio", and "Restaurant" respectively but upon inspection of the list of all present category names, these rules applied to a majority of the data and should give a pretty clear idea of the overall mean frequencies.

The category name filtering, each filter is put into it's own dataframe.

```
to_regroup_gym1= pd.DataFrame(toronto_grouped.filter(like='Gym'))
to_regroup_gym2= pd.DataFrame(toronto_grouped.filter(like='Studio'))
to_regroup_food = pd.DataFrame(toronto_grouped.filter(like='Restaurant'))
to_regroup_store = pd.DataFrame(toronto_grouped.filter(like='Store'))
to_regroup_shop= pd.DataFrame(toronto_grouped.filter(like='Shop'))
to_regroup_bbt = pd.DataFrame(toronto_grouped.filter(like='Bubble Tea'))
to_regroup_cafe = pd.DataFrame(toronto_grouped.filter(like='Café'))
to_regroup_coffee = pd.DataFrame(toronto_grouped.filter(like='Coffee'))
```

The gym, shopping and coffee joint dataframes are joined together since these three required 2 seperate naming filters. A new dataframe is also created.

```
to_regroup_gym = pd.concat([to_regroup_gym1, to_regroup_gym2],ignore_index=True, sort=False)
to_regroup_shopping = pd.concat([to_regroup_store, to_regroup_shop],ignore_index=True, sort=False)
to_regroup_coffee = pd.concat([to_regroup_cafe, to_regroup_coffee],ignore_index=True, sort=False)

headings = ['Neighborhood', 'Gym', 'Food', 'Shopping', 'BBT', 'Coffees']
to_regroup = pd.DataFrame(columns = headings) #creates a new dataframe
```

New columns in each of the grouped dataframes for venue category groupings are added and the sum of the mean frequencies for the individual specific categories (ie. "Afghan Restaurant" and "Chinese Restaurant") are added together to find a group venue category (ie. "Food") mean frequency.

```
to_regroup_gym.loc[:,'Gyms'] = to_regroup_gym.sum(axis=1)
to_regroup_food.loc[:,'Foods'] = to_regroup_food.sum(axis=1)
to_regroup_shopping.loc[:,'BuyAllThings'] = to_regroup_shopping.sum(axis=1)
to_regroup_bbt.loc[:,'BBT'] = to_regroup_bbt.sum(axis=1)
to_regroup_coffee.loc[:,'Caffine'] = to_regroup_coffee.sum(axis=1)
```

```
to_regroup['Neighborhood'] = toronto_grouped['Neighborhood']
to_regroup['Gym'] = to_regroup_gym['Gyms']
to_regroup['Food'] = to_regroup_food['Foods']
to_regroup['Shopping'] = to_regroup_shopping['BuyAllThings']
to_regroup['BBT'] = to_regroup_bbt['BBT']
to_regroup['Coffees'] = to_regroup_coffee['Caffine']

to_regroup
```

	Neighborhood	Gym	Food	Shopping	BBT	Coffees
0	Adelaide,King,Richmond	0.030000	1.860000	0.060000	0.000000	0.040000
1	Berczy Park	0.000000	1.368421	0.035088	0.000000	0.035088
2	Brockton,Exhibition Place,Parkdale Village	0.083333	0.750000	0.166667	0.000000	0.125000
3	Business Reply Mail Processing Centre 969 Eastern	0.000000	0.750000	0.000000	0.000000	0.000000
4	CN Tower,Bathurst Quay,Island airport,Harbourf...	0.000000	0.000000	0.000000	0.000000	0.000000

With this, the relevant category groups can be ranked by frequency by neighbourhood.

```
num_top_venues = 5

for hood in toronto_grouped['Neighborhood']:
    print("----"+hood+"----")
    temp = to_regroup[to_regroup['Neighborhood'] == hood].T.reset_index()
    temp.columns = ['venue','freq']
    temp = temp.iloc[1:]
    temp['freq'] = temp['freq'].astype(float)
    temp = temp.round({'freq': 2})
    print(temp.sort_values('freq', ascending=False).reset_index(drop=True).head(num_top_venues))
    print('\n')
```

```
----Adelaide,King,Richmond----
      venue  freq
0      Food  1.86
1  Shopping  0.06
2   Coffees  0.04
3       Gym  0.03
4       BBT  0.00


----Berczy Park----
      venue  freq
0      Food  1.37
1  Shopping  0.04
2   Coffees  0.04
3       Gym  0.00
4       BBT  0.00


----Brockton,Exhibition Place,Parkdale Village----
      venue  freq
0      Food  0.75
1  Shopping  0.17
2   Coffees  0.12
3       Gym  0.08
4       BBT  0.00


----Business Reply Mail Processing Centre 969 Eastern----
      venue  freq
0      Food  0.75
1       Gym  0.00
2  Shopping  0.00
3       BBT  0.00
4   Coffees  0.00


----CN Tower,Bathurst Quay,Island airport,Harbourfront West,King and Spadina,Railway Lands,South Niagara--
--
      venue  freq
0       Gym   0.0
1      Food   0.0
2  Shopping   0.0
3       BBT   0.0
4   Coffees   0.0
```

Specifically, we sort them in descending order into a pandas dataframe.

```python
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]
```

```
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'] = to_regroup['Neighborhood']

for ind in np.arange(toronto_grouped.shape[0]):
    neighborhoods_venues_sorted.iloc[ind, 1:] = return_most_common_venues(to_regroup.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
0	Adelaide,King,Richmond	Food	Shopping	Coffees	Gym	BBT
1	Berczy Park	Food	Coffees	Shopping	BBT	Gym
2	Brockton,Exhibition Place,Parkdale Village	Food	Shopping	Coffees	Gym	BBT
3	Business Reply Mail Processing Centre 969 Eastern	Food	Coffees	BBT	Shopping	Gym
4	CN Tower,Bathurst Quay,Island airport,Harbourf...	Coffees	BBT	Shopping	Food	Gym

K-means clustering is then used to group the neighbourhoods into 5 clusters (a previously found optimal number of clusters for these same boroughs and neighbourhoods of Toronto from a previous assignment).

```
# set number of clusters
kclusters = 5

toronto_grouped_clustering = to_regroup.drop('Neighborhood', 1)

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

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

```
array([4, 3, 2, 2, 0, 1, 4, 4, 2, 4], dtype=int32)
```

A dataframe is created to show all of the information.

```
# add clustering labels
neighborhoods_venues_sorted['Cluster Labels'] = kmeans.labels_

toronto_merged = dfm

# merge toronto_grouped with toronto_data to add latitude/longitude for each neighborhood
toronto_merged = toronto_merged.join(neighborhoods_venues_sorted.set_index('Neighborhood'), on='Neighborhood')

toronto_merged.head() # check the last columns!
```

PostalCode	Borough	Neighborhood	Latitude	Longitude	0	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	Cluster Labels
M4E	East Toronto	The Beaches	43.676357	-79.293031	0	Shopping	Coffees	BBT	Food	Gym	0
M4K	East Toronto	The Danforth West,Riverdale	43.679557	-79.352188	4	Food	BBT	Shopping	Coffees	Gym	4
M4L	East Toronto	The Beaches West,India Bazaar	43.668999	-79.315572	3	Food	Shopping	Gym	Coffees	BBT	3
M4M	East Toronto	Studio District	43.659526	-79.340923	3	Food	Shopping	Coffees	Gym	BBT	3
M4N	Central Toronto	Lawrence Park	43.728020	-79.388790	0	Coffees	BBT	Shopping	Food	Gym	0

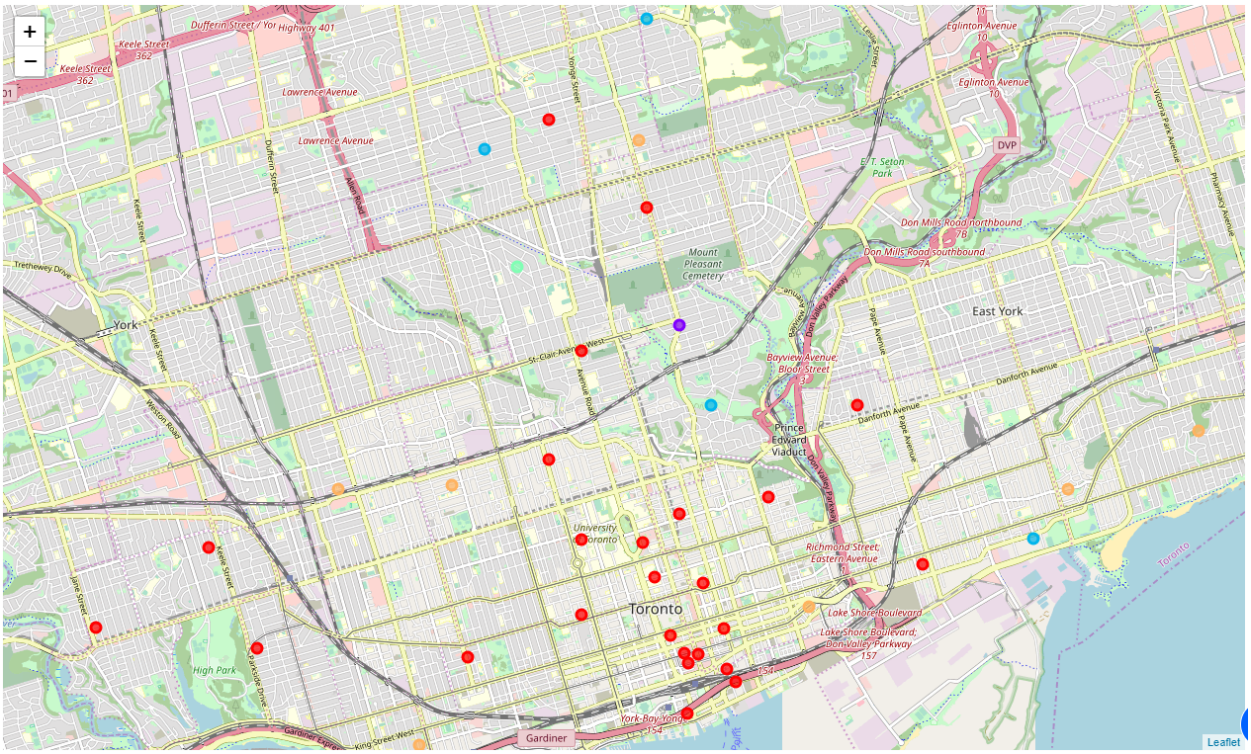
Now a map can be generated to visualize the clusters.

```
# create map
map_clusters = folium.Map(location=[latitude, longitude], zoom_start=11)

# set color scheme for the clusters
x = np.arange(kclusters)
ys = [i + x + (i*x)**2 for i in range(kclusters)]
colors_array = cm.rainbow(np.linspace(0, 1, len(ys)))
rainbow = [colors.rgb2hex(i) for i in colors_array]

# add markers to the map
markers_colors = []
for lat, lon, poi, cluster in zip(toronto_merged['Latitude'], toronto_merged['Longitude'], toronto_merged['Neighborhood'], toronto_merged['Cluster Labels']):
    label = folium.Popup(str(poi) + ' Cluster ' + str(cluster), parse_html=True)
    folium.CircleMarker(
        [lat, lon],
        radius=5,
        popup=label,
        color=rainbow[cluster-1],
        fill=True,
        fill_color=rainbow[cluster-1],
        fill_opacity=0.7).add_to(map_clusters)

map_clusters
```



The neighbourhoods are separated into dataframes for each cluster.

Cluster 1:

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

	Borough	0	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	Cluster Labels
0	East Toronto	0	Shopping	Coffees	BBT	Food	Gym	0
4	Central Toronto	0	Coffees	BBT	Shopping	Food	Gym	0
8	Central Toronto	0	Coffees	BBT	Shopping	Food	Gym	0
10	Downtown Toronto	0	Coffees	BBT	Shopping	Food	Gym	0
22	Central Toronto	0	Coffees	BBT	Shopping	Food	Gym	0
27	Downtown Toronto	0	Coffees	BBT	Shopping	Food	Gym	0
31	West Toronto	0	Food	Shopping	Coffees	Gym	BBT	0

Cluster 2:

```
toronto_merged.loc[toronto_merged['Cluster Labels'] == 1, toronto_merged.columns[[1] + list(range(5, toron
to_merged.shape[1]))]]
```

	Borough	0	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	Cluster Labels
7	Central Toronto	1	Food	Coffees	Shopping	Gym	BBT	1
11	Downtown Toronto	1	Food	Shopping	Coffees	BBT	Gym	1
20	Downtown Toronto	1	Food	Coffees	Gym	BBT	Shopping	1
21	Downtown Toronto	1	Food	Coffees	Gym	Shopping	BBT	1
23	Central Toronto	1	Food	Shopping	Coffees	BBT	Gym	1
25	Downtown Toronto	1	Food	Coffees	Shopping	Gym	BBT	1
29	Downtown Toronto	1	Food	Coffees	Gym	Shopping	BBT	1
32	West Toronto	1	Food	Shopping	Coffees	BBT	Gym	1
35	West Toronto	1	Food	Coffees	BBT	Shopping	Gym	1
36	West Toronto	1	Food	Coffees	Shopping	Gym	BBT	1

Cluster 3:

```
toronto_merged.loc[toronto_merged['Cluster Labels'] == 2, toronto_merged.columns[[1] + list(range(5, toron
to_merged.shape[1]))]]
```

	Borough	0	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	Cluster Labels
5	Central Toronto	2	Food	Shopping	Gym	Coffees	BBT	2
13	Downtown Toronto	2	Food	Shopping	Coffees	Gym	BBT	2
19	Downtown Toronto	2	Food	BBT	Coffees	Shopping	Gym	2
24	Central Toronto	2	Food	Coffees	Shopping	BBT	Gym	2
30	Downtown Toronto	2	Food	Shopping	Coffees	BBT	Gym	2
33	West Toronto	2	Food	Shopping	Coffees	Gym	BBT	2
38	East Toronto	2	Food	Coffees	BBT	Shopping	Gym	2

Cluster 4:

```
toronto_merged.loc[toronto_merged['Cluster Labels'] == 3, toronto_merged.columns[[1] + list(range(5, toron
to_merged.shape[1]))]]
```

	Borough	0	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	Cluster Labels
2	East Toronto	3	Food	Shopping	Gym	Coffees	BBT	3
3	East Toronto	3	Food	Shopping	Coffees	Gym	BBT	3
6	Central Toronto	3	Food	Shopping	Coffees	Gym	BBT	3
9	Central Toronto	3	Food	Shopping	Coffees	BBT	Gym	3
14	Downtown Toronto	3	Food	BBT	Shopping	Coffees	Gym	3
15	Downtown Toronto	3	Food	Shopping	Coffees	Gym	BBT	3
16	Downtown Toronto	3	Food	Coffees	Shopping	BBT	Gym	3
28	Downtown Toronto	3	Food	Coffees	Shopping	Gym	BBT	3
37	Downtown Toronto	3	Food	Shopping	Gym	Coffees	BBT	3

and Cluster 5:

```
toronto_merged.loc[toronto_merged['Cluster Labels'] == 4, toronto_merged.columns[[1] + list(range(5, toron
to_merged.shape[1]))]]
```

	Borough	0	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	Cluster Labels
1	East Toronto	4	Food	BBT	Shopping	Coffees	Gym	4
12	Downtown Toronto	4	Food	BBT	Shopping	Coffees	Gym	4
17	Downtown Toronto	4	Food	BBT	Shopping	Coffees	Gym	4
18	Downtown Toronto	4	Food	Shopping	Coffees	Gym	BBT	4
26	Downtown Toronto	4	Food	BBT	Coffees	Shopping	Gym	4
34	West Toronto	4	Food	Shopping	Coffees	BBT	Gym	4

Results:

Section Description: Results section where you discuss the results.

The results of the analysis show that the 39 boroughs containing "Toronto" in their name were seperated into 5 clusters based on which of the groupings of venues of interest were the most to least frequent in that area.

A closer look at the results of the clustering of the neighbourhoods is broken down and analysed in the 'Discussions' section below.

Discussion:

Section Description: Discussion section where you discuss any observations you noted and any recommendations you can make based on the results.

We can see that the first cluster with the exception of East Toronto and West Toronto (index 0 and 31 respectively) that the most common venues in order of frequency are all Coffee places, BBT shops, Shops and store, Food locations and then Gym locations. The second cluster is the largest cluster and like the rest of the clusters has a large concentration of restaurants. There are also a lot of coffee shops in those areas which is no surprise since half of

the boroughs are Downtown Toronto. This may be a catch-all cluster that happens to have more exercise related facilities in the area considering how scattered the last 3 columns of 'common venues' are.

The last three clusters also all have food locations as their most common venue in the area. Again, as 'Downtown Toronto' seems to be the majority of these clusters that would make sense. Cluster three and four seem pretty similar since in addition to food they also have many shopping locations and coffee spots in that order. Cluster three however, has more bubble tea locations than cluster four, while the latter cluster is more popular for gyms than bubble tea shops. Cluster five has the highest frequency for bubble tea shops , the least gyms, and shopping and coffee frequencies fall in between the two.

Based on these observations, I would recommend that the investor look into boroughs of Downtown Toronto or Central Toronto in cluster 3 and 4. Specifically, the neighbourhoods where food and shopping are the top most common venues (in that order respectively), since this is where the most relevant foot traffic will be. Ideally Coffee locations and Bubble tea shops will be next since coffee and bubble tea are roughly substitute goods. Bubble tea locations also seem to like to cluster together to give consumers choice more often than opening in locations where there do not exist many others so this is also something that may be important to consider, especially in an area of high population density like Toronto (compared to more rural areas).

Conclusion:

Section Description: Conclusion section where you conclude the report.

In conclusion, we saw results of clustering locations in Toronto based on how many restaurants, gyms, bubble tea shops, coffee shops and stores there are in those areas. Based on these results, we were able to distinguish what made each cluster different from the others and we also created a map to visualize where these neighbourhoods are geographically.

Upon analyzing these resulting clusters we can see that based on our criteria for the business problem outlined in the introduction, that there are certain neighbourhoods in certain clusters that fit the criteria more appropriately.

Happy bubble tea drinking!