

Battle of the Neighborhoods in Toronto

Introduction

In this capstone project I decided to utilize the Foursquare data to study the market for restaurants and service oriented small businesses in the Downtown Toronto area. For the analyses of this study, K-means clusterin is used the cluster the neighborhoods and amenities into clusters. The study explores the neighborhoods that currently have a higher concentration of restaurant businesses, and areas where there are opportunities for opening service oriented businesses.

Problem Description and Background

An important aspect of determining the potential for new businesses in an area before investing into the place is to do market research. Market research helps businesses determine the potential strengths, opportunities, potential competition, and the likelihood of success when opening a new business when entering a new market. Until now most market research was conducted through data collected from questionnaires and surveys. These data were then analyzed. However the emergence of data science and information technology, and the ability to access platforms such as foursquare that make such data available for research has helped in evolving this process of exploring a new market or area, a more scientific endeavor.

In week 3 we learned in this class, how to use and access Foursquare data. As part of the week 3 assignment, we also segmented and clustered Toronto neighborhoods. So, it was determined that applying the foursquare data to compare the potential strengths, opportunities, and competition for opening new restaurants or small service oriented businesses in Toronto will be an interesting study. Therefore, I decided to pursue my analysis to examine the potential for opening restaurants and service oriented business by comparing different Toronto neighborhoods.

Data Description

The data for this study was obtained from two different sources:

- Postal codes and neighborhood information for Toronto was obtained from the following wikipedia page (https://en.wikipedia.org/wiki/List_of_postal_codes_of_Canada:_M (https://en.wikipedia.org/wiki/List_of_postal_codes_of_Canada:_M)). The data for Toronto was then scraped and cleaned as described below and a data frame include postal code, neighborhood and borough information was then constructed.
- Information on services and amenities in different locations were obtained from the Foursquare. As suggested in week 3, a free developer account was first setup with foursquare.com. Client Id, client secret information were obtained. Then using a version number the data for downtown Toronto was then extracted and merged with the table created above. These processes are explained below.

First the tools needed for these analysis were imported and installed as shown below:

```
In [3]: import numpy as np # library to handle data as arrays and vectors

import pandas as pd # Tools for handling data structures
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)

import json # library to handle JSON files

!conda install -c conda-forge geopy --yes
from geopy.geocoders import Nominatim # Conversion of an address to latitudes and long
itudes

import requests # library to handle requests
from pandas.io.json import json_normalize # transforming JSON to a pandas dataframe

# Importing 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
import folium # map rendering library

print('Libraries imported.')
```

Solving environment: done

Package Plan

environment location: /opt/conda/envs/Python36

added / updated specs:

- geopy

The following packages will be downloaded:

package	build		
certifi-2019.11.28	py36_0	149 KB	conda-forge
geographiclib-1.50	py_0	34 KB	conda-forge
openssl-1.1.1d	h516909a_0	2.1 MB	conda-forge
ca-certificates-2019.11.28	hecc5488_0	145 KB	conda-forge
geopy-1.20.0	py_0	57 KB	conda-forge
Total:		2.5 MB	

The following NEW packages will be INSTALLED:

geographiclib:	1.50-py_0	conda-forge
geopy:	1.20.0-py_0	conda-forge

The following packages will be UPDATED:

ca-certificates:	2019.11.27-0	--> 2019.11.28-hecc5488_0	conda-f
orge			
certifi:	2019.11.28-py36_0	--> 2019.11.28-py36_0	conda-f
orge			

The following packages will be DOWNGRADED:

openssl:	1.1.1d-h7b6447c_3	--> 1.1.1d-h516909a_0	conda-f
orge			

Downloading and Extracting Packages

certifi-2019.11.28	149 KB	#####	100%
geographiclib-1.50	34 KB	#####	100%
openssl-1.1.1d	2.1 MB	#####	100%
ca-certificates-2019	145 KB	#####	100%
geopy-1.20.0	57 KB	#####	100%

Preparing transaction: done

Verifying transaction: done

Executing transaction: done

Solving environment: done

Package Plan

environment location: /opt/conda/envs/Python36

added / updated specs:

- folium=0.5.0

The following packages will be downloaded:

package	build		
branca-0.3.1	py_0	25 KB	conda-forge
altair-4.0.0	py_0	606 KB	conda-forge
folium-0.5.0	py_0	45 KB	conda-forge

In []:

Web scraping the table with postal codes from the Wikipedia page

In [4]: `from bs4 import BeautifulSoup` # Next BeautifulSoup is imported for scraping the table of the wikipedia page

In [5]: `# getting data from internet`
`link='https://en.wikipedia.org/wiki/List_of_postal_codes_of_Canada:_M'`
`link_page= requests.get(link).text`

`# using beautiful soup to parse the HTML/XML codes.`
`soup = BeautifulSoup(link_page, 'lxml')`

In [6]: `table = soup.find('table', {'class': 'wikitable sortable'}).tbody` #Source info on the wikipedia page reveals that the table is classified as wiki sortable

In [7]: `rows = table.find_all('tr')` # Page source also reveals that tr indicates tags for rows and th represents tags for columns

`# taking out the spaces in between`
`columns = ['PostalCode', 'Borough', 'Neighborhood']`
`# [v.text.replace('\n', '') for v in rows[0].find_all('th')]`
`print(columns)`

`['PostalCode', 'Borough', 'Neighborhood']`

Assigning columns to the data frames

In [8]: `df = pd.DataFrame(columns=columns)`
`df`

Out[8]:

PostalCode	Borough	Neighborhood
------------	---------	--------------

Preparing the table in csv format

In [9]: `from pandas import DataFrame`

`for i in range(1, len(rows)):`
`tds = rows[i].find_all('td')`

`if len(tds) == 3:`
`values = [tds[0].text, tds[1].text, tds[2].text.replace('\n', '')]`
`else:`
`values = [td.text.replace('\n', '') for td in tds]`

`# populating the dataframe with the data that we just extracted from the HTML table`
`df = df.append(pd.Series(values, index=columns), ignore_index=True)`

`export_csv= df.to_csv(r'C:\Users\fhce\Desktop\export_dataframe.csv', index = None, header=True)`

```
In [10]: print(df)
```

	PostalCode	Borough \
0	M1A	Not assigned
1	M2A	Not assigned
2	M3A	North York
3	M4A	North York
4	M5A	Downtown Toronto
5	M6A	North York
6	M6A	North York
7	M7A	Downtown Toronto
8	M8A	Not assigned
9	M9A	Queen's Park
10	M1B	Scarborough
11	M1B	Scarborough
12	M2B	Not assigned
13	M3B	North York
14	M4B	East York
15	M4B	East York
16	M5B	Downtown Toronto
17	M5B	Downtown Toronto
18	M6B	North York
19	M7B	Not assigned
20	M8B	Not assigned
21	M9B	Etobicoke
22	M9B	Etobicoke
23	M9B	Etobicoke
24	M9B	Etobicoke
25	M9B	Etobicoke
26	M1C	Scarborough
27	M1C	Scarborough
28	M1C	Scarborough
29	M2C	Not assigned
30	M3C	North York
31	M3C	North York
32	M4C	East York
33	M5C	Downtown Toronto
34	M6C	York
35	M7C	Not assigned
36	M8C	Not assigned
37	M9C	Etobicoke
38	M9C	Etobicoke
39	M9C	Etobicoke
40	M9C	Etobicoke
41	M1E	Scarborough
42	M1E	Scarborough
43	M1E	Scarborough
44	M2E	Not assigned
45	M3E	Not assigned
46	M4E	East Toronto
47	M5E	Downtown Toronto
48	M6E	York
49	M7E	Not assigned
50	M8E	Not assigned
51	M9E	Not assigned
52	M1G	Scarborough
53	M2G	Not assigned
54	M3G	Not assigned
55	M4G	East York
56	M5G	Downtown Toronto
57	M6G	Downtown Toronto
58	M7G	Not assigned
59	M8G	Not assigned
60	M9G	Not assigned
61	M1H	Scarborough
62	M2H	North York
63	M3H	North York
64	M3H	North York

Next the not assigned cells are removed

```
In [11]: df.head()
```

Out [11]:

	PostalCode	Borough	Neighborhood
0	M1A	Not assigned	Not assigned
1	M2A	Not assigned	Not assigned
2	M3A	North York	Parkwoods
3	M4A	North York	Victoria Village
4	M5A	Downtown Toronto	Harbourfront

```
In [12]: data = df[df.Borough != 'Not assigned'] # Removing the not assigned rows
data.head()
```

Out [12]:

	PostalCode	Borough	Neighborhood
2	M3A	North York	Parkwoods
3	M4A	North York	Victoria Village
4	M5A	Downtown Toronto	Harbourfront
5	M6A	North York	Lawrence Heights
6	M6A	North York	Lawrence Manor

Integrating the Neighborhoods with duplicate postal codes

```
In [13]: data = data.groupby(['PostalCode', 'Borough'])['Neighborhood'].apply(lambda x: ', '.join(x)).reset_index()
data.head()
```

Out [13]:

	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

Number of rows in the cleaned table

```
In [14]: data.shape[0]
```

Out [14]: 103

Obtaining the Latitudes and Longitudes

```
In [15]: geocoder = pd.read_csv("https://cocl.us/Geospatial_data")

# Both tables should have the same column name for postal codes before the two tables
are merged

geocoder.rename(columns={'Postal Code': 'PostalCode'}, inplace=True)
```

```
In [47]: geocoder.head()
```

Out [47]:

	PostalCode	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

```
In [16]: geo = pd.merge(data, geocoder, on='PostalCode') #Merging the two tables by postal code
```

```
In [17]: geo.head()
```

Out [17]:

	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

```
In [18]: geo.tail()
```

Out [18]:

	PostalCode	Borough	Neighborhood	Latitude	Longitude
98	M9N	York	Weston	43.706876	-79.518188
99	M9P	Etobicoke	Westmount	43.696319	-79.532242
100	M9R	Etobicoke	Kingsview Village, Martin Grove Gardens, Richv...	43.688905	-79.554724
101	M9V	Etobicoke	Albion Gardens, Beaumont Heights, Humbergate, ...	43.739416	-79.588437
102	M9W	Etobicoke	Northwest	43.706748	-79.594054

Getting latitudes and longitudes for Toronto

```
In [19]: address = 'Toronto, ON'

geolocator = Nominatim(user_agent="to_explorer")
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude
print('The geograpical coordinates of Toronto are {}, {}'.format(latitude, longitud
e))
```

The geograpical coordinates of Toronto are 43.653963, -79.387207.

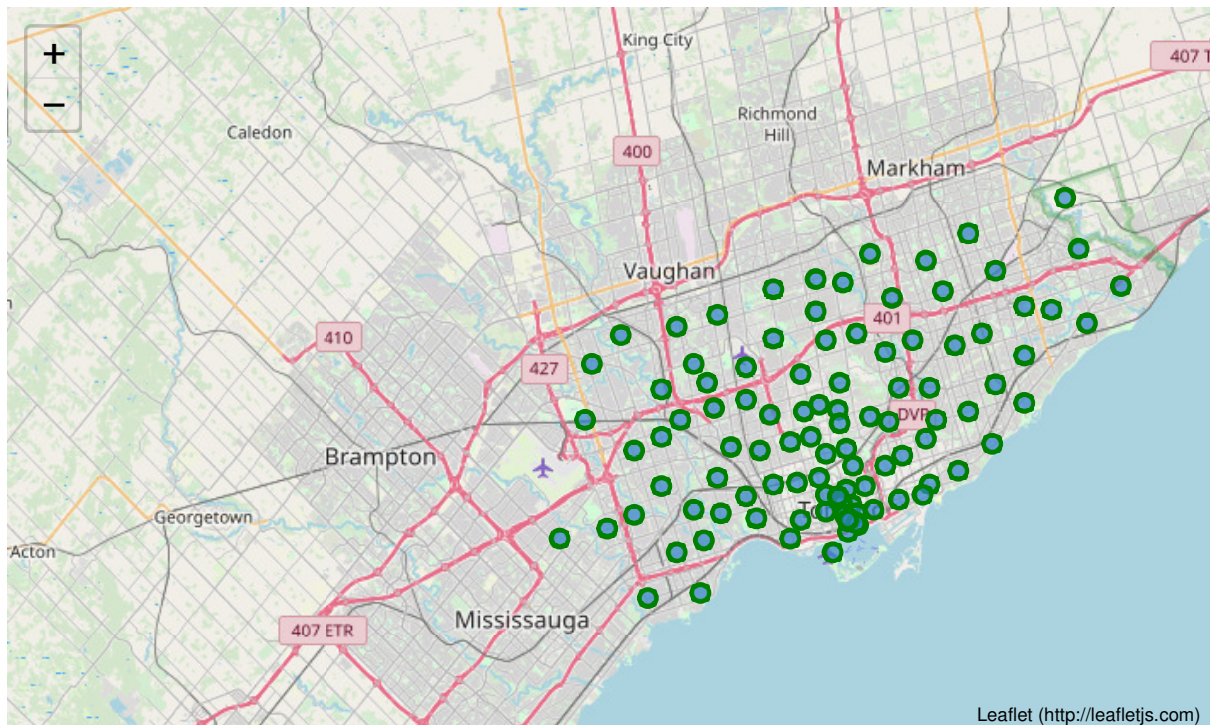
Next a map of the different neighborhoods of Toronto are generated

```
In [45]: map_to = folium.Map(location=[latitude, longitude], zoom_start=10)

# add the map markers
for lat, lng, borough, neighborhood in zip(geo['Latitude'], geo['Longitude'], geo['Borough'], geo['Neighborhood']):
    label = '{} , {}'.format(neighborhood, borough)
    label = folium.Popup(label, parse_html = True)
    folium.CircleMarker(
        [lat, lng],
        radius = 5,
        popup = label,
        color = 'green',
        fill = True,
        fill_color = '#3186cc',
        fill_opacity = 0.7,
        parse_html = False).add_to(map_to)
```

map_to

Out [45]:



```
In [53]: tn_data = geo[geo['Borough'] == 'Downtown Toronto'].reset_index(drop=True)
         tn_data
```

Out [53]:

	PostalCode	Borough	Neighborhood	Latitude	Longitude
0	M4W	Downtown Toronto	Rosedale	43.679563	-79.377529
1	M4X	Downtown Toronto	Cabbagetown, St. James Town	43.667967	-79.367675
2	M4Y	Downtown Toronto	Church and Wellesley	43.665860	-79.383160
3	M5A	Downtown Toronto	Harbourfront	43.654260	-79.360636
4	M5B	Downtown Toronto	Ryerson, Garden District	43.657162	-79.378937
5	M5C	Downtown Toronto	St. James Town	43.651494	-79.375418
6	M5E	Downtown Toronto	Berczy Park	43.644771	-79.373306
7	M5G	Downtown Toronto	Central Bay Street	43.657952	-79.387383
8	M5H	Downtown Toronto	Adelaide, King, Richmond	43.650571	-79.384568
9	M5J	Downtown Toronto	Harbourfront East, Toronto Islands, Union Station	43.640816	-79.381752
10	M5K	Downtown Toronto	Design Exchange, Toronto Dominion Centre	43.647177	-79.381576
11	M5L	Downtown Toronto	Commerce Court, Victoria Hotel	43.648198	-79.379817
12	M5S	Downtown Toronto	Harbord, University of Toronto	43.662696	-79.400049
13	M5T	Downtown Toronto	Chinatown, Grange Park, Kensington Market	43.653206	-79.400049
14	M5V	Downtown Toronto	CN Tower, Bathurst Quay, Island airport, Harbo...	43.628947	-79.394420
15	M5W	Downtown Toronto	Stn A PO Boxes 25 The Esplanade	43.646435	-79.374846
16	M5X	Downtown Toronto	First Canadian Place, Underground city	43.648429	-79.382280
17	M6G	Downtown Toronto	Christie	43.669542	-79.422564
18	M7A	Downtown Toronto	Queen's Park	43.662301	-79.389494

Exploration of Toronto using Foursquare API

The first glimpse of the venues

```
In [90]: address = 'Downtown Toronto, ON'

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

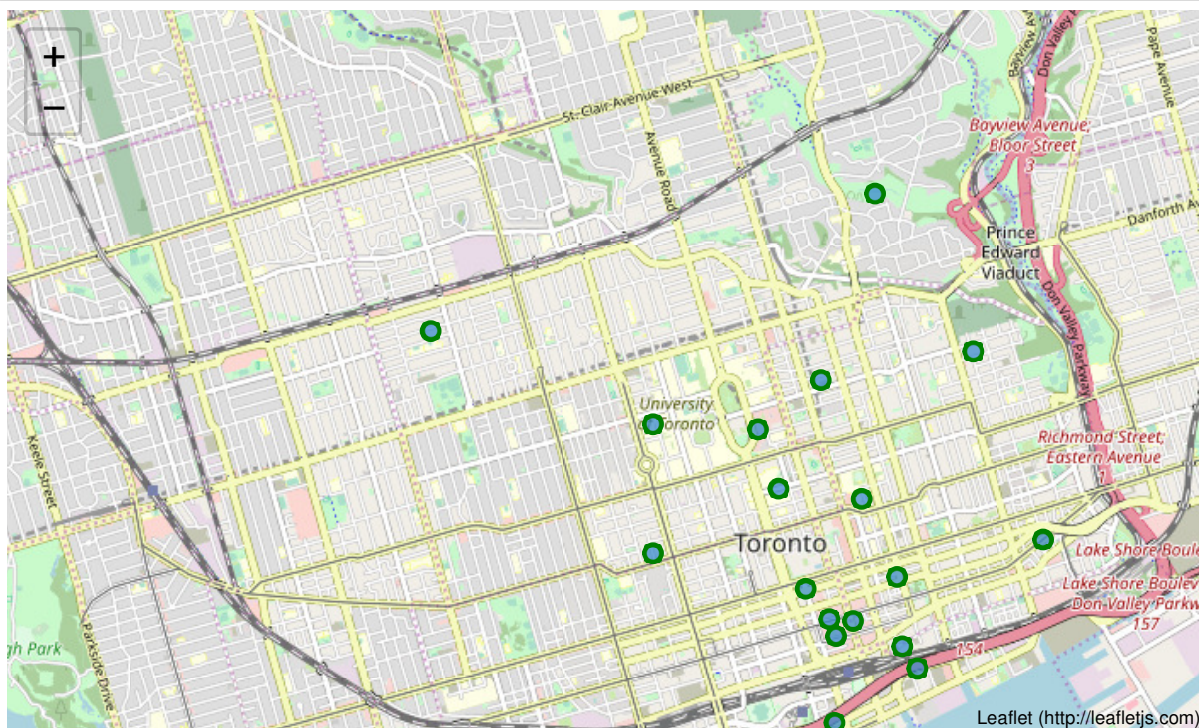
The geograpical coordinate of Downtown Toronto are 43.6563221, -79.3809161.

```
In [92]: # create map of Downtown Toronto using latitude and longitude values
map_dt = folium.Map(location=[latitude, longitude], zoom_start=13)

# add markers to map
for lat, lng, label in zip(dt_data['Latitude'], dt_data['Longitude'], dt_data['Neighborhood']):
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius = 5,
        popup = label,
        color = 'green',
        fill = True,
        fill_color = '#3186cc',
        fill_opacity = 0.7,
        parse_html = False).add_to(map_dt)

map_dt
```

Out [92]:



```
In [93]: CLIENT_ID = 'TKWB5N0HKVWA3JDXVKKZED0VKRX4AHU2UEVGZ0BHRMDET0HF' # your Foursquare ID
CLIENT_SECRET = 'VQE04FETG0HFOFVM2PYLUFLMPQZFHH314RZHEN0FFNVIEKBL' # your Foursquare Secret
VERSION = '20191231' # Foursquare API version
```

The Chinatown, Grange Park, Kensington Market neighborhood of Toronto was next explored

```
In [94]: dt_data.loc[13, 'Neighborhood']
```

Out [94]: 'Chinatown, Grange Park, Kensington Market'

```
In [95]: to_geo_latitude = dt_data.loc[13, 'Latitude'] # neighborhood latitude value
to_geo_longitude = dt_data.loc[13, 'Longitude'] # neighborhood longitude value

to_geo_name = dt_data.loc[13, 'Neighborhood'] # neighborhood name

print('Latitude and longitude values of {} are {}, {}'.format(to_geo_name,
                                                              to_geo_latitude,
                                                              to_geo_longitude))
```

Latitude and longitude values of Chinatown, Grange Park, Kensington Market are 43.6532057, -79.4000493.

```
In [116]: 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,
    to_geo_latitude,
    to_geo_longitude,
    radius,
    LIMIT)
url
```

```
Out[116]: 'https://api.foursquare.com/v2/venues/explore?&client_id=TKWB5N0HKVWA3JDXVKKZED0VKRX
4AHU2UEVGZ0BHRMDET0HF&client_secret=VQE04FETG0HFOFVM2PYLUFLMPQZFHH314RZHEN0FFNVIEKB
L&v=20191231&ll=43.6532057,-79.4000493&radius=500&limit=100'
```

```
In [117]: results = requests.get(url).json()
```

```
In [118]: # 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'][0]['groups'][0]['items']  
nearby_venues = json_normalize(venues) # flatten JSON
```

filter columns

```
filtered_columns = ['venue.name', 'venue.categories', 'venue.location.lat', 'venue.location.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(".")[0] for col in nearby_venues.columns]  
nearby_venues.head()
```

```
In [120]: print('{} venues were returned by Foursquare.'.format(nearby_venues.shape[0]))  
92 venues were returned by Foursquare.
```

Other neighborhoods in the downtown Toronto are were next explored:


```
In [121]: 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)
```

```
In [122]: dt_venues = getNearbyVenues(names = dt_data['Neighborhood'],
                                         latitudes = dt_data['Latitude'],
                                         longitudes = dt_data['Longitude']
                                         )
```

Rosedale
Cabbagetown, St. James Town
Church and Wellesley
Harbourfront
Ryerson, Garden District
St. James Town
Berczy Park
Central Bay Street
Adelaide, King, Richmond
Harbourfront East, Toronto Islands, Union Station
Design Exchange, Toronto Dominion Centre
Commerce Court, Victoria Hotel
Harbord, University of Toronto
Chinatown, Grange Park, Kensington Market
CN Tower, Bathurst Quay, Island airport, Harbourfront West, King and Spadina, Railwa
y Lands, South Niagara
Stn A PO Boxes 25 The Esplanade
First Canadian Place, Underground city
Christie
Queen's Park

Data Exploration of number of venues in the Downtown Toronto area:

```
In [123]: print(dt_venues.shape)
          dt_venues.head()
```

(1314, 7)

Out [123]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Rosedale	43.679563	-79.377529	Rosedale Park	43.682328	-79.378934	Playground
1	Rosedale	43.679563	-79.377529	Whitney Park	43.682036	-79.373788	Park
2	Rosedale	43.679563	-79.377529	Alex Murray Parkette	43.678300	-79.382773	Park
3	Rosedale	43.679563	-79.377529	Milkman's Lane	43.676352	-79.373842	Trail
4	Cabbagetown, St. James Town	43.667967	-79.367675	Cranberries	43.667843	-79.369407	Diner

Number of venues in each area

```
In [124]: dt_venues.groupby('Neighborhood').count()
```

```
Out [124]:
```

	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
Neighborhood						
Adelaide, King, Richmond	100	100	100	100	100	100
Berczy Park	57	57	57	57	57	57
CN Tower, Bathurst Quay, Island airport, Harbourfront West, King and Spadina, Railway Lands, South Niagara	12	12	12	12	12	12
Cabbagetown, St. James Town	43	43	43	43	43	43
Central Bay Street	82	82	82	82	82	82
Chinatown, Grange Park, Kensington Market	92	92	92	92	92	92
Christie	17	17	17	17	17	17
Church and Wellesley	85	85	85	85	85	85
Commerce Court, Victoria Hotel	100	100	100	100	100	100
Design Exchange, Toronto Dominion Centre	100	100	100	100	100	100
First Canadian Place, Underground city	100	100	100	100	100	100
Harbord, University of Toronto	36	36	36	36	36	36
Harbourfront	48	48	48	48	48	48
Harbourfront East, Toronto Islands, Union Station	100	100	100	100	100	100
Queen's Park	41	41	41	41	41	41
Rosedale	4	4	4	4	4	4
Ryerson, Garden District	100	100	100	100	100	100
St. James Town	100	100	100	100	100	100
Stn A PO Boxes 25 The Esplanade	97	97	97	97	97	97

```
In [125]: print('There are {} uniques categories.'.format(len(dt_venues['Venue Category'].unique())))
```

There are 211 uniques categories.

Next 'one hot encoding' is applied to obtain the different venue categories:


```
In [126]: # one hot encoding
dt_onehot = pd.get_dummies(dt_venues[['Venue Category']], prefix="", prefix_sep="")

# add neighborhood column back to dataframe
dt_onehot['Neighborhoods'] = dt_venues['Neighborhood']
# using 'Neighborhoods' instead of 'Neighborhood' to avoid confusions with the original neighborhood column in the dataframe

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

dt_onehot.head()
```

Out [126]:

	Neighborhoods	Airport	Airport Food Court	Airport Lounge	Airport Service	Airport Terminal	American Restaurant	Antique Shop	Aquarium	Art Gallery	Arts & Crafts Store	Re
0	Rosedale	0	0	0	0	0	0	0	0	0	0	
1	Rosedale	0	0	0	0	0	0	0	0	0	0	
2	Rosedale	0	0	0	0	0	0	0	0	0	0	
3	Rosedale	0	0	0	0	0	0	0	0	0	0	
4	Cabbagetown, St. James Town	0	0	0	0	0	0	0	0	0	0	

```
In [127]: dt_onehot.shape
```

Out [127]: (1314, 212)

The data is explored further to examine the frequencies within each category when grouped by neighborhoods:

```
In [128]: dt_grouped = dt_onehot.groupby('Neighborhoods').mean().reset_index()
dt_grouped
```

Out[128]:

	Neighborhoods	Airport	Airport Food Court	Airport Lounge	Airport Service	Airport Terminal	American Restaurant	Antique Shop	Aquarium	Art Gallery	
0	Adelaide, King, Richmond	0.000000	0.000000	0.000000	0.000000	0.000000	0.020000	0.000000	0.00	0.010000	0
1	Berczy Park	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00	0.017544	0
2	CN Tower, Bathurst Quay, Island airport, Harbo...	0.083333	0.083333	0.166667	0.166667	0.083333	0.000000	0.000000	0.00	0.000000	0
3	Cabbagetown, St. James Town	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	0
4	Central Bay Street	0.000000	0.000000	0.000000	0.000000	0.000000	0.012195	0.000000	0.00	0.000000	0
5	Chinatown, Grange Park, Kensington Market	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	0
6	Christie	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	0
7	Church and Wellesley	0.000000	0.000000	0.000000	0.000000	0.000000	0.011765	0.000000	0.00	0.000000	0
8	Commerce Court, Victoria Hotel	0.000000	0.000000	0.000000	0.000000	0.000000	0.030000	0.000000	0.00	0.010000	0
9	Design Exchange, Toronto Dominion Centre	0.000000	0.000000	0.000000	0.000000	0.000000	0.030000	0.000000	0.00	0.010000	0
10	First Canadian Place, Underground city	0.000000	0.000000	0.000000	0.000000	0.000000	0.020000	0.000000	0.00	0.010000	0
11	Harbord, University of Toronto	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	0
12	Harbourfront	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.020833	0.00	0.020833	0
13	Harbourfront East, Toronto Islands, Union Station	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.05	0.010000	0
14	Queen's Park	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	0
15	Rosedale	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	0
16	Ryerson, Garden District	0.000000	0.000000	0.000000	0.000000	0.000000	0.010000	0.000000	0.00	0.010000	0
17	St. James Town	0.000000	0.000000	0.000000	0.000000	0.000000	0.010000	0.000000	0.00	0.010000	0
18	Stn A PO Boxes 25 The Esplanade	0.000000	0.000000	0.000000	0.000000	0.000000	0.010309	0.010309	0.00	0.020619	0

```
In [129]: dt_grouped.shape
```

Out[129]: (19, 212)

The 5 most common venues for each neighborhood are identified

```
In [130]: num_top_venues = 5

for hood in dt_grouped['Neighborhoods']:
    print("-----"+hood+"-----")
    temp = dt_grouped[dt_grouped['Neighborhoods'] == 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	Coffee Shop	0.08
1	Café	0.05
2	Bar	0.04
3	Steakhouse	0.04
4	Sushi Restaurant	0.03

----Berczy Park----

	venue	freq
0	Coffee Shop	0.09
1	Cocktail Bar	0.05
2	Bakery	0.04
3	Farmers Market	0.04
4	Beer Bar	0.04

----CN Tower, Bathurst Quay, Island airport, Harbourfront West, King and Spadina, Railway Lands, South Niagara----

	venue	freq
0	Airport Lounge	0.17
1	Airport Service	0.17
2	Airport	0.08
3	Boat or Ferry	0.08
4	Sculpture Garden	0.08

----Cabbagetown, St. James Town----

	venue	freq
0	Café	0.07
1	Coffee Shop	0.07
2	Italian Restaurant	0.05
3	Bakery	0.05
4	Pub	0.05

----Central Bay Street----

	venue	freq
0	Coffee Shop	0.17
1	Café	0.05
2	Italian Restaurant	0.05
3	Burger Joint	0.04
4	Sandwich Place	0.04

----Chinatown, Grange Park, Kensington Market----

	venue	freq
0	Café	0.07
1	Vietnamese Restaurant	0.05
2	Vegetarian / Vegan Restaurant	0.05
3	Chinese Restaurant	0.04
4	Coffee Shop	0.04

----Christie----

	venue	freq
0	Café	0.18
1	Grocery Store	0.18
2	Park	0.12
3	Convenience Store	0.06
4	Athletics & Sports	0.06

----Church and Wellesley----

	venue	freq
--	-------	------

A data frame is constructed to include the above information

The venues are sorted in descending order:

```
In [131]: 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]
```

Similarly a top 10 list of common venues for each are is then constructed:

```
In [132]: num_top_venues = 10

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

# create columns according to number of top venues
columns = ['Neighborhoods']
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['Neighborhoods'] = dt_grouped['Neighborhoods']

for ind in np.arange(dt_grouped.shape[0]):
    neighborhoods_venues_sorted.iloc[ind, 1:] = return_most_common_venues(dt_grouped.i
loc[ind, :], num_top_venues)

neighborhoods_venues_sorted.head()
```

Out [132]:

	Neighborhoods	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 M Com Ve
0	Adelaide, King, Richmond	Coffee Shop	Café	Steakhouse	Bar	Restaurant	Burger Joint	Sushi Restaurant	Asian Restaurant	Restau
1	Berczy Park	Coffee Shop	Cocktail Bar	Farmers Market	Seafood Restaurant	Steakhouse	Bakery	Beer Bar	Cheese Shop	
2	CN Tower, Bathurst Quay, Island airport, Harbo...	Airport Lounge	Airport Service	Harbor / Marina	Sculpture Garden	Airport Food Court	Airport Terminal	Boat or Ferry	Boutique	Rental Loc
3	Cabbagetown, St. James Town	Café	Coffee Shop	Pizza Place	Restaurant	Pub	Bakery	Italian Restaurant	Liquor Store	Pet S
4	Central Bay Street	Coffee Shop	Café	Italian Restaurant	Burger Joint	Sandwich Place	Ice Cream Shop	Chinese Restaurant	Japanese Restaurant	Bu Tea S

K-Mean Clustering is then run to sort the neighborhoods into 5 clusters

```
In [133]: # set number of clusters
kclusters = 5

dt_grouped_clustering = dt_grouped.drop('Neighborhoods', 1)

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

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

Out[133]: array([0, 0, 2, 0, 3, 0, 4, 0, 0, 0], dtype=int32)

A new dataframe is then constructed comprising of the clusters and the top 10 venues for each neighborhood:

```
In [134]: # add clustering labels
neighborhoods_venues_sorted.insert(0, 'Cluster Labels', kmeans.labels_)

dt_merged = dt_data

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

#neighborhoods_venues_sorted = neighborhoods_venues_sorted.drop(columns = ['Cluster Labels'])

#neighborhoods_venues_sorted.head()

dt_merged.head()
```

Out[134]:

	PostalCode	Borough	Neighborhood	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue
0	M4W	Downtown Toronto	Rosedale	43.679563	-79.377529	1	Park	Playground	Trail	Dessert Shop
1	M4X	Downtown Toronto	Cabbagetown, St. James Town	43.667967	-79.367675	0	Café	Coffee Shop	Pizza Place	Restaurant
2	M4Y	Downtown Toronto	Church and Wellesley	43.665860	-79.383160	0	Coffee Shop	Japanese Restaurant	Sushi Restaurant	Gay Bar
3	M5A	Downtown Toronto	Harbourfront	43.654260	-79.360636	3	Coffee Shop	Park	Bakery	Pizzeria
4	M5B	Downtown Toronto	Ryerson, Garden District	43.657162	-79.378937	0	Coffee Shop	Clothing Store	Café	Cosmetic Shop

Visualization of the Clustered Analysis

Out [138]:




```
In [139]: dt_merged.loc[dt_merged['Cluster Labels'] == 0, dt_merged.columns[[2] + list(range(5, dt_merged.shape[1]))]]
```

Out [139]:

	Neighborhood	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 Common Venue
1	Cabbagetown, St. James Town	0	Café	Coffee Shop	Pizza Place	Restaurant	Pub	Bakery	Italian Restaurant	
2	Church and Wellesley	0	Coffee Shop	Japanese Restaurant	Sushi Restaurant	Gay Bar	Restaurant	Burger Joint	Gym	Bubbly
4	Ryerson, Garden District	0	Coffee Shop	Clothing Store	Café	Cosmetics Shop	Bakery	Middle Eastern Restaurant	Theater	Street
5	St. James Town	0	Café	Coffee Shop	Restaurant	Hotel	Clothing Store	Cosmetics Shop	Beer Bar	Coffee
6	Berczy Park	0	Coffee Shop	Cocktail Bar	Farmers Market	Seafood Restaurant	Steakhouse	Bakery	Beer Bar	Coffee
8	Adelaide, King, Richmond	0	Coffee Shop	Café	Steakhouse	Bar	Restaurant	Burger Joint	Sushi Restaurant	Restaurant
9	Harbourfront East, Toronto Islands, Union Station	0	Coffee Shop	Aquarium	Italian Restaurant	Hotel	Café	Scenic Lookout	Restaurant	Bar
10	Design Exchange, Toronto Dominion Centre	0	Coffee Shop	Hotel	Café	Restaurant	Bar	American Restaurant	Seafood Restaurant	Steak
11	Commerce Court, Victoria Hotel	0	Coffee Shop	Café	Restaurant	Hotel	American Restaurant	Seafood Restaurant	Gym	Steak
12	Harbord, University of Toronto	0	Café	Restaurant	Bakery	Sandwich Place	Bookstore	Japanese Restaurant	Italian Restaurant	
13	Chinatown, Grange Park, Kensington Market	0	Café	Vietnamese Restaurant	Vegetarian / Vegan Restaurant	Bar	Dumpling Restaurant	Coffee Shop	Chinese Restaurant	Middle Eastern Restaurant
15	Stn A PO Boxes 25 The Esplanade	0	Coffee Shop	Café	Restaurant	Japanese Restaurant	Cocktail Bar	Hotel	Seafood Restaurant	Beer
16	First Canadian Place, Underground city	0	Coffee Shop	Café	Gym	Hotel	Restaurant	Steakhouse	Asian Restaurant	Bar

Cluster 1 appears to have a lot of coffee shops and restaurants

Exploration of Cluster 2:

```
In [140]: dt_merged.loc[dt_merged['Cluster Labels'] == 1, dt_merged.columns[[2] + list(range(5, dt_merged.shape[1]))]]
```

Out [140]:

	Neighborhood	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
0	Rosedale	1	Park	Playground	Trail	Dessert Shop	Ethiopian Restaurant	Empanada Restaurant	Electronics Store	Eastern European Restaurant

Exploration of Cluster 3

```
In [141]: dt_merged.loc[dt_merged['Cluster Labels'] == 2, dt_merged.columns[[2] + list(range(5, dt_merged.shape[1]))]]
```

Out [141]:

	Neighborhood	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 Cor V
14	CN Tower, Bathurst Quay, Island airport, Harbo...	2	Airport Lounge	Airport Service	Harbor / Marina	Sculpture Garden	Airport Food Court	Airport Terminal	Boat or Ferry	Boutique	F Loc

Exploration of Cluster 4

```
In [142]: dt_merged.loc[dt_merged['Cluster Labels'] == 3, dt_merged.columns[[2] + list(range(5, dt_merged.shape[1]))]]
```

Out [142]:

	Neighborhood	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
3	Harbourfront	3	Coffee Shop	Park	Bakery	Pub	Café	Mexican Restaurant	Breakfast Spot	Restaurant
7	Central Bay Street	3	Coffee Shop	Café	Italian Restaurant	Burger Joint	Sandwich Place	Ice Cream Shop	Chinese Restaurant	Japanese Restaurant
18	Queen's Park	3	Coffee Shop	Gym	Park	College Auditorium	Smoothie Shop	Sandwich Place	Burger Joint	Burrito Place

Cluster 4 also has a number of restaurants and coffee shops

Exploration of Cluster 5

```
In [143]: dt_merged.loc[dt_merged['Cluster Labels'] == 4, dt_merged.columns[[2] + list(range(5,
dt_merged.shape[1]))]]
```

Out[143]:

	Neighborhood	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
17	Christie	4	Grocery Store	Café	Park	Candy Store	Diner	Italian Restaurant	Baby Store	Athletics & Sports	Restaurants

Cluster 5 appears to be more of a residential area with service businesses

Results

The results of this study are shown above. The results indicate that when we use the K-means clustering for the Toronto downtown area, it arranges the venues across 5 different clusters. Cluster 1 has a heavier concentration of restaurants and coffee shops. Clusters 2 and 5 appear more residential with a higher concentration of grocery stores, parks, playgrounds, candy stores, baby stores etc. Cluster 3 is the area around the airport. Finally, cluster 4 also has a high concentration of restaurants, coffee shops and other food places. Cluster 4 appears to be an area that caters to students and young adults.

Discussion and Conclusion

The results from this study indicate that while Clusters 1 and 4 present the biggest market for opening restaurants, there is also a lot of competition given the heavy concentration of food businesses in these areas. Perhaps opening a restaurant in Cluster 2 or Cluster 5 will have higher risk but also the potential for greater opportunities. Cluster 3, which is the airport area, is also another possibility, since Toronto is a large international airport and many tourists and travelers pass by this area everyday.

For small service-oriented businesses such as laundry, day care, plumbing, electrical, financial services etc., the residential clusters 2 and 5 have the most opportunity. Additionally, Cluster 3 that has educational institutions nearby, also provides the opportunity for opening some of these businesses as well.

Overall, the capstone case was a beneficial exercise, which helped me pull all of the learning from the previous courses and integrating the tools in analyzing this case study. I felt like I learned a lot about geospatial analysis, and obtaining a RESTful API from a service like Foursquare.