CAPSTONE PROJECT: BATTLE OF NEIGHBORHOODS

Asim Islam
C9_wk5_CAPSTONE_BattleofNeighborhoods

---

1. Introduction

The population of Toronto has grown considerably in the past decade and is very diverse. It is a great opportunity for entrepreneurs to cater to such a multi-cultural population.

1.1 Business Problem

A group of investors is looking to open an authentic South Asian restaurant and is in the process of finding a good location. The business problem is to identify neighborhoods where there is a high number of residents that are of South Asian descent.

1.1 Target Audience

The target audience is, like the population, diverse. Toronto has a variety of restaurants catering to almost everyone. The intent of the investors is to target the South Asian communities, as well as other residents, visitors and tourists

---

2. Data

Postal codes of Toronto, Neighborhood profiles from the City of Tornoto, Geocoder ARCGIS and FourSquare will be used to collect the data.

Importing Libraries

In [29]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from bs4 import BeautifulSoup  # scrape websites
import requests
import geocoder
import folium
from pandas.io.json import json_normalize

%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

2.1 Postal Codes of Toronto

Postal Codes of Toronto will be scraped from Toronto's Postal Code wiki-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 will be used in conjunction with ARCGIS and Four Square to explore the neighborhoods.

The Postal Code dataframe will be cleaned as follows:

- Borough "Not assigned" will be removed
- Neighborhood that are "Not assigned" will be the same as Borough

The finished dataframe from this section will contain:

- PostalCode
- Borough
- Neighborhood
- Latitude
- Longitude

In [12]:
```
#=====================================================
#  Tonronto Postal Codes
#=====================================================
url = 'https://en.wikipedia.org/wiki/List_of_postal_codes_of_Canada:_M'
wiki_html = requests.get(url).text
soup = BeautifulSoup(wiki_html, 'lxml')
#print(soup.prettify())

#  Scrape 'tr' and 'td' tags in loop
wiki = []
for tr in soup.tbody.find_all('tr'):
    wiki.append([ td.get_text().strip() for td in tr.find_all('td')])

#  Setup the data frame
df = pd.DataFrame(data=wiki, columns=['PostalCode','Borough','Neighborhoo
df.head()

#   Ignore Boroughs that are 'Not assigned'
df = df[df['Borough'] != 'Not assigned']

#  COMBINE -
df = df.groupby('PostalCode')['Borough','Neighborhood'].agg(lambda x: ',


#  "Not assigned" neighborhood, will be the same as the borough
df.Neighborhood[df.Neighborhood == 'Not assigned']    # look for "Not ass
df.Neighborhood[df.Neighborhood == 'Not assigned'] = df.Borough[df.Neighb

df[df['Neighborhood'] == 'Not assigned']   #  check
df[df['Neighborhood'] == 'Queen\'s Park']  #  check
```

Out[12]:

	PostalCode	Borough	Neighborhood
85	M7A	Queen's Park	Queen's Park

Using Geocoder ARCGIS to get the Latitude, Logitude of the Postal Codes

In [13]:
```
#  Get LATITUDE and LONGITUDE
#  Define function to get latitude & longitude using postal codes
def get_latlon(postal_code):
    lat_lng_coords = None
    while(lat_lng_coords is None):
        g = geocoder.arcgis('{}, Toronto, Ontario'.format(postal_code))
        lat_lng_coords = g.latlng
    return lat_lng_coords
```

Out[13]: [43.64969222700006, -79.55394499999994]

Updating the dataframe with latitude & longitude

In [14]:
```
postal_codes = df['PostalCode']
#  change postal_codes from series to list for the loop
geo_latlon = [get_latlon(postal_code) for postal_code in postal_codes.tol
df_latlng = pd.DataFrame(data = geo_latlon, columns = {'Latitude','Longit
df_latlng.columns = ['Latitude','Longitude']   #  sometimes dataframe fli

#  Add Latitude and Longitude to the original dataframe
df['Latitude']  = df_latlng['Latitude']
df['Longitude'] = df_latlng['Longitude']

#  Toronto DataFrame
print("PostalCode:\t",len(df['PostalCode'].unique()))
print("Borough:\t",len(df['Borough'].unique()))
print("Neighborhood:\t",len(df['Neighborhood'].unique()))
```

```
PostalCode:     103
Borough:        11
Neighborhood:   103
```

In [15]:
```
df.head()
```

Out[15]:

	PostalCode	Borough	Neighborhood	Latitude	Longitude
0	M1B	Scarborough	Rouge, Malvern	43.811525	-79.195517
1	M1C	Scarborough	Highland Creek, Rouge Hill, Port Union	43.785730	-79.158750
2	M1E	Scarborough	Guildwood, Morningside, West Hill	43.765690	-79.175256
3	M1G	Scarborough	Woburn	43.768359	-79.217590
4	M1H	Scarborough	Cedarbrae	43.769688	-79.239440

2.2 Toronto Neighborhood Profile

Toronto Neighborhood Profile dataset can be downloaded as a .csv file from (https://portal0.cf.opendata.inter.sandbox-toronto.ca/dataset/Neighborhood-profiles/) (https://portal0.cf.opendata.inter.sandbox-toronto.ca/dataset/Neighborhood-profiles/)). From this census data, **language spoken by population** criteria will help identify the neighborhoods that can be targeted as locations for a South Asian restaurant.

Steps for creating the language (South Asian) population dataframe:

- read the dataframe
- select criteria: Topic contains ('language|mother|tongue')
- create the dataframe
- sort the dataframe based on population
- update Toronto dataframe based on profile dataframe

Read in the profile dataframe

In [16]:
```
#  read from download file
df_lang = pd.read_csv('C:/Users/ACER/Desktop/JAVA/IBM_Certificate/data/Ne
df_lang.head()
```

Out[16]:

	_id	Category	Topic	Data Source	Characteristic	City of Toronto	Agincourt North	Agincour South Malveri Wes
0	1	Neighbourhood Information	Neighbourhood Information	City of Toronto	Neighbourhood Number	NaN	129	12
1	2	Neighbourhood Information	Neighbourhood Information	City of Toronto	TSNS2020 Designation	NaN	No Designation	N Designatio
2	3	Population	Population and dwellings	Census Profile 98-316-X2016001	Population, 2016	2,731,571	29,113	23,75
3	4	Population	Population and dwellings	Census Profile 98-316-X2016001	Population, 2011	2,615,060	30,279	21,98
4	5	Population	Population and dwellings	Census Profile 98-316-X2016001	Population Change 2011-2016	4.50%	-3.90%	8.00

5 rows × 146 columns

Select the language criteria

Knowledge of languages will give us the best numbers of the South Asian population.

In [17]:
```
#  Picking the correct sub-cat of languages
df_lang.Topic[df_lang.Topic.str.contains('language|mother|tongue',case=Fa
```

Out[17]:
```
['Knowledge of official languages',
 'First official language spoken',
 'Mother tongue',
 'Language spoken most often at home',
 'Other language spoken regularly at home',
 'Knowledge of languages',
 'Language used most often at work',
 'Other language used regularly at work']
```

Languages under Ingo-Aryan languages

Selecting the language super-category will give us a better population number than selecting, for example, country of origin.

NOTE: "Indo-Aryan" was selected by searching the profile spreadsheet.

In [18]:
```
df_lang.index[df_lang.Topic.str.contains('Knowledge of languages') & df_l
df_lang['Characteristic'][849:849+17]
```

Out[18]:
```
849                Indo-Aryan languages
850                             Bengali
851                            Gujarati
852                               Hindi
853                            Kashmiri
854                             Konkani
855                             Marathi
856                              Nepali
857                        Oriya (Odia)
858                   Punjabi (Panjabi)
859                              Sindhi
860                 Sinhala (Sinhalese)
861                                Urdu
862                   Iranian languages
863                             Kurdish
864                              Pashto
865                     Persian (Farsi)
Name: Characteristic, dtype: object
```

Creating Neighborhood/Language DataFrame

In [19]:
```
#  Creating Neighborhood/Language DataFrame
df_lang = df_lang.loc[df_lang.index[df_lang.Topic.str.contains('Knowledge
df_lang = df_lang.transpose().reset_index()
df_lang = df_lang[['index', 849, 674]]   #  reorder columns
df_lang.columns = ['Neighborhood','IndoAryan','English']
df_lang = df_lang[6:]   #  updating dataframe to only Neighborhood
df_lang = df_lang.reset_index(drop=True)

#  convert to int64 for sorting
df_lang.info()
df_lang.Neighborhood = df_lang.Neighborhood.astype(str)
df_lang.IndoAryan = pd.to_numeric(df_lang.IndoAryan.str.replace(',',''),
df_lang.English = pd.to_numeric(df_lang.English.str.replace(',',''), erro
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 140 entries, 0 to 139
Data columns (total 3 columns):
Neighborhood    140 non-null object
IndoAryan       140 non-null object
English         140 non-null object
dtypes: object(3)
memory usage: 3.4+ KB
```

Sorting dataframe

In [24]:
```
df_lang_sorted = df_lang.sort_values(['IndoAryan','English'], ascending=[
print('\n\t--- Top IndoAryan Speaking Neighborhoods---\n\n',df_lang_sorte
```

```
	--- Top IndoAryan Speaking Neighborhoods---

                           Neighborhood  IndoAryan  English
0                                Woburn      12760    50395
1                West Humber-Clairville      10775    31380
2                      Thorncliffe Park       8845    19510
3    Mount Olive-Silverstone-Jamestown       8110    30240
4                               Malvern       7750    41615
5                        Flemingdon Park       6115    20245
6                                 Rouge       5430    44585
7                         Taylor-Massey       5095    14935
8                              Bendale       4735    27505
9                    Clairlea-Birchmount       4575    25330
10  Waterfront Communities-The Island       4290    64665
11                             Oakridge       4110    13020
12                          Dorset Park       3145    23130
13                        Humber Summit       3085    11395
14                   Scarborough Village       2920    15765
15        Kingsview Village-The Westway       2735    21125
16                            West Hill       2625    26325
17                           L'Amoreaux       2550    37545
18                           Humbermede       2535    14360
19                         Kennedy Park       2400    16505
20              Tam O'Shanter-Sullivan       2365    24290
21               York University Heights       2305    26140
22                     Parkwoods-Donalda       2295    33660
23                         Eglinton East       2225    21510
24          Agincourt South-Malvern West       2215    20125
25                 North St. James Town       2200    17830
26                           Henry Farm       2200    14855
27          Thistletown-Beaumond Heights       2130     9540
28             Islington-City Centre West       2120    41810
29                      Victoria Village       2075    16495
```

Cleanup Neighborhoods/PostalCodes in the Dataframes*

This was a manual and time-consuing task since some neighborhoods in the sorted list had the same Postal Codes. The following steps resulted in a clean dataframe that could be used for location, modeling and analysis:

1. Lookup neighborhoods in the Toronto DataFrame (df)
2. Lookup missing postal codes at https://worldpostalcode.com/search (https://worldpostalcode.com/search)
3. Lookup postal codes in the Toronto DataFrame (df)
4. Setup an index and check for duplicates
5. Update the drateframe and do a check

1. Lookup neighborhoods in the Toronto DataFrame (df)

Empty cells "[]" were not found in the postal dataframe.

In [25]:
```
for i in df_lang_sorted.Neighborhood:
    print(df.index[df.Neighborhood.str.contains(i,case=False)],i)
```

```
Int64Index([3], dtype='int64') Woburn
Int64Index([], dtype='int64') West Humber-Clairville
Int64Index([39], dtype='int64') Thorncliffe Park
Int64Index([], dtype='int64') Mount Olive-Silverstone-Jamestown
Int64Index([0], dtype='int64') Malvern
Int64Index([27], dtype='int64') Flemingdon Park
Int64Index([0, 1, 16], dtype='int64') Rouge
Int64Index([], dtype='int64') Taylor-Massey
Int64Index([], dtype='int64') Bendale
Int64Index([], dtype='int64') Clairlea-Birchmount
Int64Index([], dtype='int64') Waterfront Communities-The Island
Int64Index([7], dtype='int64') Oakridge
Int64Index([10], dtype='int64') Dorset Park
Int64Index([96], dtype='int64') Humber Summit
Int64Index([5, 8], dtype='int64') Scarborough Village
Int64Index([], dtype='int64') Kingsview Village-The Westway
Int64Index([2], dtype='int64') West Hill
Int64Index([14, 15], dtype='int64') L'Amoreaux
Int64Index([], dtype='int64') Humbermede
Int64Index([6], dtype='int64') Kennedy Park
Int64Index([], dtype='int64') Tam O'Shanter-Sullivan
Int64Index([], dtype='int64') York University Heights
Int64Index([], dtype='int64') Parkwoods-Donalda
Int64Index([], dtype='int64') Eglinton East
Int64Index([], dtype='int64') Agincourt South-Malvern West
Int64Index([], dtype='int64') North St. James Town
Int64Index([18], dtype='int64') Henry Farm
Int64Index([], dtype='int64') Thistletown-Beaumond Heights
Int64Index([], dtype='int64') Islington-City Centre West
Int64Index([34], dtype='int64') Victoria Village
```

2. Lookup missing postal codes at [https://worldpostalcode.com/search (https://worldpostalcode.com/search)](https://worldpostalcode.com/search)

3. Lookup postal codes in the Toronto DataFrame (df)

After searching [https://worldpostalcode.com/search (https://worldpostalcode.com/search)](https://worldpostalcode.com/search), the postal codes were again seached one by one in the postal dataframe.

In [26]:
```
df.loc[df.index[df.PostalCode.str.contains('M9W')]]   #  102  West Humber
df.loc[df.index[df.PostalCode.str.contains('M9V')]]   #  101  Mount Olive
df.loc[df.index[df.PostalCode.str.contains('M1B')]]   #    0  Malvern
df.loc[df.index[df.PostalCode.str.contains('M1B')]]   #    1  Rouge
df.loc[df.index[df.PostalCode.str.contains('M4C')]]   #   36  Taylor-Mass
df.loc[df.index[df.PostalCode.str.contains('M1P')]]   #   10  Bendale
df.loc[df.index[df.PostalCode.str.contains('M1L')]]   #    7  Clairlea-Bi
df.loc[df.index[df.PostalCode.str.contains('M5E')]]   #   56  Waterfront
df.loc[df.index[df.PostalCode.str.contains('M9R')]]   #  100  Kingsview V
df.loc[df.index[df.PostalCode.str.contains('M9M')]]   #   97  Humbermede
df.loc[df.index[df.PostalCode.str.contains('M1T')]]   #   13  Tam O'Shant
df.loc[df.index[df.PostalCode.str.contains('M3J')]]   #   29  York Univer
df.loc[df.index[df.PostalCode.str.contains('M3A')]]   #   25  Parkwoods-D
```

Out[26]:

	PostalCode	Borough	Neighborhood	Latitude	Longitude
25	M3A	North York	Parkwoods	43.75244	-79.329271

```
*4.  Setup an index and check for duplicates*
Set up index and update the dataframe.
"set()" = None means that there are no duplicates.  So we are on our way
```

In [27]:
```
neigh_index = [3,102,39,101,0,27,1,36,10,7,56,96,5,100,2,14,97,13,29,25]
len(neigh_index)          # list length
set([x for x in neigh_index if neigh_index.count(x)>1])  #  check for dup
```

Out[27]: set()

5. Update the drateframe and do a check

The postal dataframe will be updated with the neighborhood names. Also, the commas will be removed since the entry will have only one neighborhood.

In [34]:

```
#  Update df dataframe with neighborhood names from df_lang_sorted
#  NOTE:  only missing postal codes are being updated
df['Neighborhood'].iloc[102] = 'West Humber-Clairville'
df['Neighborhood'].iloc[101] = 'Mount Olive-Silverstone-Jamestown'
df['Neighborhood'].iloc[0]   = 'Malvern'
df['Neighborhood'].iloc[27]  = 'Flemingdon Park'
df['Neighborhood'].iloc[1]   = 'Rouge'
df['Neighborhood'].iloc[36]  = 'Taylor-Massey'
df['Neighborhood'].iloc[10]  = 'Bendale'
df['Neighborhood'].iloc[7]   = 'Clairlea-Birchmount'
df['Neighborhood'].iloc[56]  = 'Waterfront Communities-The Island'
df['Neighborhood'].iloc[96]  = 'Humber Summit'
df['Neighborhood'].iloc[5]   = 'Scarborough Village'
df['Neighborhood'].iloc[100] = 'Kingsview Village-The Westway'
df['Neighborhood'].iloc[2]   = 'West Hill'
df['Neighborhood'].iloc[14]  = 'L\'Amoreaux'
df['Neighborhood'].iloc[97]  = 'Humbermede'
df['Neighborhood'].iloc[13]  = 'Tam O\'Shanter-Sullivan'
df['Neighborhood'].iloc[29]  = 'York University Heights'
df['Neighborhood'].iloc[25]  = 'Parkwoods-Donalda'

#  Check
df[['PostalCode','Neighborhood']].loc[neigh_index]
```

Out[34]:

	PostalCode	Neighborhood
3	M1G	Woburn
102	M9W	West Humber-Clairville
39	M4H	Thorncliffe Park
101	M9V	Mount Olive-Silverstone-Jamestown
0	M1B	Malvern
27	M3C	Flemingdon Park
1	M1C	Rouge
36	M4C	Taylor-Massey
10	M1P	Bendale
7	M1L	Clairlea-Birchmount
56	M5E	Waterfront Communities-The Island
96	M9L	Humber Summit
5	M1J	Scarborough Village
100	M9R	Kingsview Village-The Westway
2	M1E	West Hill
14	M1V	L'Amoreaux
97	M9M	Humbermede
13	M1T	Tam O'Shanter-Sullivan
29	M3J	York University Heights
25	M3A	Parkwoods-Donalda

In [35]:
```
#  Create dataframe with the top 20 IndoAryan speaking neighborhoods
df_hood = df.loc[neigh_index].reset_index(drop=True)
df_hood.head()
```

Out[35]:

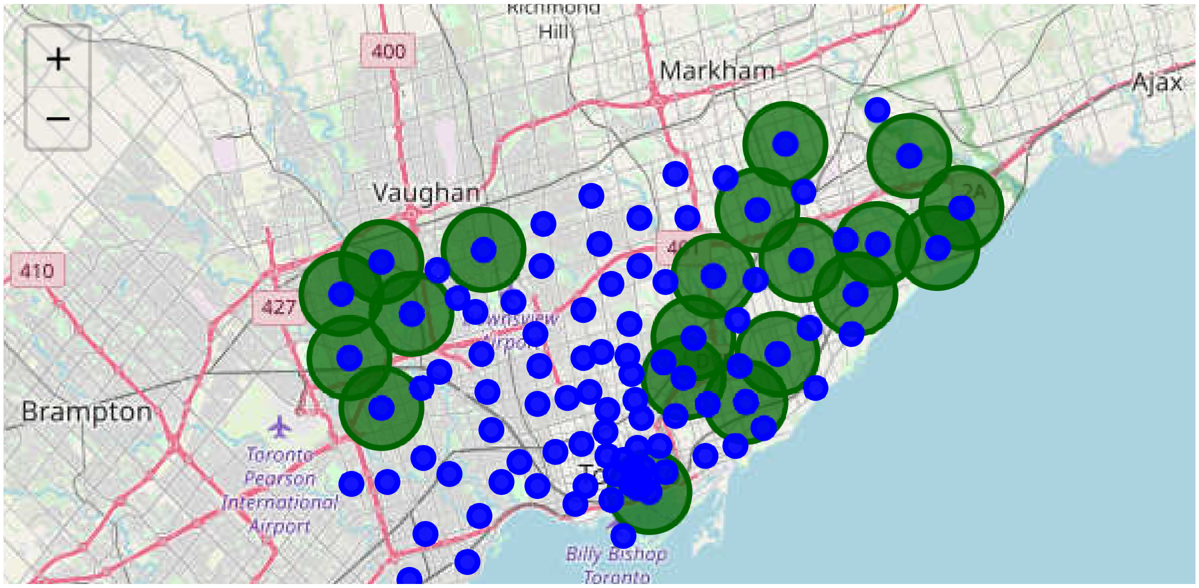
	PostalCode	Borough	Neighborhood	Latitude	Longitude
0	M1G	Scarborough	Woburn	43.768359	-79.217590
1	M9W	Etobicoke	West Humber-Clairville	43.711740	-79.579181
2	M4H	East York	Thorncliffe Park	43.701270	-79.349844
3	M9V	Etobicoke	Mount Olive-Silverstone-Jamestown	43.743205	-79.584701
4	M1B	Scarborough	Malvern	43.811525	-79.195517

FOLIUM map of the Toronto Neighborhoods

Generate a map of all the neighborhoods in Toronto (BLUE) and also of the top Indo Aryan speaking neighborhoods (GREEN).

In [38]:
```
#  Generate Map of Boroughs based on Postal Codes
#====================================================
toronto_map = folium.Map(location=[df.Latitude.mean(), df.Longitude.mean(

# Plot the top 20 IndoAryan speaking neighborhoods
for lat, lng, label in zip(df_hood.Latitude, df_hood.Longitude, df_hood.N
    if label.find(',') != -1:
        label = label.split(',')[0]
    label = '{} (IndoAryan)'.format(label)  # being overwritten by next C
    label = folium.Popup(label, parse_html=True)

    folium.CircleMarker(
            [lat, lng],
            radius=20,
            popup=label,    #   IndoAryan neighborhoods
            fill=True,
            color='darkgreen',
            fill_color='darkgreen',
            fill_opacity=0.7
            ).add_to(toronto_map)

# add points for all the neighborhoods
for lat, lng, label in zip(df.Latitude, df.Longitude, df.Neighborhood):
    if label.find(',') != -1:
        label = label.split(',')[0]
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
            [lat, lng],
            radius=5,
            popup=label,  #  all neighborhoods
            fill=True,
            color='blue',
            fill_color='blue',
            fill_opacity=0.9
            ).add_to(toronto_map)

# display map
toronto_map
```

Out[38]:



2.3 Geocoder - ARCGIS

ARCGIS will be used for looking up the longitude and latitudes of the neighborhoods. It is part of the geocoder package.

*NOTE: Geocoder was used in Toronto Postal Code dataframe section**

2.4 Four Square

Four Square app will be used for exploring venues in a neighborhood.

Four Square Credentials

In [42]:
```
#  Foursquare Credentials and Version
#CLIENT_ID = '<deleted after execution>'
#CLIENT_SECRET = '<deleted after execution>'
VERSION = '20190627'
print('Your credentails:')
#print('CLIENT_ID: ' + CLIENT_ID)
#print('CLIENT_SECRET:' + CLIENT_SECRET)
print('VERSION: ' + VERSION)
```

```
Your credentails:
VERSION:  20190627
```

Four Square - explore the first neighborhood - Woburn, M1G*
Woburn had the highest population of Indo Aryan language speakers. Following section will explore Woburn using Four Square and Folium.

- Four Square explore Woburn, Toronto
- Folium Map of Woburn, Toronto
- Plot of Venues in Woburn, Toronto

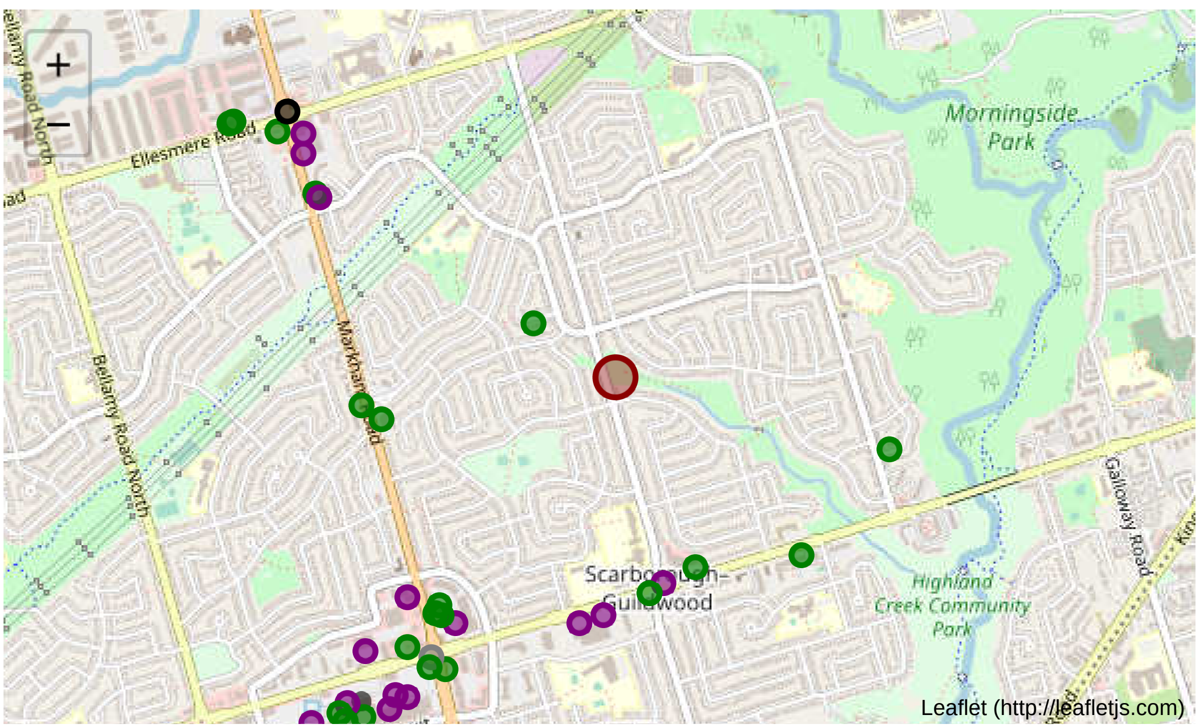
Four Square explore Woburn, Toronto:

In [43]:
```
#====================================================
#  Explore ONE Neighborhood - Woburn, M1G
#====================================================
radius = 1610   #  1 mile radius
LIMIT  = 500    #  4SQ gives a max of 100 venues
latitude =   df_hood.Latitude[0]
longitude =  df_hood.Longitude[0]

url = 'https://api.foursquare.com/v2/venues/explore?client_id={}&client_s
url

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

print('There are {} venues around {} neighborhood.'.format(len(results['r

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

df_jsonE = json_normalize(items) # flatten JSON
df_jsonE.columns


#====================
#  Where to extract the top category for venue
df_jsonE['venue.categories']
df_jsonE['venue.categories'][4][0]
df_jsonE['venue.categories'][4][0]['icon']['prefix']
df_jsonE['venue.categories'][4][0]['icon']['prefix'].split('/')[5]

#  Extract the top category for venue
for i in range(len(df_jsonE)):
    df_jsonE.loc[i,'TopCategory'] = df_jsonE['venue.categories'][i][0]['i
    if (df_jsonE.loc[i,'TopCategory'] == 'food'):
        df_jsonE.loc[i,'TopCatColor'] = 'green'
    if (df_jsonE.loc[i,'TopCategory'] == 'arts_entertainment'):
        df_jsonE.loc[i,'TopCatColor'] = 'blue'
    if (df_jsonE.loc[i,'TopCategory'] == 'travel'):
        df_jsonE.loc[i,'TopCatColor'] = 'black'
    if (df_jsonE.loc[i,'TopCategory'] == 'shops'):
        df_jsonE.loc[i,'TopCatColor'] = 'purple'
    if (df_jsonE.loc[i,'TopCategory'] == 'nightlife'):
        df_jsonE.loc[i,'TopCatColor'] = 'gray'
    if (df_jsonE.loc[i,'TopCategory'] == 'parks_outdoors'):
        df_jsonE.loc[i,'TopCatColor'] = 'red'
    if (df_jsonE.loc[i,'TopCategory'] == 'gym'):
        df_jsonE.loc[i,'TopCatColor'] = 'black'

df_jsonE['TopCategory'].value_counts()
#==============================================


# filter columns
filtered_columns = ['venue.name', 'venue.categories', 'TopCategory','venu
df_woburn = df_jsonE.loc[:, filtered_columns]

# filter the category for each row
```

Folium Map of Woburn, Toronto:

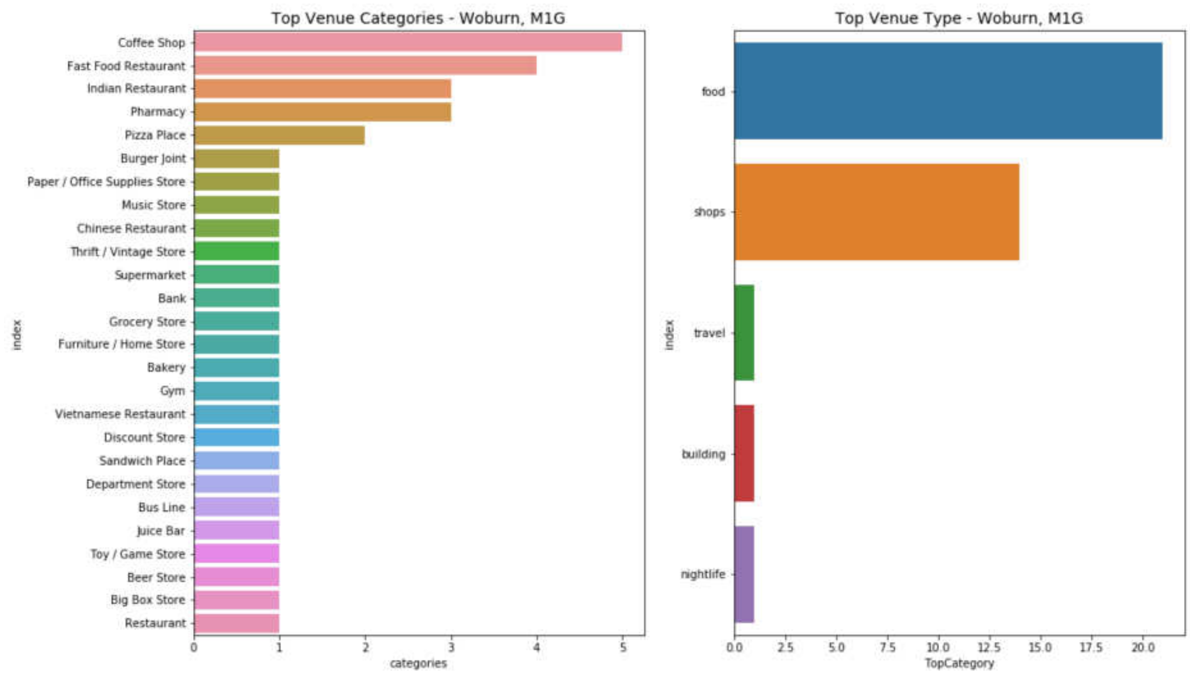
In [54]:
```
Woburn_map = folium.Map(location=[latitude, longitude], zoom_start=14)

# add darkred circle mark for Klyde Warren Park
folium.CircleMarker(
    [latitude, longitude],
    radius=10,
    popup='Woburn, Toronto',
    fill=True,
    color='darkred',
    fill_color='darkred',
    fill_opacity=0.4
    ).add_to(Woburn_map)


# add popular spots to the map as blue circle markers
for lat, lng, cat_label, cat_color in zip(df_woburn.lat, df_woburn.lng, d
    folium.CircleMarker(
        [lat, lng],
        radius=5,
        popup=cat_label,
        fill=True,
        color=cat_color,
        fill_color=cat_color,
        fill_opacity=0.6
        ).add_to(Woburn_map)

# display map
Woburn_map
```

Out[54]:



Plot of Venues in Woburn, Toronto

In [53]:
```
#  Barplot of categories
fig = plt.figure(figsize=(16,10))
fig.add_subplot(121)
df_woburn_cat2_barplt = df_woburn.categories.value_counts().reset_index()
plt.title('Top Venue Categories - Woburn, M1G', fontsize=14)
sns.barplot(data=df_woburn_cat2_barplt, y='index',x='categories')

fig.add_subplot(122)
df_woburn_cat1_barplt = df_woburn.TopCategory.value_counts().reset_index(
plt.title('Top Venue Type - Woburn, M1G', fontsize=14)
plt.ylabel(".")
sns.barplot(data=df_woburn_cat1_barplt, y='index',x='TopCategory')
plt.show()
```



Explore MULTIPLE neighborhoods

Now that we are confident that the Woburn data is good, let's get the venues for the remaining neighborhoods. Out intent is to collect information using Four Square for all of the top 20 Inro Aryan speaking neighborhoods in Toronto.

In [55]:
```
#  function to get FourSquare info for multiple neighborhoods
#------------------------------------------------------
def getNearbyVenues(names, latitudes, longitudes, radius=2000):

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

        # create the API request URL
        url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}
            CLIENT_ID,
            CLIENT_SECRET,
            VERSION,
            lat,
            lng,
            radius,
            LIMIT)

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

        # 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
    nearby_venues.columns = ['Neighborhood',
                  'Neighborhood Latitude',
                  'Neighborhood Longitude',
                  'Venue',
                  'Venue Latitude',
                  'Venue Longitude',
                  'Venue Category']

    return(nearby_venues)

#  Setup dataframe for the multiple neighborhoods
df_multi = getNearbyVenues(names=df_hood['Neighborhood'], latitudes=df_ho
```

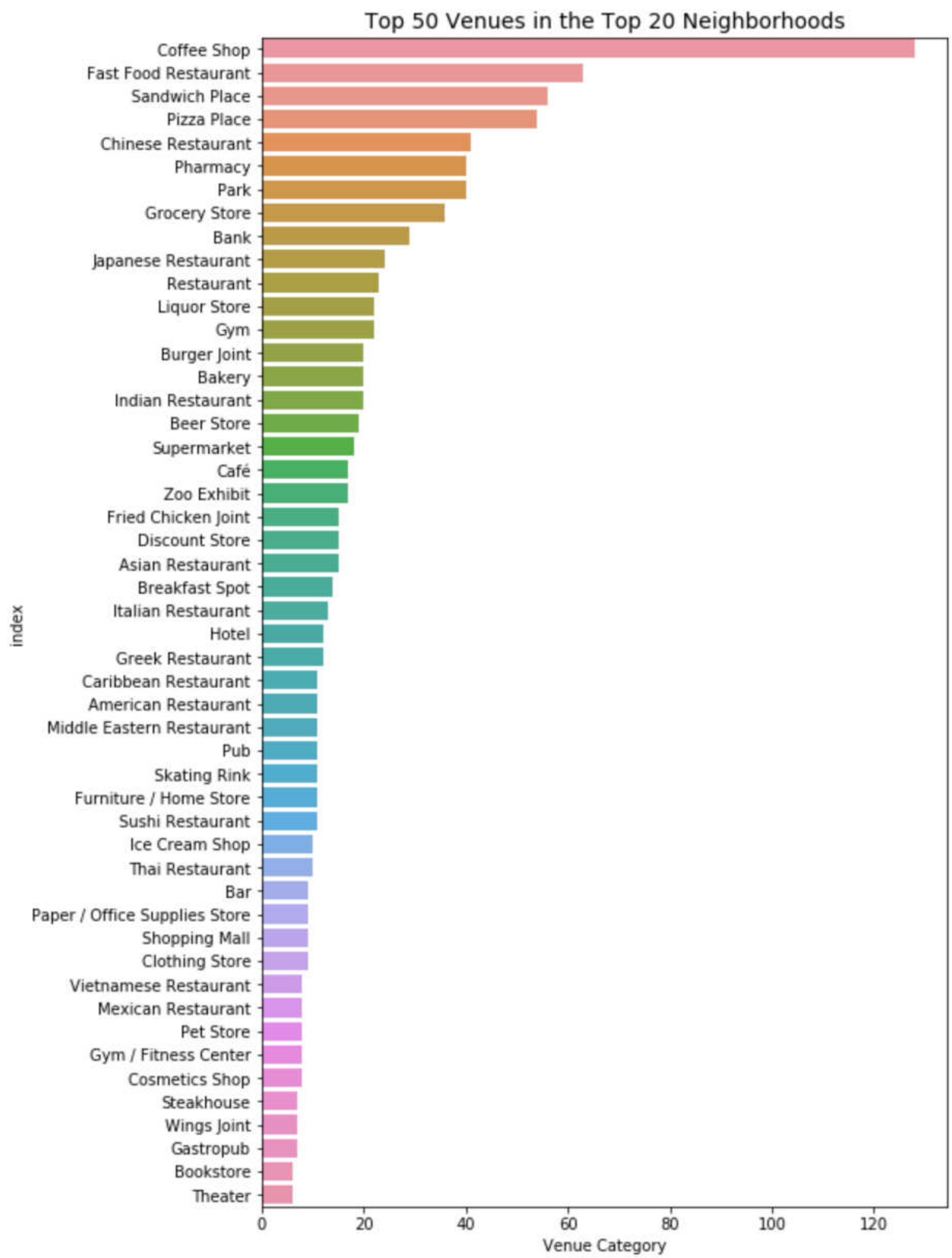
In [61]:
```
print(df_multi.head())
print('\n\nData shape: {}'.format(df_multi.shape))
print('There are {} uniques categories in the top {} neighborhoods.'.form
```

```
  Neighborhood  Neighborhood Latitude  Neighborhood Longitude  \
0       Woburn              43.768359               -79.21759
1       Woburn              43.768359               -79.21759
2       Woburn              43.768359               -79.21759
3       Woburn              43.768359               -79.21759
4       Woburn              43.768359               -79.21759

                                     Venue  Venue Latitude  \
0             The Real McCoy Burgers And Pizza       43.774081
1                                Tim Hortons       43.775992
2  GoodLife Fitness Scarborough Cedarbrae Mall       43.758303
3                                  Starbucks       43.770037
4            Scarborough Golf and Country Club       43.752915

   Venue Longitude Venue Category
0       -79.230496   Burger Joint
1       -79.232135    Coffee Shop
2       -79.228533            Gym
3       -79.221156    Coffee Shop
4       -79.210850    Golf Course


Data shape: (1319, 7)
There are 187 uniques categories in the top 20 neighborhoods.
```

Plot the venues in the neighborhoods

In [63]:
```
#  Barplot of top 50 venues for the top 20 neighborhoods
df_multi_cat_barplt = df_multi['Venue Category'].value_counts().head(50).
plt.figure(figsize=(8,14))
plt.title('Top 50 Venues in the Top 20 Neighborhoods', fontsize=14)
plt.ylabel(".")
sns.barplot(data=df_multi_cat_barplt, y='index',x='Venue Category')
```

Out[63]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x18d7ae48>
```



We now have a high confidence in the data that was collected and organized in the dataframes. We can continue with our analysis of finding the best location for the restaurant.

3. Methodology

Now that we have the data, we can start analyzing the neighborhoods using the following methodologies:

1. One Hot Encoding
2. Exploratory Data Analysis per Neighborhoods
   A. Statistical spread of "Indian Restaurants" per Neighborhoods
   B. Statistical Spread of Venues per Neighborhood
   C. Statistical Spread of Top 10 Venues per Neighborhood
   D. Create Dataframe with Top 10 Venues per Neighborhood

3.1 One Hot Encoding

One hot encoding is a representation of categorical variables as binary vectors. This first requires that the categorical values be mapped to integer values. Then, each integer value is represented as a binary vector that is all zero values except the index of the integer, which is marked with a 1.

In [64]:
```
# one hot encoding
df_oneHot = pd.get_dummies(df_multi['Venue Category'], prefix = "", prefi

# add neighborhood column back to dataframe
df_oneHot['Neighborhood'] = df_multi['Neighborhood']

# move neighborhood column to the first column
cols = df_oneHot.columns.tolist()
cols.insert(0, cols.pop(cols.index('Neighborhood')))
cols.insert(1, cols.pop(cols.index('Indian Restaurant')))
df_oneHot = df_oneHot.reindex(columns = cols)
df_oneHot.columns
```

Out[64]:
```
Index(['Neighborhood', 'Indian Restaurant', 'Afghan Restaurant',
       'American Restaurant', 'Aquarium', 'Arts & Crafts Store',
       'Asian Restaurant', 'Athletics & Sports', 'Auto Dealership',
       'Automotive Shop',
       ...
       'Turkish Restaurant', 'Vegetarian / Vegan Restaurant',
       'Video Game Store', 'Vietnamese Restaurant', 'Warehouse Store',
       'Wings Joint', 'Women's Store', 'Yoga Studio', 'Zoo', 'Zoo Exhi
bit'],
      dtype='object', length=187)
```

3.2 Exploratory Data Analysis per Neighborhoods

3.2.1 Statistical spread of "Indian Restaurants" per Neighborhoods

Generally speaking, South Asian restaurants are Indian restaurants 😊. Here we look at the statistical spread of Indian restaurants in the top 20 Indo-Aryan speaking neighborhoods in Toronto.

In [65]:
```
#  Indian Restaurant in the top 20 neighborhoods
df_oneHot[['Neighborhood','Indian Restaurant']].loc[df_oneHot['Indian Res
df_oneHot[['Neighborhood','Indian Restaurant']].groupby('Neighborhood').m
```

Out[65]:

Neighborhood	Indian Restaurant
Mount Olive-Silverstone-Jamestown	0.098039
Thorncliffe Park	0.040404
Woburn	0.035088
L'Amoreaux	0.031579
Flemingdon Park	0.030000
Bendale	0.022222
Scarborough Village	0.014085
Clairlea-Birchmount	0.010000
Taylor-Massey	0.000000
West Humber-Clairville	0.000000
West Hill	0.000000
Waterfront Communities-The Island	0.000000
Rouge	0.000000
Tam O'Shanter-Sullivan	0.000000
Parkwoods-Donalda	0.000000
Malvern	0.000000
Kingsview Village-The Westway	0.000000
Humbermede	0.000000
Humber Summit	0.000000
York University Heights	0.000000

3.2.2 Statistical Spread of Venues per Neighborhood

Display the statistical spread of venues per neighborhoods.

In [71]:
```
df_oneHot_grp = df_oneHot.groupby('Neighborhood').mean().reset_index()
df_oneHot_grp
```

Out[71]:

	Neighborhood	Indian Restaurant	Afghan Restaurant	American Restaurant	Aquarium	Arts & Crafts Store	Asian Restaurant	Athletics & Sports
0	Bendale	0.022222	0.000000	0.022222	0.00	0.022222	0.022222	0.00000
1	Clairlea-Birchmount	0.010000	0.000000	0.010000	0.00	0.010000	0.000000	0.00000
2	Flemingdon Park	0.030000	0.020000	0.020000	0.00	0.000000	0.020000	0.00000
3	Humber Summit	0.000000	0.000000	0.000000	0.00	0.000000	0.050000	0.00000
4	Humbermede	0.000000	0.000000	0.000000	0.00	0.000000	0.083333	0.00000
5	Kingsview Village-The Westway	0.000000	0.000000	0.040816	0.00	0.000000	0.000000	0.02040
6	L'Amoreaux	0.031579	0.000000	0.000000	0.00	0.000000	0.021053	0.00000
7	Malvern	0.000000	0.000000	0.000000	0.00	0.000000	0.000000	0.00000
8	Mount Olive-Silverstone-Jamestown	0.098039	0.000000	0.000000	0.00	0.000000	0.000000	0.00000
9	Parkwoods-Donalda	0.000000	0.000000	0.010417	0.00	0.000000	0.031250	0.00000
10	Rouge	0.000000	0.000000	0.000000	0.00	0.000000	0.000000	0.00000
11	Scarborough Village	0.014085	0.000000	0.000000	0.00	0.000000	0.000000	0.00000
12	Tam O'Shanter-Sullivan	0.000000	0.000000	0.000000	0.00	0.012195	0.012195	0.00000
13	Taylor-Massey	0.000000	0.000000	0.020000	0.00	0.000000	0.000000	0.01000
14	Thorncliffe Park	0.040404	0.020202	0.000000	0.00	0.000000	0.000000	0.01010
15	Waterfront Communities-The Island	0.000000	0.000000	0.020000	0.02	0.000000	0.000000	0.00000
16	West Hill	0.000000	0.000000	0.000000	0.00	0.000000	0.029412	0.02941
17	West Humber-Clairville	0.000000	0.029412	0.000000	0.00	0.000000	0.058824	0.00000
18	Woburn	0.035088	0.000000	0.000000	0.00	0.000000	0.000000	0.01754
19	York University Heights	0.000000	0.000000	0.000000	0.00	0.012658	0.000000	0.00000

20 rows × 187 columns

3.2.3 Statistical Spread of Top 10 Venues per Neighborhood

Display the statistical spread of the top 10 venues per neighborhood.

In [75]:
```
#  Group the top venues for each neighborhood
num_top_venues = 10   #   Number of top venues needed

for nhood in df_oneHot_grp['Neighborhood']:
    print("\n-----  "+nhood.upper()+"  -----")
    nhood_temp = df_oneHot_grp[df_oneHot_grp['Neighborhood'] == nhood].T.
    nhood_temp.columns = ['venue', 'freq']
    nhood_temp = nhood_temp.iloc[1:]
    nhood_temp['freq'] = nhood_temp['freq'].astype(float)
    nhood_temp = nhood_temp.round({'freq': 4})
    print(nhood_temp.sort_values('freq', ascending = False).reset_index(d
```

```
-----  BENDALE  -----
                   venue    freq
0            Coffee Shop  0.1333
1    Fast Food Restaurant  0.0444
2          Grocery Store  0.0444
3         Sandwich Place  0.0444
4     Chinese Restaurant  0.0444
5      Indian Restaurant  0.0222
6           Burger Joint  0.0222
7          Burrito Place  0.0222
8            Supermarket  0.0222
9   Sri Lankan Restaurant  0.0222

-----  CLAIRLEA-BIRCHMOUNT  -----
                  venue  freq
0           Coffee Shop  0.10
1  Fast Food Restaurant  0.09
2        Sandwich Place  0.06
3          Burger Joint  0.04
4        Clothing Store  0.04
5                  Bank  0.03
6         Grocery Store  0.03
7          Pizza Place  0.03
8           Supermarket  0.03
9   Japanese Restaurant  0.02

-----  FLEMINGDON PARK  -----
                 venue  freq
0          Coffee Shop  0.07
1       Sandwich Place  0.05
2  Japanese Restaurant  0.04
3                  Gym  0.04
4    Indian Restaurant  0.03
5           Restaurant  0.03
6             Pharmacy  0.03
7        Grocery Store  0.03
8                 Park  0.03
9         Burger Joint  0.03

-----  HUMBER SUMMIT  -----
                   venue   freq
0            Coffee Shop  0.100
1                   Park  0.100
```

3.2.4 Create Dataframe with Top 10 Venues per Neighborhood

Consolidate the top 10 most common venues per neighborhood into a dataframe.

In [76]:
```
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[
    except:
        columns.append('{}th Most Common Venue'.format(ind+1))



# create a new dataframe
df_hoods_venues_sorted = pd.DataFrame(columns=columns)
df_hoods_venues_sorted['Neighborhood'] = df_oneHot_grp['Neighborhood']

for ind in np.arange(df_oneHot_grp.shape[0]):
    df_hoods_venues_sorted.iloc[ind, 1:] = return_most_common_venues(df_o

#  Display the results
df_hoods_venues_sorted
```

Out[76]:

	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 Mo Comm Ver
0	Bendale	Coffee Shop	Grocery Store	Sandwich Place	Chinese Restaurant	Fast Food Restaurant	Indian Restaurant	Dog F
1	Clairlea-Birchmount	Coffee Shop	Fast Food Restaurant	Sandwich Place	Burger Joint	Clothing Store	Pizza Place	Supermar
2	Flemingdon Park	Coffee Shop	Sandwich Place	Japanese Restaurant	Gym	Indian Restaurant	Burger Joint	P
3	Humber Summit	Park	Coffee Shop	Italian Restaurant	Pharmacy	Clothing Store	Fast Food Restaurant	As Restaur
4	Humbermede	Coffee Shop	Park	Golf Course	Nightclub	Sandwich Place	Café	Supermar
5	Kingsview Village-The Westway	Coffee Shop	Pharmacy	Pizza Place	Sandwich Place	Restaurant	American Restaurant	Wings Jo
6	L'Amoreaux	Chinese Restaurant	Coffee Shop	Park	Pizza Place	Bakery	Indian Restaurant	Dess Sh
7	Malvern	Zoo Exhibit	Fast Food Restaurant	Zoo	Grocery Store	Pizza Place	Spa	Hobby Sh
	Mount Olive-							

3.3 Modeling

Now we analyze the data that we have compiled and explored and see if we can gleam an understanding that can be presented to the clients. Our best approach is to cluster these neighborhoods and zoom into the locations that the clients can consider.

We have been looking at neighborhoods where there is a high population of Indo-Aryan speakers, and not looking at areas with a low Indo-Aryan population. So for clustering, our best approach is to use K-Means Clustering.

3.3.1 K-Means Clustering

K-Means is a type of partitioning clustering, that is, it divides the data into K non-overlapping subsets or clusters without any cluster internal structure or labels. Objects within a cluster are very similar, and objects across different clusters are very different or dissimilar.

K-Means is heuristic algorithm, there is no guarantee that it will converge to the global optimum, but the result may be a local optimum. The algorithm can be run multiple times in order to get better outcomes.

Import Libraries for KMeans and set-up the dataframe

In [84]:
```
from sklearn.cluster import KMeans
#  Create data frame for K-Means
df_oneHot_grp_cluster = df_oneHot_grp.drop(columns='Neighborhood')
```

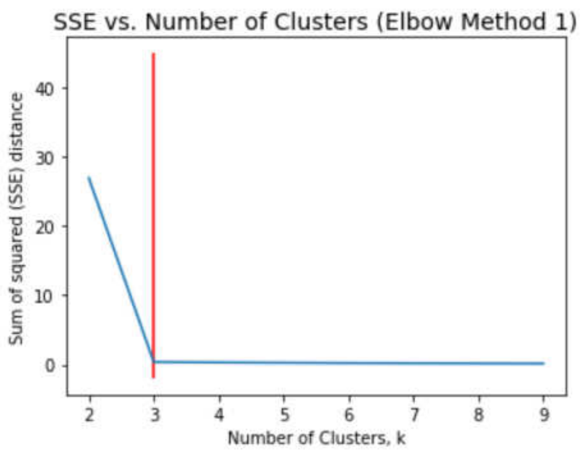
Elbow Method - determine optimal cluster number

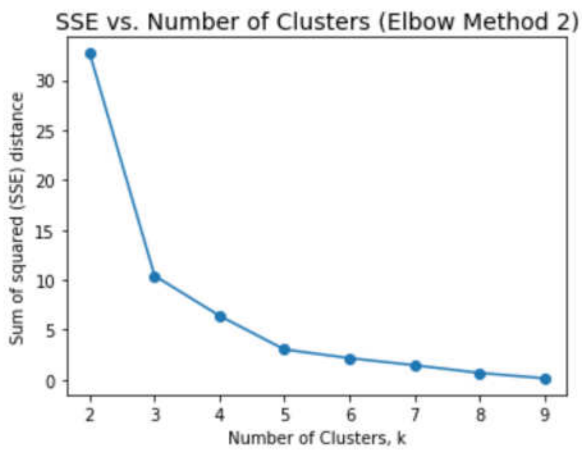
The Elbow method is a method of interpretation and validation of consistency within cluster analysis designed to help finding the appropriate number of clusters in a dataset.

Since centroids are chosen randomly, the model will have high error. Clusters can be reshaped in such a way that the total distance of all members of a cluster from its centroid can be minimized using the **Sum of Square Error (SSE)** to create better clusters with less error.

In order to find the optimal number of clusters, we plot the curve of SSE according to the number of clusters k. The location of a bend (knee) in the plot is considered as an indicator of the appropriate number of clusters.

In [88]:
```
#  ELBOW
# 1
# SSE is initialize with empty values
sse = {}

for num_cluster in range(2, 10):
    kmeans1 = KMeans(n_clusters = num_cluster, max_iter = 500).fit(df_one
    df_oneHot_grp_cluster["clusters"] = kmeans1.labels_
    sse[num_cluster] = kmeans1.inertia_

fig = plt.figure(figsize=(12,4))
fig.add_subplot(121)
#plt.figure()
plt.plot(list(sse.keys()), list(sse.values()))
plt.xlabel(r'Number of Clusters, k')
plt.ylabel(r'Sum of squared (SSE) distance')
plt.title(r'SSE vs. Number of Clusters (Elbow Method 1)', fontsize=14)
plt.vlines(3, ymin = -2, ymax = 45, colors = 'red')


# 2
# SSE is initialize with empty values
sse = []

for k in range(2, 10):
    kmeans2 = KMeans(n_clusters=k, max_iter = 500).fit(df_oneHot_grp_clus
    sse.append(kmeans2.inertia_)


fig.add_subplot(122)
#plt.figure()
plt.plot(list(range(2, 10)), sse, '-o')
plt.xlabel(r'Number of Clusters, k')
plt.ylabel(r'Sum of squared (SSE) distance')
plt.title(r'SSE vs. Number of Clusters (Elbow Method 2)', fontsize=14)
plt.show()
```





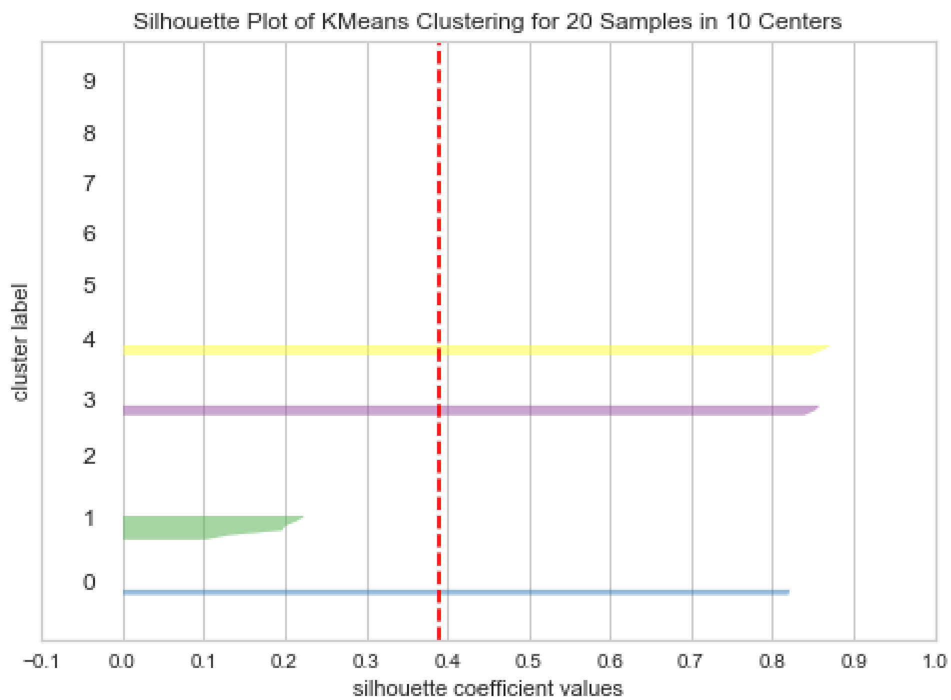
Elbow #1: Clearly shows that the optimal number of cluster shoud be **3**. Elbow #2: Shows that the kink (elbow) is most pronounced at k=3

So we going to go with **3** clusters.

Silhouette Coefficient

The silhouette value is a measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation). The silhouette ranges from −1 to +1, where a high value indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters.

In [89]:
```
from yellowbrick.cluster import SilhouetteVisualizer

model = KMeans(n_clusters = 10, random_state=0)
visualizer = SilhouetteVisualizer(model)

visualizer.fit(df_oneHot_grp_cluster)
visualizer.poof() # Draw/show/poof the data
```



We see that cluster numbers 0, 3 and 4 are the best. We can discard "0". "4" can be considered, but based on the elbow method, **3** cluster is still the optimal choice.

K-Means with Cluster Number = 3

Run K-Means and add the cluster numbers to the dataframe.

In [93]:
```
#  K-Means
clusterNum = 3
kmeans = KMeans(n_clusters = clusterNum, random_state=0).fit(df_oneHot_gr
#kmeans.labels_
#kmeans.cluster_centers_.head

#  Add clustering labels to dataframe
df_hoods_venues_sorted.insert(0, 'Cluster', kmeans.labels_)
df_hoods_venues_sorted.head()
```

Out[93]:

	Cluster	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	
0	1	Bendale	Coffee Shop	Grocery Store	Sandwich Place	Chinese Restaurant	Fast Food Restaurant	Indian Restaurant	
1	1	Clairlea-Birchmount	Coffee Shop	Fast Food Restaurant	Sandwich Place	Burger Joint	Clothing Store	Pizza Place	Su
2	1	Flemingdon Park	Coffee Shop	Sandwich Place	Japanese Restaurant	Gym	Indian Restaurant	Burger Joint	
3	0	Humber Summit	Park	Coffee Shop	Italian Restaurant	Pharmacy	Clothing Store	Fast Food Restaurant	I
4	0	Humbermede	Coffee Shop	Park	Golf Course	Nightclub	Sandwich Place	Café	Su

3.3.2 Print out the Clusters

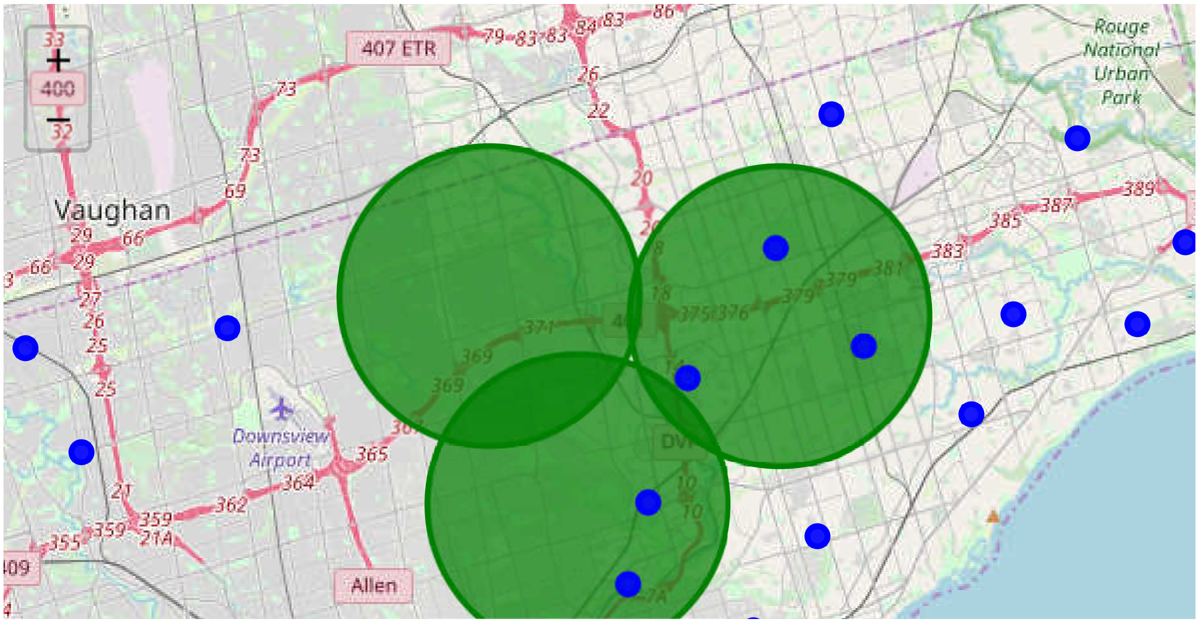
In [96]:
```
for i in np.arange(0,clusterNum):
    print("\n\n---  CLUSTER NUMBER: {}  ---\n{}".format(i,df_hoods_venues
```

```
---  CLUSTER NUMBER: 0  ---
    Cluster             Neighborhood 1st Most Common Venue  \
3         0            Humber Summit                  Park
4         0                Humbermede           Coffee Shop
7         0                   Malvern           Zoo Exhibit
12        0  Tam O'Shanter-Sullivan    Chinese Restaurant

   2nd Most Common Venue 3rd Most Common Venue 4th Most Common Venue
\
3            Coffee Shop    Italian Restaurant              Pharmacy
4                   Park           Golf Course             Nightclub
7    Fast Food Restaurant                  Zoo         Grocery Store
12           Coffee Shop  Fast Food Restaurant                  Park

   5th Most Common Venue 6th Most Common Venue 7th Most Common Venue
\
3         Clothing Store  Fast Food Restaurant      Asian Restaurant
4         Sandwich Place                  Café           Supermarket
7            Pizza Place                   Spa           Hobby Shop
12                  Bank              Pharmacy        Sandwich Place

   8th Most Common Venue    9th Most Common Venue 10th Most Common Ven
ue
3                   Bank           Sandwich Place           Shopping Ma
ll
4        Thai Restaurant                 Pharmacy       Asian Restaura
nt
7              Gift Shop  Fruit & Vegetable Store           Liquor Sto
re
12  Cantonese Restaurant    Vietnamese Restaurant           Liquor Sto
re


---  CLUSTER NUMBER: 1  ---
    Cluster                          Neighborhood 1st Most Common Venue
\
0         1                              Bendale           Coffee Shop
1         1                    Clairlea-Birchmount           Coffee Shop
2         1                        Flemingdon Park           Coffee Shop
5         1       Kingsview Village-The Westway           Coffee Shop
9         1                    Parkwoods-Donalda           Coffee Shop
10        1                                Rouge           Coffee Shop
13        1                         Taylor-Massey           Coffee Shop
14        1                      Thorncliffe Park           Coffee Shop
15        1  Waterfront Communities-The Island           Coffee Shop
17        1                  West Humber-Clairville           Coffee Shop
19        1                York University Heights           Coffee Shop

   2nd Most Common Venue 3rd Most Common Venue 4th Most Common Venue
\
0          Grocery Store        Sandwich Place    Chinese Restaurant
```

3.3.3 Folium Map of the Clusters

In [97]:
```
df_hoods_venues_sorted_location = df_hood.join(df_hoods_venues_sorted.set

#  For Folium
df_latlng_clust = df_hoods_venues_sorted_location[['Neighborhood','Latitu
df_latlng_clust = df_latlng_clust.reset_index()#('Cluster')


cluster_map = folium.Map(location=[df_latlng_clust.Latitude.mean(), df_la

# Plot the top 20 IndoAryan speaking neighborhoods
for lat, lng in zip(df_latlng_clust.Latitude, df_latlng_clust.Longitude):
    folium.CircleMarker(
            [lat, lng],
            radius=75,
            #popup=label,   #   clusters
            fill=True,
            color='green',
            fill_color='green',
            fill_opacity=0.7
            ).add_to(cluster_map)

# add points for all the neighborhoods
for lat, lng, label in zip(df_hood.Latitude, df_hood.Longitude, df_hood.N
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
            [lat, lng],
            radius=5,
            popup=label,  #  neighborhoods
            fill=True,
            color='blue',
            fill_color='blue',
            fill_opacity=0.9
            ).add_to(cluster_map)


# display map
cluster_map
```

Out[97]:



4. Results and Discussion

All three cluster have quite a few eating places in their top 10 common venues: coffee shops, fast food/sandwich places, Asian and Italian restaurants are very popular. The populations for these neighborhoods are very open to eating out and this bodes very well to the idea of investing in a South Asian restaurant. The three cluster are also in close proximity to each other as well to neighborhoods with a higher Indo-Aryan speaking population, which is very good news for our investors.

Of course, for this class project we did not do a comprehensive analysis involving real-estate prices, renting/leasing/buying, crime statistics, etc. After all, I do want to finish this certification this week and not next year 😊

5. Conclusion

Our analysis showed that the three clusters selected for opening a restaurant have very a fairly high chance of success, considering the out-going population and proximity to other neighborhoods.

Thank you for reviewing!!