Introduction:

The purpose of this Project is to help people in exploring better facilities around their neighborhood. It will help people making smart and efficient decision on selecting great neighborhood out of numbers of other neighborhoods in Scarborough, Toranto.

Lots of people are migrating to various states of Canada and needed lots of research for good housing prices and reputated schools for their children. This project is for those people who are looking for better neighborhoods. For ease of accessing to Cafe, School, Super market, medical shops, grocery shops, mall, theatre, hospital, like minded people, etc.

This Project aim to create an analysis of features for a people migrating to Scarborough to search a best neighborhood as a comparative analysis between neighborhoods. The features include median housing price and better school according to ratings, crime rates of that particular area, road connectivity, weather conditions, good management for emergency, water resources both freash and waste water and excrement conveyed in sewers and recreational facilities.

It will help people to get awareness of the area and neighborhood before moving to a new city, state, country or place for their work or to start a new fresh life.

The Business Problem To Solve

When deciding where to move, people tend to want to move to places that have facilities they will enjoy. The purpose of this project will be to categorize the neighbohoods in Toronto, Canada based on on the types of facilities nearby.

Using the data of the the neighbohoods in Toronto, the goal is to provide a tool used to make an educated decision on where to move for new job opportunities.

Data Description:

Data Link: https://en.wikipedia.org/wiki/List_of_postal_codes_of_Canada:_M

Will use Scarborough dataset which we scrapped from wikipedia on Week 3. Dataset consisting of latitude and longitude, zip codes.

Foursquare API Data:

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

After finding the list of neighborhoods, we then connect to the Foursquare API to gather information about venues inside each and every neighborhood. For each neighborhood, we have chosen the radius to be 100 meter.

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

1. Neighborhood
2. Neighborhood Latitude
3. Neighborhood Longitude
4. Venue
5. Name of the venue e.g. the name of a store or restaurant
6. Venue Latitude
7. Venue Longitude
8. Venue Category

Tools Used First we look to see what tools were needed to do the project. We wanted to use maps to show locations as well as use several libraries that allow for data extraction, manipulation and display.

In [1]:
```
import pandas as pd
import numpy as np
import requests
import geocoder
from geopy.geocoders import Nominatim
import folium
import json
from pandas.io.json import json_normalize
from sklearn.cluster import KMeans
import matplotlib.cm as cm
import matplotlib.colors as colors
```

This is where we extracted the data for later use and manipulation. As you see below, we removed rows in which data was not assigned and therfore not useful. Then we imported the Geospatial data to use together along with the wikidata.

In [2]:
```
wiki = 'https://en.wikipedia.org/wiki/List_of_postal_codes_of_Canada:_M'
wiki_page = requests.get(wiki)

wiki_raw = pd.read_html(wiki_page.content, header = 0)[0]
df=wiki_raw
df = df[df['Neighbourhood'] != 'Not assigned']
df = df[df['Borough'] != 'Not assigned']
df.reset_index(drop = True, inplace = True)
url = 'http://cocl.us/Geospatial_data'
df_geo = pd.read_csv(url)
df_geo.reset_index(drop = True, inplace = True)
df = df.join(df_geo.set_index('Postal Code'), on='Postal Code')
address = 'Toronto, Ontario'
geolocator = Nominatim(user_agent="toronto_explorer")
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude

df
df.head()
```

Out[2]:

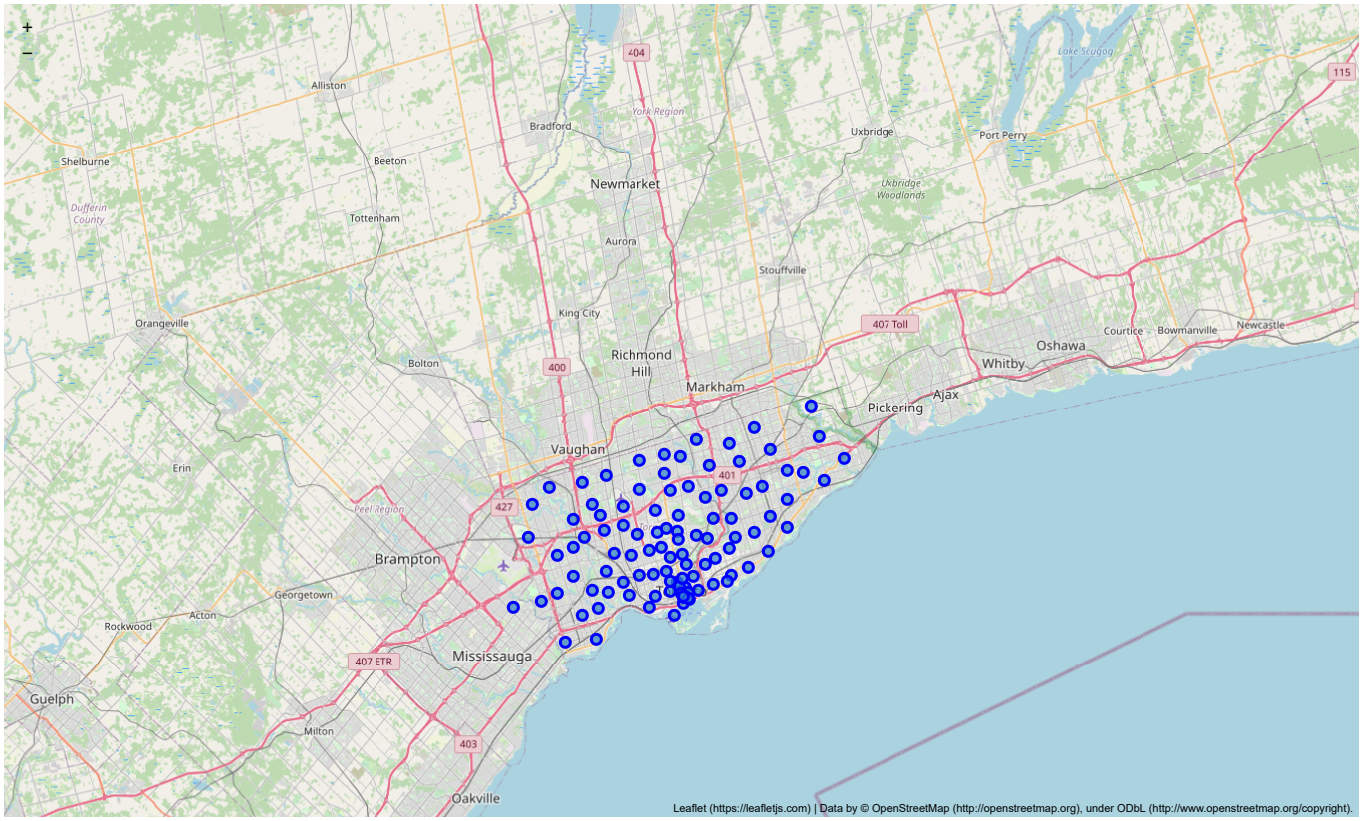
	Postal Code	Borough	Neighbourhood	Latitude	Longitude
0	M3A	North York	Parkwoods	43.753259	-79.329656
1	M4A	North York	Victoria Village	43.725882	-79.315572
2	M5A	Downtown Toronto	Regent Park, Harbourfront	43.654260	-79.360636
3	M6A	North York	Lawrence Manor, Lawrence Heights	43.718518	-79.464763
4	M7A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

Next, we created a map that shows the distribution of the neighborhoods to make it easier to see where they are located.

In [3]:
```
# create map of Toronto using latitude and longitude values
map_Too = folium.Map(location=[latitude, longitude], zoom_start=10)

# add markers to map
for lat, lng, borough, neighborhood in zip(df['Latitude'], df['Longitude'], df['Borough'], df['Neighbourhood']):
    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,
        ).add_to(map_Too)

map_Too
```

Out[3]:



Added in the Foursquare API Creditials for later use in the project.

In [4]:
```
# @hidden_cell
CLIENT_ID = 'YKI35K3RM3SI1PEJM4AKOYS51SRXMRQIAGWJKMMPUBMIIHZ5' # your Foursquare ID
CLIENT_SECRET = 'RTISQNAKGBAVHHLH0YV5DCHVFNOPDHHQ3SYYLS5YIPTDYRHS' # your Foursquare Secret
VERSION = '20180604' # Foursquare API version
```

Here we get the latitude and longitude of our target neighborhood.

In [5]:
```
neighborhood_latitude = df.loc[0, 'Latitude'] # neighborhood latitude value
neighborhood_longitude = df.loc[0, 'Longitude'] # neighborhood longitude value

neighborhood_name = df.loc[0, 'Neighbourhood'] # neighborhood name

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

```
Latitude and longitude values of Parkwoods are 43.7532586, -79.3296565.
```

Then we do a search around that neighborhood for venues.

In [6]:
```
LIMIT = 100
radius = 500
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
```

Out[6]:
```
'https://api.foursquare.com/v2/venues/explore?&client_id=YKI35K3RM3SI1PEJM4AKOYS51SRXMRQIAGWJKMMPUBMIIHZ5&client_secret=RTISQNAKGBAVHHLH0YV5DCHVFNOPDHHQ3SYYLS5YIPTDYRHS&v=20180604&ll=43.753258
6,-79.3296565&radius=500&limit=100'
```

In [7]:
```
results = requests.get(url).json()
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.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(".")[-1] for col in nearby_venues.columns]

nearby_venues.head()
```

```
<ipython-input-7-1d6617dc71cb>:16: FutureWarning: pandas.io.json.json_normalize is deprecated, use pandas.json_normalize instead
  nearby_venues = json_normalize(venues) # flatten JSON
```

Out[7]:

	name	categories	lat	lng
0	Brookbanks Park	Park	43.751976	-79.332140
1	Variety Store	Food & Drink Shop	43.751974	-79.333114

Here, we get nearby Venues for the location we chose.

In [8]:
```
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 = ['Neighbourhood',
                  'Neighborhood Latitude',
                  'Neighborhood Longitude',
                  'Venue',
                  'Venue Latitude',
                  'Venue Longitude',
                  'Venue Category']

    return(nearby_venues)
```

Then we list out the venues that are found in the nearby call.

In [9]:
```
toronto_venues = getNearbyVenues(names=df['Neighbourhood'],
                                 latitudes=df['Latitude'],
                                 longitudes=df['Longitude']
                                )
```

```
Parkwoods
Victoria Village
Regent Park, Harbourfront
Lawrence Manor, Lawrence Heights
Queen's Park, Ontario Provincial Government
Islington Avenue, Humber Valley Village
Malvern, Rouge
Don Mills
Parkview Hill, Woodbine Gardens
Garden District, Ryerson
Glencairn
West Deane Park, Princess Gardens, Martin Grove, Islington, Cloverdale
Rouge Hill, Port Union, Highland Creek
Don Mills
Woodbine Heights
St. James Town
Humewood-Cedarvale
Eringate, Bloordale Gardens, Old Burnhamthorpe, Markland Wood
Guildwood, Morningside, West Hill
The Beaches
Berczy Park
Caledonia-Fairbanks
Woburn
Leaside
Central Bay Street
Christie
Cedarbrae
Hillcrest Village
Bathurst Manor, Wilson Heights, Downsview North
Thorncliffe Park
Richmond, Adelaide, King
Dufferin, Dovercourt Village
Scarborough Village
Fairview, Henry Farm, Oriole
Northwood Park, York University
East Toronto, Broadview North (Old East York)
Harbourfront East, Union Station, Toronto Islands
Little Portugal, Trinity
Kennedy Park, Ionview, East Birchmount Park
Bayview Village
Downsview
The Danforth West, Riverdale
Toronto Dominion Centre, Design Exchange
Brockton, Parkdale Village, Exhibition Place
Golden Mile, Clairlea, Oakridge
York Mills, Silver Hills
Downsview
India Bazaar, The Beaches West
Commerce Court, Victoria Hotel
North Park, Maple Leaf Park, Upwood Park
Humber Summit
Cliffside, Cliffcrest, Scarborough Village West
Willowdale, Newtonbrook
Downsview
Studio District
Bedford Park, Lawrence Manor East
Del Ray, Mount Dennis, Keelsdale and Silverthorn
Humberlea, Emery
Birch Cliff, Cliffside West
Willowdale, Willowdale East
Downsview
Lawrence Park
Roselawn
Runnymede, The Junction North
Weston
Dorset Park, Wexford Heights, Scarborough Town Centre
York Mills West
Davisville North
```

```
Forest Hill North & West, Forest Hill Road Park
High Park, The Junction South
Westmount
Wexford, Maryvale
Willowdale, Willowdale West
North Toronto West, Lawrence Park
The Annex, North Midtown, Yorkville
Parkdale, Roncesvalles
Canada Post Gateway Processing Centre
Kingsview Village, St. Phillips, Martin Grove Gardens, Richview Gardens
Agincourt
Davisville
University of Toronto, Harbord
Runnymede, Swansea
Clarks Corners, Tam O'Shanter, Sullivan
Moore Park, Summerhill East
Kensington Market, Chinatown, Grange Park
Milliken, Agincourt North, Steeles East, L'Amoreaux East
Summerhill West, Rathnelly, South Hill, Forest Hill SE, Deer Park
CN Tower, King and Spadina, Railway Lands, Harbourfront West, Bathurst Quay, South Niagara, Island airport
New Toronto, Mimico South, Humber Bay Shores
South Steeles, Silverstone, Humbergate, Jamestown, Mount Olive, Beaumond Heights, Thistletown, Albion Gardens
Steeles West, L'Amoreaux West
Rosedale
Stn A PO Boxes
Alderwood, Long Branch
Northwest, West Humber - Clairville
Upper Rouge
St. James Town, Cabbagetown
First Canadian Place, Underground city
The Kingsway, Montgomery Road, Old Mill North
Church and Wellesley
Business reply mail Processing Centre, South Central Letter Processing Plant Toronto
Old Mill South, King's Mill Park, Sunnylea, Humber Bay, Mimico NE, The Queensway East, Royal York South East, Kingsway Park South East
Mimico NW, The Queensway West, South of Bloor, Kingsway Park South West, Royal York South West
```

Then we list the potential nearby venues by various charactaristics for sorting.

In [10]:
```
# one hot encoding
toronto_onehot = pd.get_dummies(toronto_venues[['Venue Category']],
                                prefix="",
                                prefix_sep="")

# add neighborhood column back to dataframe
toronto_onehot['Neighbourhood'] = toronto_venues['Neighbourhood']

# move neighborhood column to the first column
fixed_columns = [toronto_onehot.columns[-1]] + list(toronto_onehot.columns[:-1])
toronto_onehot = toronto_onehot[fixed_columns]
toronto_grouped = toronto_onehot.groupby('Neighbourhood').mean().reset_index()
toronto_onehot.head()
```

Out[10]:

	Neighbourhood	Accessories Store	Adult Boutique	Airport	Airport Food Court	Airport Gate	Airport Lounge	Airport Service	Airport Terminal	American Restaurant	...	Vegetarian / Vegan Restaurant	Video Game Store	Video Store	Vietnamese Restaurant	Warehouse Store	Wine Bar	Wine Shop	Wings Joint	Women's Store	Yoga Studio
0	Parkwoods	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	Parkwoods	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	Victoria Village	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	Victoria Village	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	Victoria Village	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

5 rows × 276 columns

Now we sorted the venues by the most common in each neighborhood.

In [11]:
```
def return_most_common_venues(row, num_top_venues):
    row_categories = row.iloc[1:]
    row_categories_sorted = row_categories.sort_values(ascending=False)

    return row_categories_sorted.index.values[0:num_top_venues]

num_top_venues = 10

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

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

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

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

neighborhoods_venues_sorted.head()
```

Out[11]:

	Neighbourhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue	9th Most Common Venue	10th Most Common Venue
0	Agincourt	Lounge	Latin American Restaurant	Breakfast Spot	Skating Rink	Event Space	Falafel Restaurant	Farm	Ethiopian Restaurant	Discount Store	Escape Room
1	Alderwood, Long Branch	Pizza Place	Coffee Shop	Sandwich Place	Pub	Skating Rink	Gym	Electronics Store	Escape Room	Eastern European Restaurant	Drugstore
2	Bathurst Manor, Wilson Heights, Downsview North	Coffee Shop	Bank	Pet Store	Fried Chicken Joint	Shopping Mall	Bridal Shop	Sandwich Place	Diner	Supermarket	Sushi Restaurant
3	Bayview Village	Café	Japanese Restaurant	Bank	Chinese Restaurant	Discount Store	Dog Run	Doner Restaurant	Donut Shop	Drugstore	Eastern European Restaurant
4	Bedford Park, Lawrence Manor East	Coffee Shop	Italian Restaurant	Sandwich Place	Butcher	Sushi Restaurant	Juice Bar	Spa	Liquor Store	Restaurant	Thai Restaurant

Now we generated clusters to train and test our model and add cluster labels to our model.

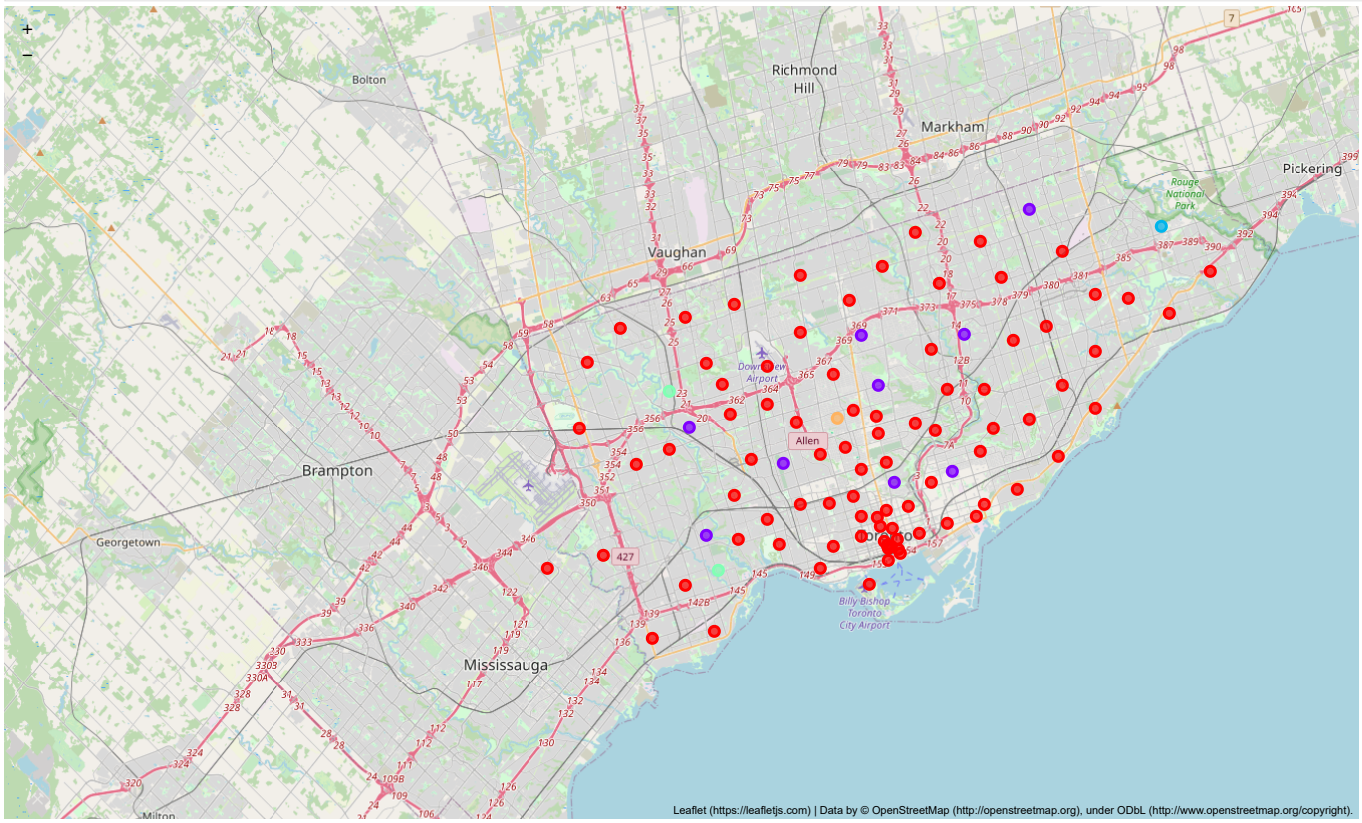
In [12]:
```
# set number of clusters
kclusters = 5

toronto_grouped_clustering = toronto_grouped.drop('Neighbourhood', 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]

# add clustering labels
neighborhoods_venues_sorted.insert(0, 'ClusterLabels', kmeans.labels_)

toronto_merged = df

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

We now create a cluster map to show our data visually and highlight the neighrbhoods based on similarities.

In [13]:
```
# 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]

toronto_merged_nonan = toronto_merged.dropna(subset=['ClusterLabels'])

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

map_clusters
```

Out[13]:



What the data shows is that there are plenty of neighborhoods that have similar qualities to our preferred characteristics. Futher, it shows that there some of the preferred neighborhoods are nearby to our target location. As such, we can use this knowledge in deciding where people should live when they relocate.

In [ ]: