# Final_Assignment_3

June 2, 2019

Lets cluster Toronto's Neighbourhoods!

<p style="font-size:70%">I read from csv-file.<br> I loop over coordinates and find venues with explore.<br>Below

In [5]:
```
import csv
import pandas as pd
import numpy as np #Work with vectors
import requests #Reach an url
import json #Read data from foursquare
from geopy.geocoders import Nominatim #Use latitude and longitude coord.
from pandas.io.json import json_normalize # Create dataframes with Jason data
```

In [6]:
```
post_can = pd.read_csv("lat_lon_can.csv")# Get my dataframe
post_can.shape
```

Out[6]: (103, 5)

My credentials won't appear in the pushed notebook, but I initialized them them here.

In [7]:
```
CLIENT_ID = '4VXMEQRGF5NKR4EJXXCXAG2U2EFO45FOX2VMOZ03WKYTU5F4' # your Foursc
CLIENT_SECRET = 'HCURSXMR4EYJA3NBHXOL0FQBKDMDXAVHIW5LF03CN5X0I1EV' # your F
VERSION = '20180605' # Foursquare API version
```

Now I go through every postal code and find the venues. I do it like we did for the last lab. I use Postal Code rather than Neighbourhood.

In [ ]:
```
radius=500
LIMIT=100
venues_list=[];

for nam, hood, lat, lon in zip(post_can["Postcode"], post_can["Neighbourhood"], post_can["Latitude"],po
    url = 'https://api.foursquare.com/v2/venues/explore?client_id={}&client_secret={}&v={}&ll={},{}&

   # make the GET request
    results = requests.get(url).json()['response']['groups'][0]['items']
   # return only relevant information for each nearby venue
    venues_list.append([(
            nam,
```

```
                hood,
                lat,
                lon,
                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 = ['Postcode', 'Neighbourhood',
                  'Postal Code Latitude',
                  'Postal Code Longitude',
                  'Venue',
                  'Venue Latitude',
                  'Venue Longitude',
                  'Venue Category']
nearby_venues.shape
```

The next step is to create a dataframe with the categories as columnsWith that we group them by postal code

```
In [27]: print('There are {} uniques categories.'.format(len(nearby_venues['Venue Category'].unique())))
```

There are 282 uniques categories.

```
In [28]: # Again, I used the old lab as reference
    toronto_venues = pd.get_dummies(nearby_venues[['Venue Category']], prefix="", prefix_sep="")
    print(toronto_venues.shape)
    # add neighborhood column back to dataframe
    toronto_venues['Postcode'] = nearby_venues['Postcode']
    #toronto_venues['Neighbourhood'] = nearby_venues['Neighbourhood']
    # move postcode, neighborhood columns to the first and second column
    fixed_columns = [toronto_venues.columns[-2], toronto_venues.columns[-1]] + toronto_venues.columns[:-2]
    toronto_venues = toronto_venues[fixed_columns]
```

(2256, 282)

```
In [29]: toronto_group = toronto_venues.groupby("Postcode").mean().reset_index()
    toronto_group.shape
```

Out[29]: (100, 283)

Clustering
Finally we import KMeans and creat the labels for each cluster

```
In [36]: toronto_cluster=toronto_group.drop("Postcode", 1)
    k=6
    from sklearn.cluster import KMeans
    kmeans = KMeans(n_clusters=k, random_state=0).fit(toronto_cluster)
    len(kmeans.labels_)
```

Out[36]: 100

```
In [37]: toronto = pd.DataFrame(columns=["Postcode", "Cluster Labels", "Latitude", "Longitude"])
         toronto["Cluster Labels"]=kmeans.labels_
         toronto["Postcode"]=toronto_group["Postcode"]
```

Unfortunately, during the "request" process I lost 3 rows (also 3 postcodes) from original dataframe (post_can), so the columns do not match and I cannot merge. I look for coordinates in the file.

```
In [38]: geo_data=pd.read_csv("Geospatial_Coordinates.csv")
         for geo, lat, lon in zip(geo_data["Postal Code"],geo_data["Latitude"],geo_data["Longitude"]):
             toronto.loc[toronto.Postcode == geo, "Latitude"]=lat
             toronto.loc[toronto.Postcode == geo, "Longitude"]=lon
         toronto.head()
```

Out[38]:

| | Postcode | Cluster Labels | Latitude | Longitude |
|---|---|---|---|---|
| 0 | M1B | 4 | 43.8067 | -79.1944 |
| 1 | M1C | 4 | 43.7845 | -79.1605 |
| 2 | M1E | 0 | 43.7636 | -79.1887 |
| 3 | M1G | 0 | 43.771 | -79.2169 |
| 4 | M1H | 4 | 43.7731 | -79.2395 |

Great! Finally a map

```
In [39]: #Colors depending on nr of clusters
         #From last lab
         import matplotlib.cm as cm
         import matplotlib.colors as colors
         x = np.arange(k)
         ys = [i + x + (i*x)**2 for i in range(k)]
         colors_array = cm.rainbow(np.linspace(0, 1, len(ys)))
         rainbow = [colors.rgb2hex(i) for i in colors_array]

         #We import the libraries
         import folium
         from geopy.geocoders import Nominatim

         #We look for toronto
         address = 'Toronto, Canada'

         geolocator = Nominatim(user_agent="can_explorer")
         location = geolocator.geocode(address)
         latitude = location.latitude
         longitude = location.longitude
```

```
In [40]: map_toronto = folium.Map(location=[latitude, longitude], zoom_start=10)
         for lat, lng, label, cluster in zip(toronto['Latitude'], toronto['Longitude'], toronto['Postcode'], toronto["Clus
             label = folium.Popup(str(label) + ' Cluster ' + str(cluster), parse_html=True)
```

```
        folium.CircleMarker(
            [lat, lng],
            radius=5,
            popup=label,
            color=rainbow[cluster-1],
            fill=True,
            fill_color=rainbow[cluster-1],
            fill_opacity=0.7,
            parse_html=False).add_to(map_toronto)
    map_toronto
```

Out[40]: <folium.folium.Map at 0x7f37698530b8>

We see most postcodes, and therefore neighbourhoods, fall into the first cluster(red)