Where should I move?

A moving guidance app based on foursquare and clustering in python

Table of Contents

Introduction/Business Problem

So many people move to an area unfamiliar to them for a job. These moves often occur in a tight time-frame due to schedule pressures with career start dates. This forces people to pick a neighborhood quickly, with limited time to see if it is a good fit. What if there was there was tool that related neighborhoods in a new city to those a person was already familiar with?

In this project, I will assume a user has lived in two regions, and is considering a move to two more. I will create maps coding the regions of both the familiar and unfamiliar places. That way, the user can select a new neighborhood that has properties they like from their old neighborhood.

The intent is for this to be a proof of concept that could be related to arbitrary old and new neighborhoods, extending the use of this tool to a broadly marketable application service

Setup

In [82]:
```
import numpy as np # library to handle data in a vectorized manner
import pandas as pd # library for data analsysis
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
import json # library to handle JSON files
import requests # library to handle requests
from pandas.io.json import json_normalize # tranform JSON file into a pandas d
ataframe
# 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
import folium # map rendering library
```

In [83]:
```
# define a helpful foursquare looper function
def getNearbyVenues(names, latitudes, longitudes, radius=500):

    LIMIT = 100
    radius = 500

    venues_list=[]

    print_count = 0
    for name, lat, lng in zip(names, latitudes, longitudes):

        if print_count <= 5:
            print(name, end=', ')
            print_count = print_count + 1
        else:
            print(name + '.')
            print_count = 0

        # create the API request URL
        url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&clie
nt_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)
```

Data

For this example, it is assumed that a person grew up in Toronto, then moved to Manhattan and is now considering a move to either Queens or Staten Island.

The New York area data comes from the Week 3's lab. I have stored the Neighborhoods data, which contains neighborhoods with their latitude, longitude and borough. For Queens and Staten Island, foursquare will need to be used to gather venue information. I have stored the venues data table for Manhattan, which already has venues from foursquare.

The Toronto area data source is Week 3's lab content, which has neighborhood with latitude, longitude and venue.

All of this data will need to be formatted for clustering, including one-hot formatting and label cleaning.

Familiar Area 1: Toronto Data

In [84]:
```
# From week 3 lab
toronto_coordinates_df = pd.read_csv("Geospatial_Coordinates.csv")
toronto_coordinates_df.head()
```

Out[84]:

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

In [85]:
```
# From week 3 lab
toronto_venues_df = pd.read_pickle("toronto_venues.p")
print(f" The shape of the toronto table is: {toronto_venues_df.shape}")
toronto_venues_df.head()
```

```
 The shape of the toronto table is: (2226, 7)
```

Out[85]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Ca
0	Rouge, Malvern	43.806686	-79.194353	Wendy's	43.807448	-79.199056	Fas Res
1	Highland Creek, Rouge Hill, Port Union	43.784535	-79.160497	Royal Canadian Legion	43.782533	-79.163085	Bar
2	Highland Creek, Rouge Hill, Port Union	43.784535	-79.160497	Affordable Toronto Movers	43.787919	-79.162977	Mov Tar
3	Highland Creek, Rouge Hill, Port Union	43.784535	-79.160497	Scarborough Historical Society	43.788755	-79.162438	Hist Mus
4	Guildwood, Morningside, West Hill	43.763573	-79.188711	Swiss Chalet Rotisserie & Grill	43.767697	-79.189914	Pizz Pla

Familiar Area 2: Manhattan Data

In [86]:
```
# From week 3 tutorial
manhattan_venues_df = pd.read_pickle("manhattan_venues.p")
print(f"The shape of the manhattan table is: {manhattan_venues_df.shape}")
manhattan_venues_df.head()
```

```
The shape of the manhattan table is: (3303, 7)
```

Out[86]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Ver Categ
0	Marble Hill	40.876551	-73.91066	Arturo's	40.874412	-73.910271	Pizza Place
1	Marble Hill	40.876551	-73.91066	Bikram Yoga	40.876844	-73.906204	Yoga Studio
2	Marble Hill	40.876551	-73.91066	Tibbett Diner	40.880404	-73.908937	Diner
3	Marble Hill	40.876551	-73.91066	Starbucks	40.877531	-73.905582	Coffee Shop
4	Marble Hill	40.876551	-73.91066	Dunkin'	40.877136	-73.906666	Donut Shop

Unfamiliar Place 1: Queens

In [87]:
```
all_new_york_df = pd.read_pickle("all_new_york_neighborhoods.p")
queens_df = all_new_york_df.loc[all_new_york_df.Borough == 'Queens']
queens_df.head()
```

Out[87]:

	Borough	Neighborhood	Latitude	Longitude
129	Queens	Astoria	40.768509	-73.915654
130	Queens	Woodside	40.746349	-73.901842
131	Queens	Jackson Heights	40.751981	-73.882821
132	Queens	Elmhurst	40.744049	-73.881656
133	Queens	Howard Beach	40.654225	-73.838138

Unfamiliar Place 2: Staten Island

In [88]:
```
staten_island_df = all_new_york_df.loc[all_new_york_df.Borough == 'Staten Isla
nd']
staten_island_df.head()
```

Out[88]:

	Borough	Neighborhood	Latitude	Longitude
197	Staten Island	St. George	40.644982	-74.079353
198	Staten Island	New Brighton	40.640615	-74.087017
199	Staten Island	Stapleton	40.626928	-74.077902
200	Staten Island	Rosebank	40.615305	-74.069805
201	Staten Island	West Brighton	40.631879	-74.107182

Foursquare Time!

In [89]:
```
info_file = "foursquare_info.sec"
info_df = pd.read_csv(info_file)
CLIENT_ID = info_df.ID.values[0] # your Foursquare ID
CLIENT_SECRET = info_df.SECRET.values[0] # your Foursquare Secret
VERSION = '20180605' # Foursquare API version
```

In [90]:
```
queens_venues_df = getNearbyVenues(names=queens_df.Neighborhood,
                                   latitudes=queens_df.Latitude,
                                   longitudes=queens_df.Longitude)
```

```
Astoria, Woodside, Jackson Heights, Elmhurst, Howard Beach, Corona, Forest Hi
lls.
Kew Gardens, Richmond Hill, Flushing, Long Island City, Sunnyside, East Elmhu
rst, Maspeth.
Ridgewood, Glendale, Rego Park, Woodhaven, Ozone Park, South Ozone Park, Coll
ege Point.
Whitestone, Bayside, Auburndale, Little Neck, Douglaston, Glen Oaks, Belleros
e.
Kew Gardens Hills, Fresh Meadows, Briarwood, Jamaica Center, Oakland Gardens,
Queens Village, Hollis.
South Jamaica, St. Albans, Rochdale, Springfield Gardens, Cambria Heights, Ro
sedale, Far Rockaway.
Broad Channel, Breezy Point, Steinway, Beechhurst, Bay Terrace, Edgemere, Arv
erne.
Rockaway Beach, Neponsit, Murray Hill, Floral Park, Holliswood, Jamaica Estat
es, Queensboro Hill.
Hillcrest, Ravenswood, Lindenwood, Laurelton, Lefrak City, Belle Harbor, Rock
away Park.
Somerville, Brookville, Bellaire, North Corona, Forest Hills Gardens, Jamaica
Hills, Utopia.
Pomonok, Astoria Heights, Hunters Point, Sunnyside Gardens, Blissville, Roxbu
ry, Middle Village.
Malba, Hammels, Bayswater, Queensbridge,
```

In [91]:
```
staten_island_venues_df = getNearbyVenues(names=staten_island_df.Neighborhood,
                                          latitudes=staten_island_df.Latitude,
                                          longitudes=staten_island_df.Longitude)
```

```
St. George, New Brighton, Stapleton, Rosebank, West Brighton, Grymes Hill, To
dt Hill.
South Beach, Port Richmond, Mariner's Harbor, Port Ivory, Castleton Corners,
New Springville, Travis.
New Dorp, Oakwood, Great Kills, Eltingville, Annadale, Woodrow, Tottenville.
Tompkinsville, Silver Lake, Sunnyside, Park Hill, Westerleigh, Graniteville,
Arlington.
Arrochar, Grasmere, Old Town, Dongan Hills, Midland Beach, Grant City, New Do
rp Beach.
Bay Terrace, Huguenot, Pleasant Plains, Butler Manor, Charleston, Rossville,
Arden Heights.
Greenridge, Heartland Village, Chelsea, Bloomfield, Bulls Head, Richmond Tow
n, Shore Acres.
Clifton, Concord, Emerson Hill, Randall Manor, Howland Hook, Elm Park, Manor
Heights.
Willowbrook, Sandy Ground, Egbertville, Prince's Bay, Lighthouse Hill, Richmo
nd Valley, Fox Hills.
```

Confirm venues for new places

Queens and Staten Island now have their venues from foursquare

In [92]:
```
print(f"The shape of queens venues df is: {queens_venues_df.shape}")
queens_venues_df.head()
```

```
The shape of queens venues df is: (2101, 7)
```

Out[92]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Ver Categ
0	Astoria	40.768509	-73.915654	Favela Grill	40.767348	-73.917897	Brazilian Restaur
1	Astoria	40.768509	-73.915654	Orange Blossom	40.769856	-73.917012	Gourme Shop
2	Astoria	40.768509	-73.915654	Titan Foods Inc.	40.769198	-73.919253	Gourme Shop
3	Astoria	40.768509	-73.915654	CrossFit Queens	40.769404	-73.918977	Gym
4	Astoria	40.768509	-73.915654	Simply Fit Astoria	40.769114	-73.912403	Gym

In [159]:
```
queens_venues_df['Venue Category'].value_counts().head()
```

Out[159]:
```
Pizza Place           84
Deli / Bodega         69
Chinese Restaurant    67
Bakery                55
Donut Shop            55
Name: Venue Category, dtype: int64
```

In [94]:
```
print(f"The shape of staten island venues df is: {staten_island_venues_df.shape}")
staten_island_venues_df.head()
```

```
The shape of staten island venues df is: (840, 7)
```

Out[94]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	V Cate
0	St. George	40.644982	-74.079353	A&S Pizzeria	40.643940	-74.077626	Pizza Place
1	St. George	40.644982	-74.079353	Beso	40.643306	-74.076508	Tapas Resta
2	St. George	40.644982	-74.079353	Richmond County Bank Ballpark	40.645056	-74.076864	Basel Stadi
3	St. George	40.644982	-74.079353	Staten Island September 11 Memorial	40.646767	-74.076510	Monu / Landr
4	St. George	40.644982	-74.079353	Nike Factory Store	40.645753	-74.077702	Sport Good Shop

In [160]:
```
staten_island_venues_df['Venue Category'].value_counts().head()
```

Out[160]:
```
Pizza Place           55
Bus Stop              44
Deli / Bodega         42
Italian Restaurant    40
Pharmacy              23
Name: Venue Category, dtype: int64
```

Prepare for Clustering

Prepare the contents of the four venues dataframes to be in one-hot format with the top venues

In [96]:
```
# Some helper functions

def get_grouped_df(venues_df):

    onehot_df = pd.get_dummies(venues_df[['Venue Category']], prefix="", prefi
x_sep="")
    onehot_df['Neighborhood'] = venues_df['Neighborhood']
    fixed_columns = [onehot_df.columns[-1]] + list(onehot_df.columns[:-1])
    onehot_df = onehot_df[fixed_columns]

    grouped_df = onehot_df.groupby('Neighborhood').mean().reset_index()

    return grouped_df

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]
```

In [97]:
```
toronto_grouped = get_grouped_df(toronto_venues_df)
manhattan_grouped = get_grouped_df(manhattan_venues_df)
queens_grouped = get_grouped_df(queens_venues_df)
staten_island_grouped = get_grouped_df(staten_island_venues_df)
```

In [98]:
```
# generate pretty column list for later
num_top_venues = 10
indicators = ['st', 'nd', 'rd']
columns = ['Neighborhood']
for ind in np.arange(num_top_venues):
    try:
        columns.append('{}{} Most Common Venue'.format(ind+1, indicators[ind
]))
    except:
        columns.append('{}th Most Common Venue'.format(ind+1))
```

In [99]:
```
# new dfs with pretty columns from above
toronto_hoods_venues_sorted = pd.DataFrame(columns=columns)
toronto_hoods_venues_sorted['Neighborhood'] = toronto_grouped['Neighborhood']
manhattan_hoods_venues_sorted = pd.DataFrame(columns=columns)
manhattan_hoods_venues_sorted['Neighborhood'] = manhattan_grouped['Neighborhoo
d']
queens_hoods_venues_sorted = pd.DataFrame(columns=columns)
queens_hoods_venues_sorted['Neighborhood'] = queens_grouped['Neighborhood']
staten_island_hoods_venues_sorted = pd.DataFrame(columns=columns)
staten_island_hoods_venues_sorted['Neighborhood'] = staten_island_grouped['Nei
ghborhood']

# pack these new dfs with their respective most common venues
for ind in np.arange(toronto_grouped.shape[0]):
    toronto_hoods_venues_sorted.iloc[ind, 1:] = return_most_common_venues(toro
nto_grouped.iloc[ind, :], num_top_venues)

for ind in np.arange(manhattan_grouped.shape[0]):
    manhattan_hoods_venues_sorted.iloc[ind, 1:] = return_most_common_venues(ma
nhattan_grouped.iloc[ind, :], num_top_venues)

for ind in np.arange(queens_grouped.shape[0]):
    queens_hoods_venues_sorted.iloc[ind, 1:] = return_most_common_venues(queen
s_grouped.iloc[ind, :], num_top_venues)

for ind in np.arange(staten_island_grouped.shape[0]):
    staten_island_hoods_venues_sorted.iloc[ind, 1:] = return_most_common_venue
s(staten_island_grouped.iloc[ind, :], num_top_venues)
```

Final form of raw data, individual

Top venues for Toronto, Manhattan, Queens, and Staten Island are sorted by neighborhood

In [100]:
```
toronto_hoods_venues_sorted.head()
```

Out[100]:

	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	7 C
0	Adelaide, King, Richmond	Coffee Shop	Café	Bar	Steakhouse	Thai Restaurant	Restaurant	B Jc
1	Agincourt	Latin American Restaurant	Lounge	Skating Rink	Breakfast Spot	Women's Store	Dumpling Restaurant	D
2	Agincourt North, L'Amoreaux East, Milliken, St...	Park	Bakery	Playground	Drugstore	Diner	Discount Store	D
3	Albion Gardens, Beaumond Heights, Humbergate, ...	Grocery Store	Pizza Place	Fried Chicken Joint	Pharmacy	Video Store	Fast Food Restaurant	B St
4	Alderwood, Long Branch	Pizza Place	Coffee Shop	Pool	Gym	Skating Rink	Pharmacy	P

In [101]:
```
manhattan_hoods_venues_sorted.head()
```

Out[101]:

	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	Battery Park City	Park	Coffee Shop	Hotel	Wine Shop	Women's Store	Gym	M S
1	Carnegie Hill	Coffee Shop	Pizza Place	Cosmetics Shop	Yoga Studio	Bakery	Gym	E
2	Central Harlem	Chinese Restaurant	African Restaurant	Cosmetics Shop	Bar	American Restaurant	French Restaurant	S F
3	Chelsea	Coffee Shop	Bakery	Ice Cream Shop	Italian Restaurant	American Restaurant	Hotel	N
4	Chinatown	Chinese Restaurant	Cocktail Bar	American Restaurant	Bakery	Hotpot Restaurant	Salon / Barbershop	V F

In [102]:
```
queens_hoods_venues_sorted.head()
```

Out[102]:

	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	Arverne	Surf Spot	Metro Station	Sandwich Place	Donut Shop	Bus Stop	Thai Restaurant
1	Astoria	Bar	Middle Eastern Restaurant	Hookah Bar	Greek Restaurant	Seafood Restaurant	Pizza Place
2	Astoria Heights	Playground	Food	Hostel	Pizza Place	Plaza	Bus Station
3	Auburndale	Ice Cream Shop	Supermarket	Gymnastics Gym	Italian Restaurant	Korean Restaurant	Furniture / Home Store
4	Bay Terrace	Clothing Store	Women's Store	American Restaurant	Lingerie Store	Mobile Phone Shop	Kids Store

In [103]:
```
staten_island_hoods_venues_sorted.head()
```

Out[103]:

	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	Annadale	Pizza Place	American Restaurant	Dance Studio	Sports Bar	Restaurant	Train Station	D
1	Arden Heights	Pharmacy	Bus Stop	Coffee Shop	Pizza Place	Event Space	Food	F S
2	Arlington	Bus Stop	Intersection	Boat or Ferry	Coffee Shop	Deli / Bodega	Hotel	E S
3	Arrochar	Deli / Bodega	Italian Restaurant	Bagel Shop	Bus Stop	Hotel	Sandwich Place	F T
4	Bay Terrace	Supermarket	Shipping Store	Insurance Office	Train Station	Sushi Restaurant	Salon / Barbershop	D S

Creating combined data

To look at all locations as a whole, we need to repeat the process above with the combinded dataset

In [104]:
```
# combine all venue data
all_venues = [toronto_venues_df, manhattan_venues_df, queens_venues_df, staten
_island_venues_df]
all_venues_df = pd.concat(all_venues)
all_venues_df.shape
```

Out[104]: (8470, 7)

In [105]:
```
all_grouped = get_grouped_df(all_venues_df)
print(f"The shape of all_grouped_df is {all_grouped.shape}")
all_grouped.head()
```

```
The shape of all_grouped_df is (277, 438)
```

Out[105]:

	Neighborhood	Yoga Studio	Accessories Store	Adult Boutique	Afghan Restaurant	African Restaurant	Airport	Airpor Food Cour
0	Adelaide, King, Richmond	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	Agincourt	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	Agincourt North, L'Amoreaux East, Milliken, St...	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	Albion Gardens, Beaumond Heights, Humbergate, ...	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	Alderwood, Long Branch	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Final data for all combined

In [106]:
```
all_hoods_venues_sorted = pd.DataFrame(columns=columns)
all_hoods_venues_sorted['Neighborhood'] = all_grouped['Neighborhood']

for ind in np.arange(all_grouped.shape[0]):
    all_hoods_venues_sorted.iloc[ind, 1:] = return_most_common_venues(all_grou
ped.iloc[ind, :], num_top_venues)

print(f"The shape of all_hoods_venues_sorted is {all_hoods_venues_sorted.shap
e}")
all_hoods_venues_sorted.head()
```

```
The shape of all_hoods_venues_sorted is (277, 11)
```

Out[106]:

	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	7 C
0	Adelaide, King, Richmond	Coffee Shop	Bar	Café	Steakhouse	Asian Restaurant	Sushi Restaurant	Bu Joi
1	Agincourt	Breakfast Spot	Lounge	Latin American Restaurant	Skating Rink	Women's Store	Ethiopian Restaurant	Dry Cle
2	Agincourt North, L'Amoreaux East, Milliken, St...	Park	Bakery	Playground	Ethiopian Restaurant	Dosa Place	Drugstore	Dry Cle
3	Albion Gardens, Beaumond Heights, Humbergate, ...	Grocery Store	Pizza Place	Fried Chicken Joint	Video Store	Pharmacy	Sandwich Place	Fas Re
4	Alderwood, Long Branch	Pizza Place	Pub	Skating Rink	Gym	Sandwich Place	Pharmacy	Co Sh

Methodology

This section deploys Kmeans clustering on the data from the Data section above. First, the data for each individual region is clustered. Then, the data for the combinded/macro/all set is clustered. For all fit clusters, 5 clusters are used. Note that this value is explored further in the Results section below (spoiler alert, 5 is likely insufficient for the macro/all set)

Individual Clustering

We cluster each region individually. This will enable the creation of a "fingerprint" map for Toronto, Manhattan, Queens and Staten Island.

In [107]:
```
def cluster_my_data(grouped_df, venues_sorted_df, lat_lon_df, kclusters=5):
    # fit cluster to the sorted venues, then merge label cluster data with geo
metric data df

    grouped_clustering = grouped_df.drop('Neighborhood', 1)
    kmeans = KMeans(n_clusters=kclusters, random_state=0).fit(grouped_clusteri
ng)

    #print(f"There are {len(kmeans.labels_)} labels, {len(grouped_df)} groups,
and {len(venues_sorted_df)} venues")

    if "Cluster Labels" in venues_sorted_df.columns:
        venues_sorted_df = venues_sorted_df.drop('Cluster Labels', 1)

    venues_sorted_df.insert(0, 'Cluster Labels', kmeans.labels_)

    clean_lat_lon_df = lat_lon_df.drop_duplicates(subset="Neighborhood")

    merged_df = clean_lat_lon_df.join(venues_sorted_df.set_index('Neighborhoo
d'), on='Neighborhood')

    merged_df.rename(columns={'Neighborhood Latitude': 'Latitude', 'Neighborho
od Longitude': 'Longitude'}, inplace=True)

    del venues_sorted_df

    return merged_df.drop(['Venue', 'Venue Category', 'Venue Latitude', 'Venue
Longitude'], 1)
```

In [108]:
```
toronto_merged = cluster_my_data(toronto_grouped, toronto_hoods_venues_sorted,
toronto_venues_df)
manhattan_merged = cluster_my_data(manhattan_grouped, manhattan_hoods_venues_s
orted, manhattan_venues_df)
queens_merged = cluster_my_data(queens_grouped, queens_hoods_venues_sorted, qu
eens_venues_df)
staten_island_merged = cluster_my_data(staten_island_grouped, staten_island_ho
ods_venues_sorted, staten_island_venues_df)
```

Macro clustering

The data for Toronto, Manhattan, Queens and Staten Island are combined, generating cluster commond accross location

In [109]:
```
all_merged = cluster_my_data(all_grouped, all_hoods_venues_sorted, all_venues_
df, kclusters=5)
print(f"The shape of all merged is {all_merged.shape}")
all_merged.head()
```

```
The shape of all merged is (277, 14)
```

Out[109]:

	Neighborhood	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Com V
0	Rouge, Malvern	43.806686	-79.194353	2	Fast Food Restaurant	Women's Store	Event Space	Dosa Plac
1	Highland Creek, Rouge Hill, Port Union	43.784535	-79.160497	1	History Museum	Bar	Moving Target	Flea Mark
4	Guildwood, Morningside, West Hill	43.763573	-79.188711	1	Spa	Rental Car Location	Mexican Restaurant	Pizz Plac
13	Woburn	43.770992	-79.216917	1	Coffee Shop	Pharmacy	Korean Restaurant	Wom Stor
17	Cedarbrae	43.773136	-79.239476	1	Thai Restaurant	Bakery	Hakka Restaurant	Bank

Results

In this section, the data above is visualized using maps with tagged icons.

First each region (toronto, manhattan, queens and staten island) are mapped individually. Then, two maps are created to display the macro data set - one center in on toronto and the other covering all New York area data. Note that these maps are showing the same clustered dataset, just two different parts of the world. These first two macro maps show that the clusters are very course. For example Manhattan and Queens contain only one cluster.

Finally, the macro data set is re-clustered with a greater cluster count. **The final two maps in this section show the macro/all dataset, with the first map centered on toronto and the second showing all of New York.**

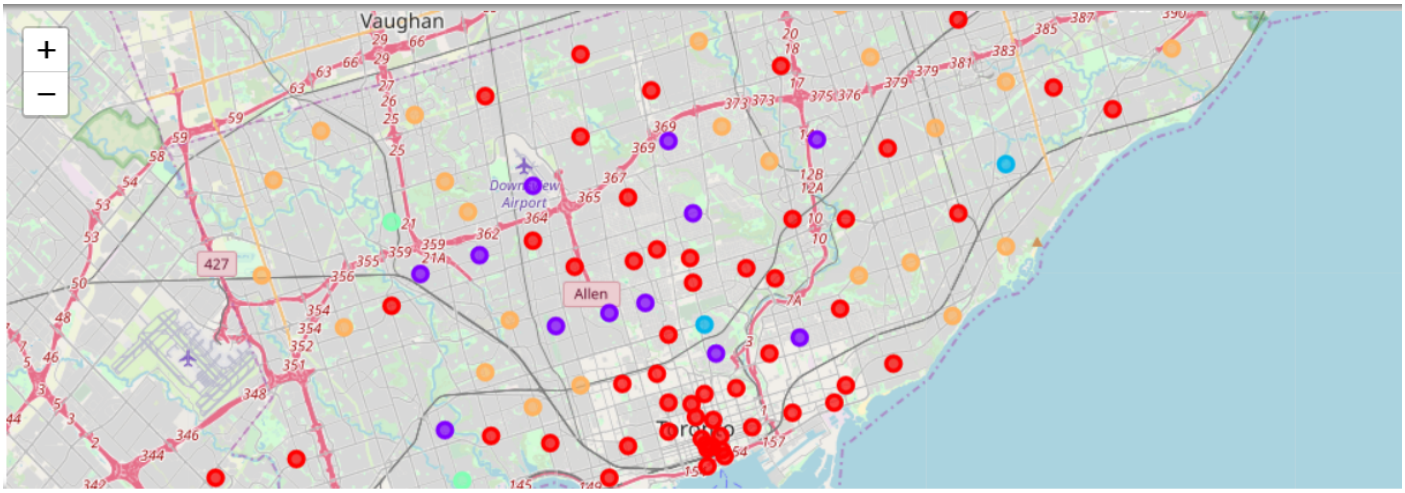
Independant clustering

What clusters appear when each location is clustered individually?

In [110]:
```
def add_clusters_to_map(merged_df, this_map):
    ### a function to add cluster data to folium map as markers

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

    # add markers to the map
    markers_colors = []
    for lat, lon, poi, cluster in zip(merged_df['Latitude'], merged_df['Longit
ude'], merged_df['Neighborhood'], merged_df['Cluster Labels']):
        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(this_map)

    return this_map
```

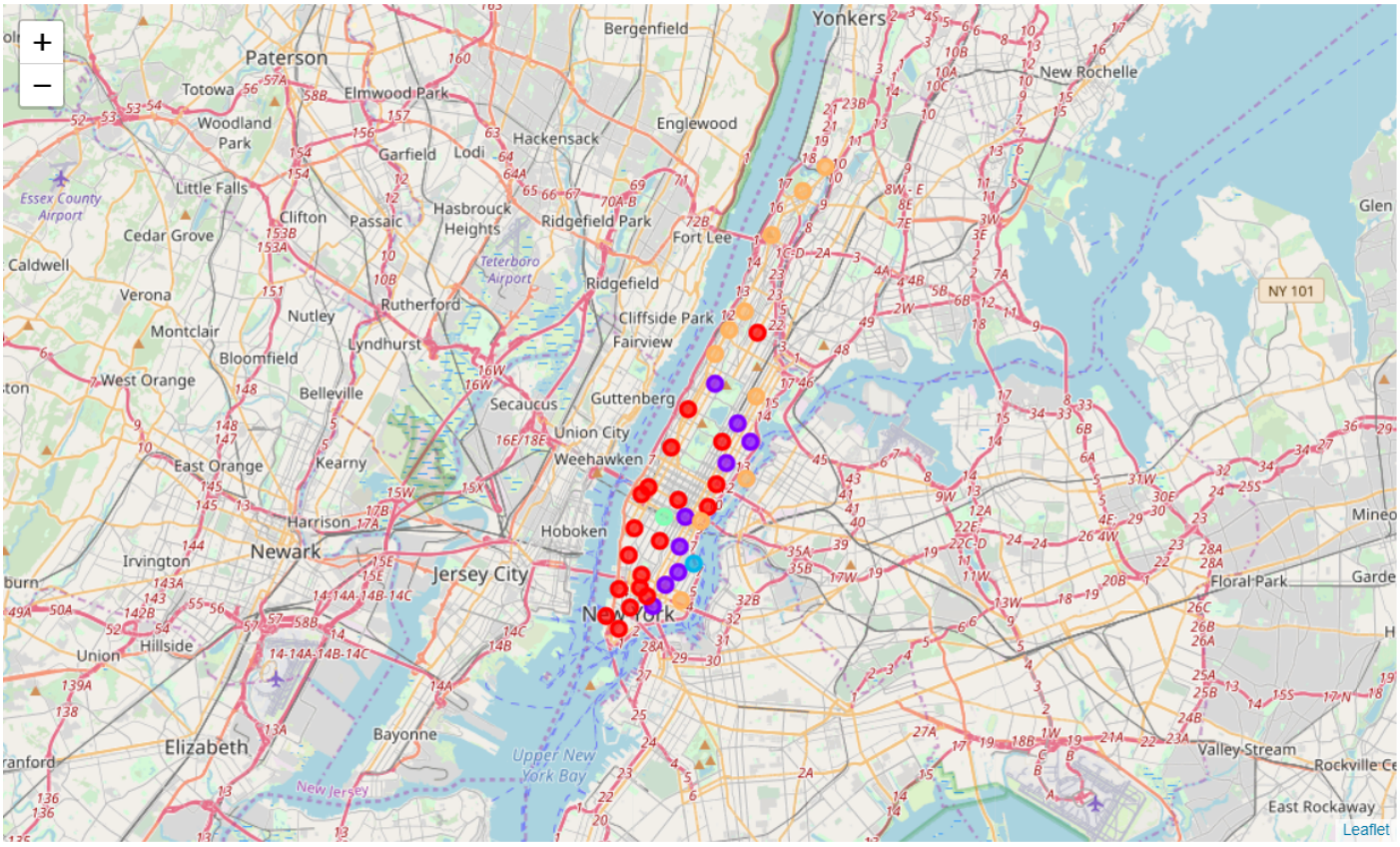
Toronto map

In [146]:
```
toronto_location = [43.6532, -79.3832]
toronto_map = folium.Map(location=toronto_location, zoom_start=11)
toronto_map = add_clusters_to_map(toronto_merged, toronto_map)
#toronto_map
```



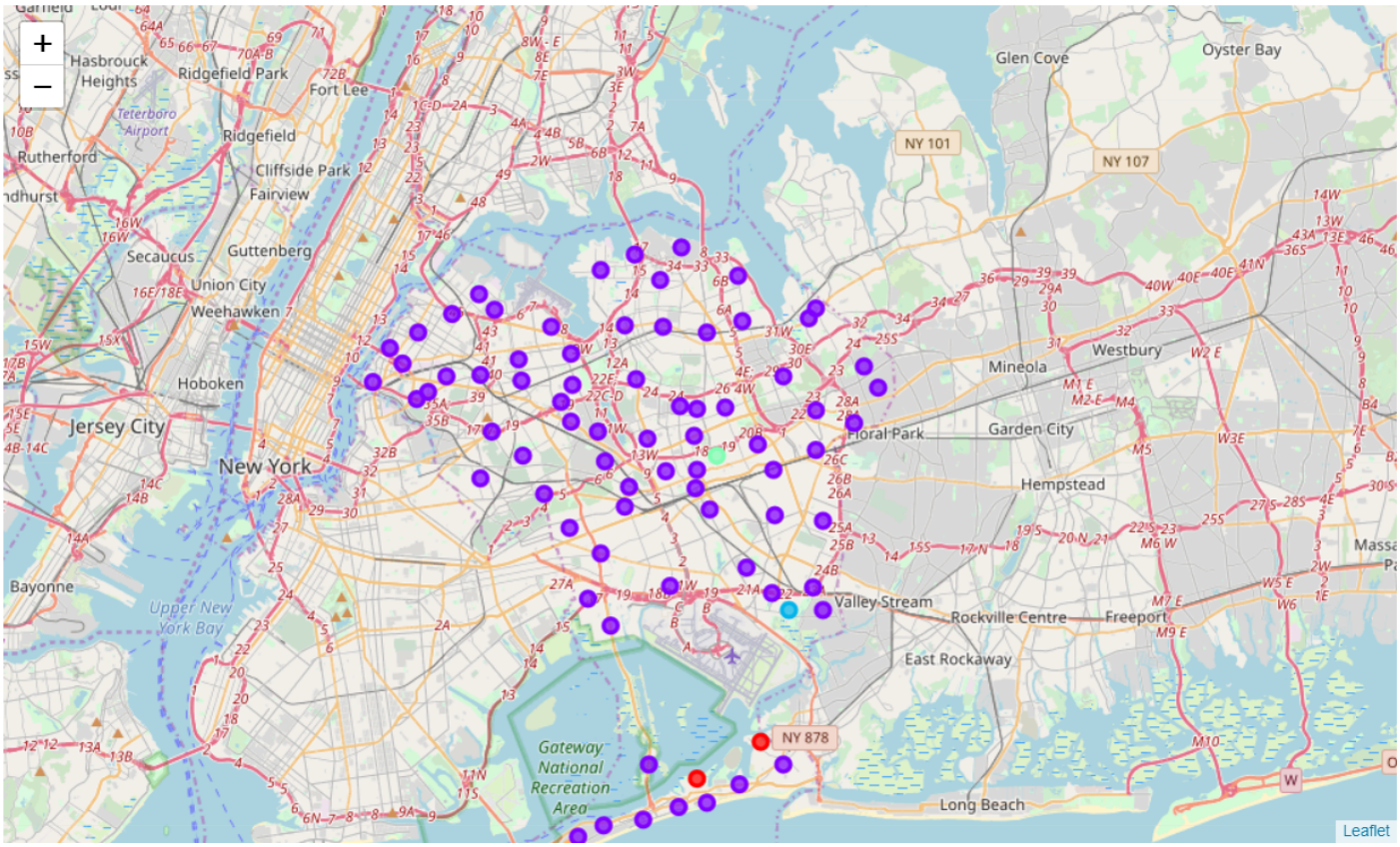
Manhattan Map

In [145]:
```
manhattan_location = [40.7831, -73.9712]
manhattan_map = folium.Map(location=manhattan_location, zoom_start=11)
manhattan_map = add_clusters_to_map(manhattan_merged, manhattan_map)
#manhattan_map
```



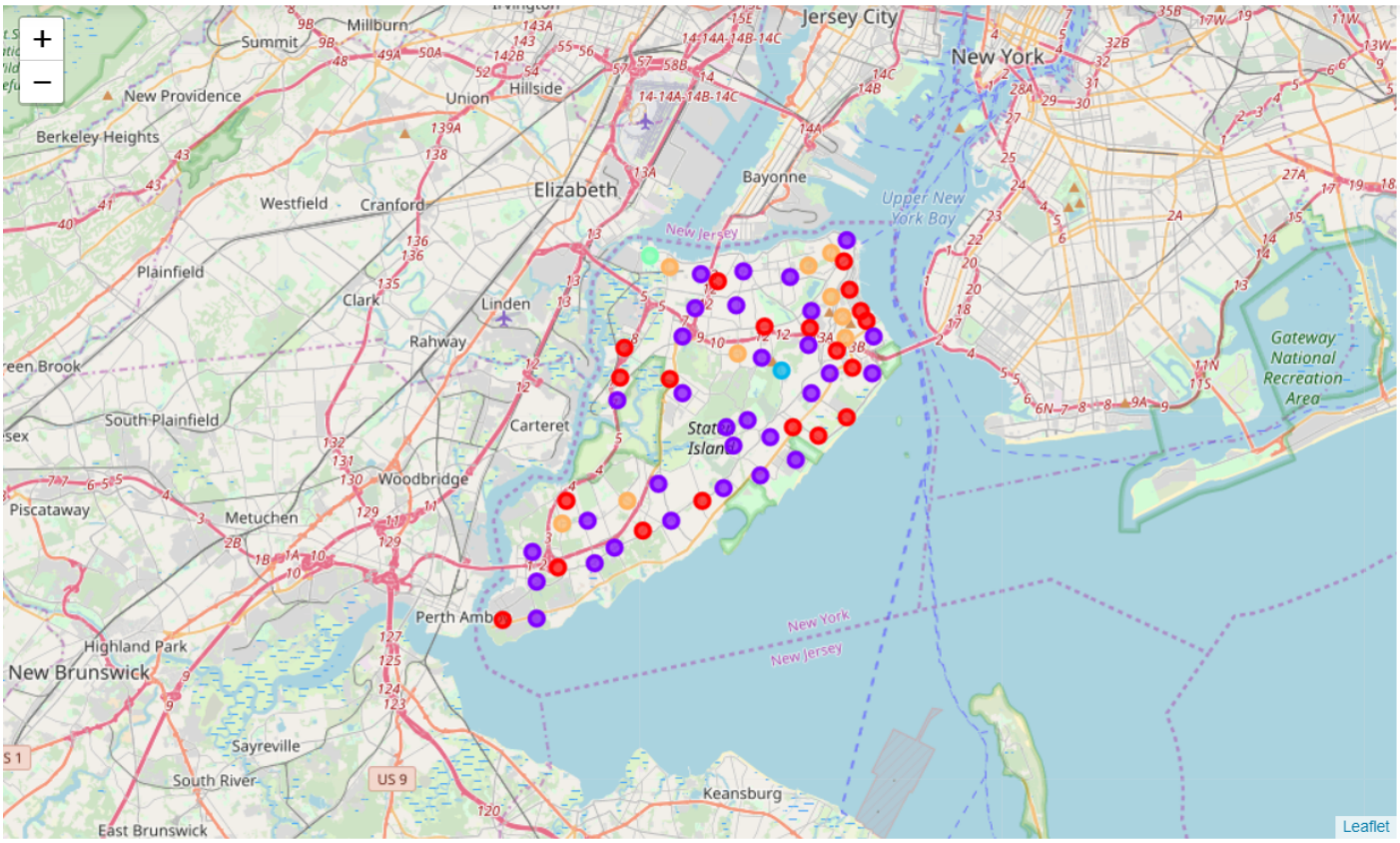
Queens map

In [144]:
```
queens_location = [40.7282, -73.7949]
queens_map = folium.Map(location=queens_location, zoom_start=11)
queens_map = add_clusters_to_map(queens_merged, queens_map)
#queens_map
```



Staten Island Map

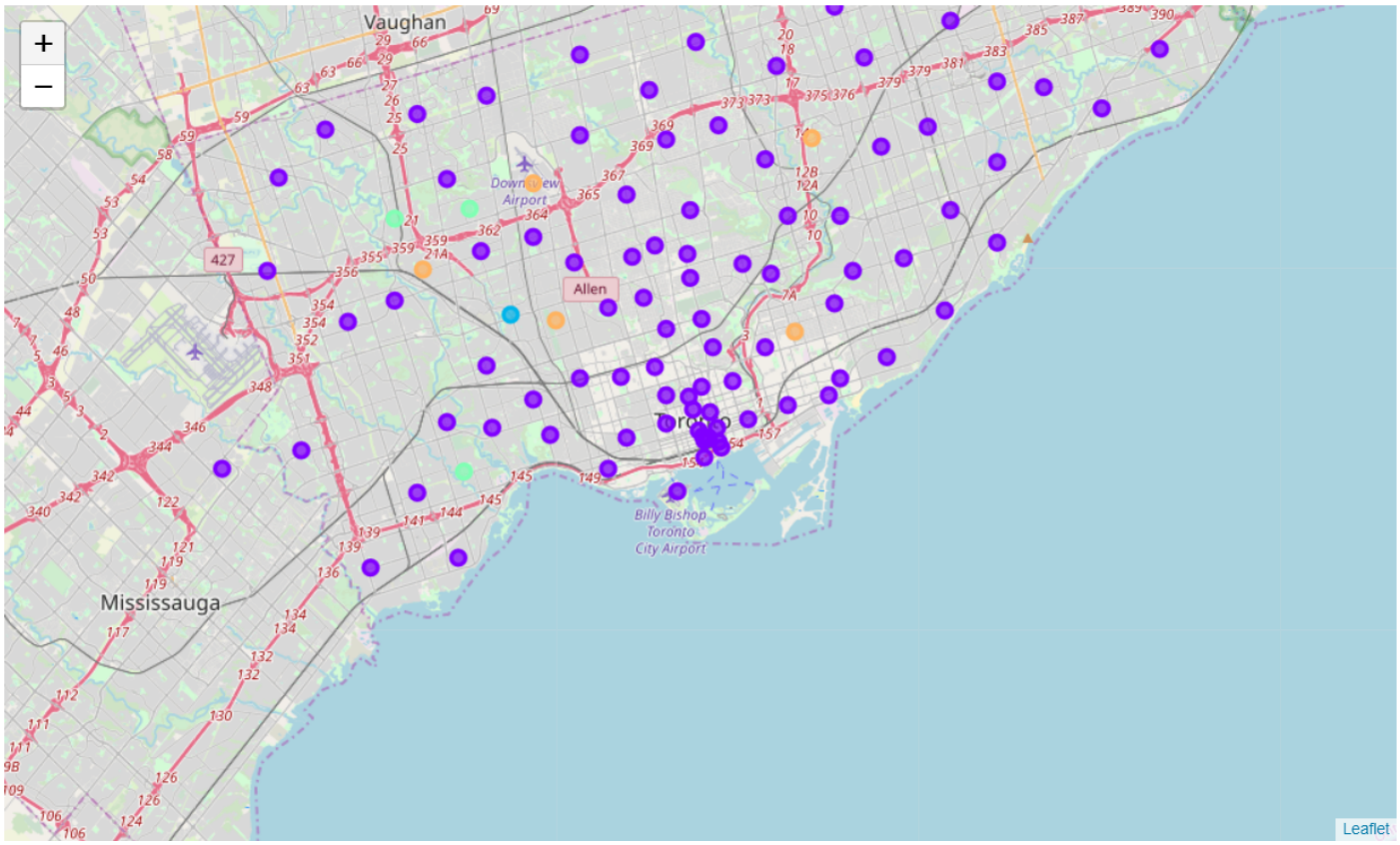
In [143]:
```
staten_island_location = [40.5795, -74.1502]
staten_island_map = folium.Map(location=staten_island_location, zoom_start=11)
staten_island_map = add_clusters_to_map(staten_island_merged, staten_island_map)
#staten_island_map
```



Macro map of all, centered in Toronto

This is the macro data, but zoomed in for clarity

In [142]:
```
all_map_toronto = folium.Map(location=toronto_location, zoom_start=11)
all_map_toronto = add_clusters_to_map(all_merged, all_map_toronto)
#all_map_toronto
```



Macro map of all, centered in New York

In [141]:
```
all_map_new_york = folium.Map(location=[40.55, -73.9442], zoom_start=10)
all_map_new_york = add_clusters_to_map(all_merged, all_map_new_york)
#all_map_new_york
```

Re-clustering macro/all, with more resolution

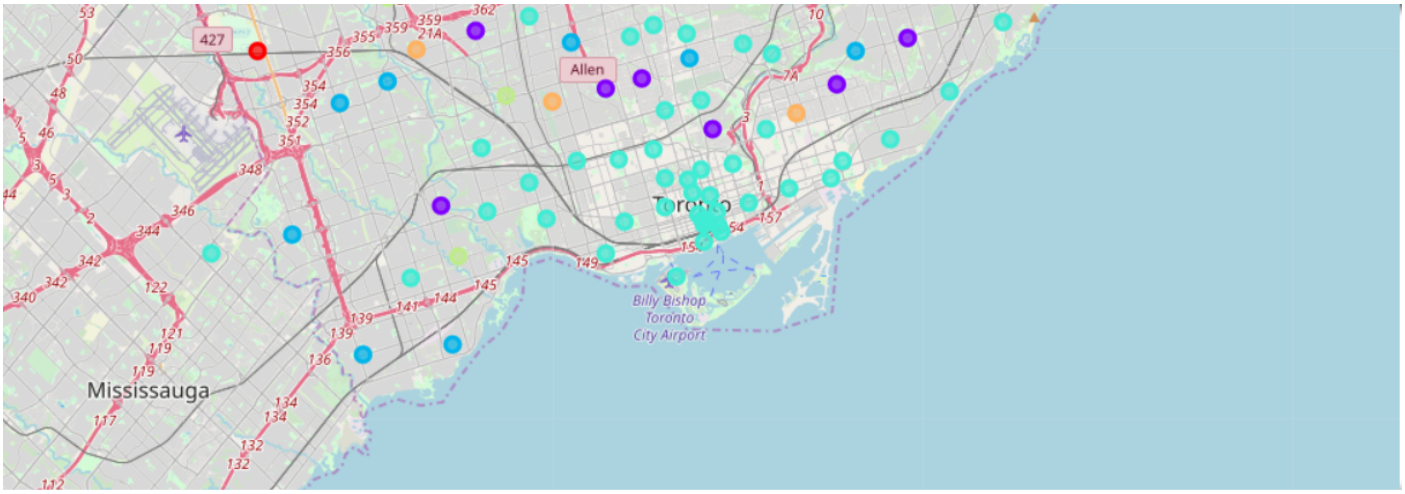
In [117]:
```
all_merged = cluster_my_data(all_grouped, all_hoods_venues_sorted, all_venues_
df, kclusters=9)
print(f"The shape of all merged is {all_merged.shape}")
```

```
The shape of all merged is (277, 14)
```

Macro map of all, centered in Toronto, higher resolution

This is the macro data, but zoomed in for clarity, with more clusters

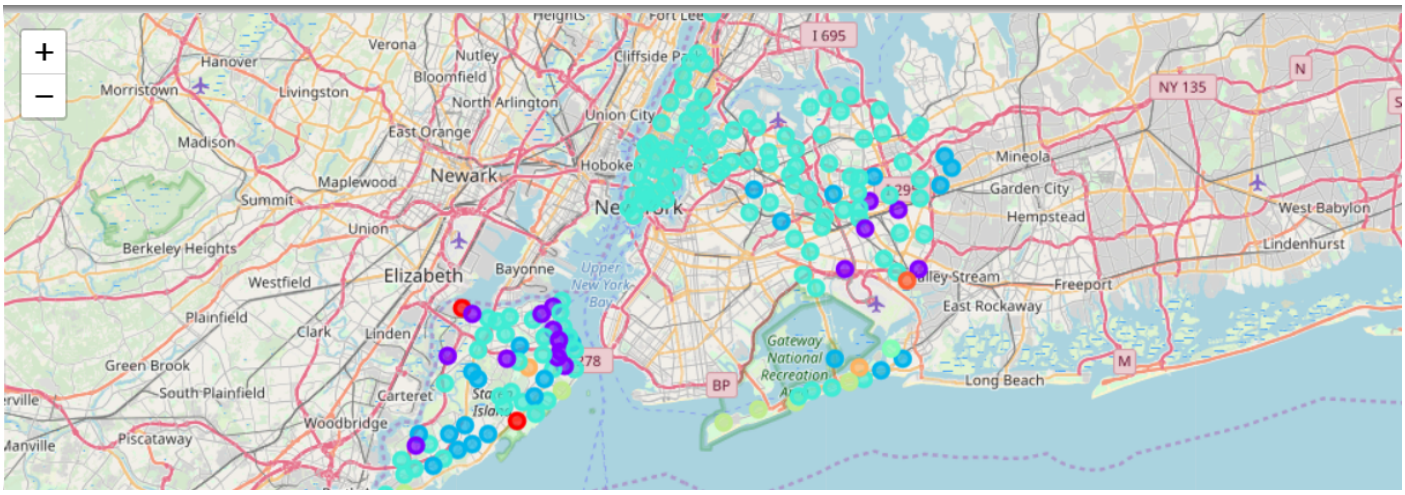
In [139]:
```
all_map_toronto = folium.Map(location=toronto_location, zoom_start=11)
all_map_toronto = add_clusters_to_map(all_merged, all_map_toronto)
#all_map_toronto
```



Macro map of all, centered in New York, higher resolution

This is the macro data, but zoomed in for clarity, with more clusters

In [137]:
```
all_map_new_york = folium.Map(location=[40.55, -73.9442], zoom_start=10)
all_map_new_york = add_clusters_to_map(all_merged, all_map_new_york)
#all_map_new_york
```



Discussion

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

Discussion of individual results

Toronto This region's clusters fall in 3 main bands centered around the city center on the lake. Bands closer to the center have more prevelent restaurants. Bands farther from the center have more prevelent shoping strip malls. The exception is cluster 1 which seems to include a band of parks at middle distance from the downtown center. Cluster 2 and 3 have so few items that they appear to be outliers, as shown below.

```
In [120]: toronto_merged['Cluster Labels'].value_counts()
```

```
Out[120]: 0    55
          4    27
          1    13
          3     2
          2     2
          Name: Cluster Labels, dtype: int64
```

Manhattan This region's clusters fall in to thee main (bottom left, bottom right and uptown categories). Clusters 2 and 3 have only one neighborhood each, so their contents are singled out below.

In [121]:
```
manhattan_merged['Cluster Labels'].value_counts()
```

Out[121]:
```
0    19
4    10
1     9
3     1
2     1
Name: Cluster Labels, dtype: int64
```

In [122]:
```
manhattan_merged.loc[manhattan_merged['Cluster Labels'] == 3]
```

Out[122]:

	Neighborhood	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Co
2720	Midtown South	40.74851	-73.988713	3	Korean Restaurant	Japanese Restaurant	Hotel	Des Sho

In [123]:
```
manhattan_merged.loc[manhattan_merged['Cluster Labels'] == 2]
```

Out[123]:

	Neighborhood	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th M Comm Ver
3101	Stuyvesant Town	40.731	-73.974052	2	Park	Bar	Baseball Field	Helipo

Queens This region's clusters are dominated by two main clusters. The first is grouped near the north-to-south and east-to-west highways that transect Queens. The second fills in the four quarters created by the highways. The remaining three regions have very few neighborhoods per cluster, so are broke out below. It does seem that cluster 4 and 2 could possible be combined, suggesting over fitting.

In [124]:
```
queens_merged['Cluster Labels'].value_counts()
```

Out[124]:
```
1    76
0     2
4     1
3     1
2     1
Name: Cluster Labels, dtype: int64
```

In [147]:
```
queens_merged.loc[queens_merged['Cluster Labels'] == 4]
```

Out[147]:

	Neighborhood	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Co
1503	Neponsit	40.572037	-73.857547	4	Beach	Bar	Fish & Chips Shop	Falaf Rest

In [151]:
```
queens_merged.loc[queens_merged['Cluster Labels'] == 3]
```

Out[151]:

	Neighborhood	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4 C
1565	Jamaica Estates	40.716805	-73.787227	3	Indian Restaurant	Intersection	Women's Store	F C S

In [152]:
```
queens_merged.loc[queens_merged['Cluster Labels'] == 2]
```

Out[152]:

	Neighborhood	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Co
1729	Brookville	40.660003	-73.751753	2	Deli / Bodega	Women's Store	Empanada Restaurant	Foc Cou

Staten Island This region's clusters are dominated by a north and south region. The remaining clusters are very small, so their contents are printed in full below.

In [128]:
```
staten_island_merged['Cluster Labels'].value_counts('')
```

Out[128]:
```
1    31
0    20
4     9
3     1
2     1
Name: Cluster Labels, dtype: int64
```

In [129]:
```
staten_island_merged.loc[staten_island_merged['Cluster Labels'] == 3]
```

Out[129]:

	Neighborhood	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Com V
166	Port Ivory	40.639683	-74.174645	3	Bar	Yoga Studio	Falafel Restaurant	Food Truck

In [150]:
```
staten_island_merged.loc[staten_island_merged['Cluster Labels'] == 2]
```

Out[150]:

	Neighborhood	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th M Comm Ven
148	Todt Hill	40.597069	-74.111329	2	Park	Yoga Studio	Event Space	Food & Drink Shop

Discussion of macro/all/combinded clusters

The macro map all - toronto shows promise in helping someone who has lived in Toronto select a neighborhood in Queens or Staten Island, because Toronto contains many (7 out of 9) common clusters clusters with both Queens and Staten Island. However, the Manhattan region of the all - new york seems to be helpfull only for those looking for what appears to be the metro region (cluster 4). Although, even this category is present in both potential move regions.

Toronto does have some clusters with only one category, so those are detailed below.

In [153]:
```
all_merged['Cluster Labels'].value_counts()
```

Out[153]:
```
4    186
3     38
1     27
6     11
7      7
0      4
5      2
8      1
2      1
Name: Cluster Labels, dtype: int64
```

In [154]:
```
all_toronto_merged = all_merged.loc[all_merged["Latitude"] > 42]
all_toronto_merged['Cluster Labels'].value_counts()
```

Out[154]:
```
4    59
3    15
1    11
7     5
6     5
0     2
5     1
2     1
Name: Cluster Labels, dtype: int64
```

In [155]:
```
all_ny_merged = all_merged.loc[all_merged["Latitude"] <= 42]
all_ny_merged['Cluster Labels'].value_counts()
```

Out[155]:
```
4    127
3     23
1     16
6      6
7      2
0      2
8      1
5      1
Name: Cluster Labels, dtype: int64
```

In [156]:
```
all_toronto_merged.loc[all_toronto_merged['Cluster Labels'] == 5]
```

Out[156]:

	Neighborhood	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th M Comm Ve
25	Scarborough Village	43.744734	-79.239476	5	Playground	Women's Store	Exhibit	Drugs

In [157]:
```
all_toronto_merged.loc[all_toronto_merged['Cluster Labels'] == 2]
```

Out[157]:

	Neighborhood	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Mc Comm Ven
163	Silver Hills, York Mills	43.75749	-79.374714	2	Cafeteria	Women's Store	Event Space	Drugstc

Final Recommendation Table

The table below uses the mapped results for the top 3 represented clusters in the all/macro data to suggest a move location, given preferences of toronto and/or manhattan clusters

Preferred cluster	**Preferred Cluster Description**	**Recommended Move location**
Cluster 4	Downtown Toronto and/or Manhattan	Northeast Queens or Northeast Staten Island
Cluster 3	Mid-town Toronto	Queens, either Laurelton Cluster or Bayswater neighborhoods
Cluster 1	Western Toronto	Southern Staten Island

Conclusion

This report stands as a successful demonstration of a move recommendation tool.

The example contained here delivers dynamically navigable maps of the "from" locations (Toronto and Manhattan) as well as the "to" locations (Queens and Staten Island) which included clustered regions. Using these maps, with their individually clustered neighborhoods, each location can be manually audited for trends.

In addition, two final maps are provided which show the clustered model of all neighborhoods. Using these maps, The majority of "from" locations can be mapped to similar location in Queens and Staten Island. This mapping is summarized in the move recommendation table above, where specific move recommendations are contained.

In general, the methods used in this report can be broadly applied to arbitrary "to" and "from" locations. Because the methods used here are automated, the goal of creating a scalable recommendation model is satisfied.