Battle of Neighborhoods in New York City: Opening a Pizza Place

Introduction

New York City is the most populated city in the United States. It is a diverse city that attracts lots of Bussinesses each year. One of the most popular bussinesses in NYC is restaurant. There are enough restaurants in NYC that you can eat for 23 years without eating in a restaurant twice. The large number of restaurants doesn't mean that all of them are successful. To thrive in such an environment, you need to do intensive study before openning one. Let's assume that we want to add a **Italian Restaurant** to the pile of restaurants in NYC. We would like to know where is the best place to open it.

Table of Contents

Before we get the data and start exploring it, let's download all the dependencies that we will need.

In [185]:
```
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

# !conda install -c conda-forge geopy --yes
from geopy.geocoders import Nominatim # convert an address into latitude and longi
tude values

import requests # library to handle requests
from pandas import json_normalize # tranform JSON file into a pandas dataframe

# Matplotlib and associated plotting modules
import matplotlib.cm as cm
import matplotlib.colors as colors
import matplotlib.pyplot as plt

# import k-means from clustering stage
from sklearn.cluster import KMeans

# !conda install -c conda-forge folium=0.5.0 --yes
import folium # map rendering library

print('Libraries imported.')
```

```
Libraries imported.
```

1. Download and Explore Dataset

New York has a total of 5 boroughs and 306 neighborhoods. We need to have a dataset that includes the essential information about each of these neighborhoods. Part of the data used in this project is extracted from the Foursquare as we go forward. So, the only information that we need from each neighborhood is it's location.

Luckily, this dataset exists for free on the web. Here is the link to the dataset: https://geo.nyu.edu/catalog/nyu_2451_34572 (https://geo.nyu.edu/catalog/nyu_2451_34572) Due to being one of the largest cities in US, there are lots of more data available for this city that can be found in internet.

Load and explore the data

For your convenience, I downloaded the files and placed it on the repository. Let's load it.

```
In [2]: with open('newyork_data.json') as json_data:
            newyork_data = json.load(json_data)
```

Let's take a quick look at the data.

```
In [136]: type(newyork_data)

Out[136]: dict
```

```
In [4]: newyork_data.keys()

Out[4]: dict_keys(['type', 'totalFeatures', 'features', 'crs', 'bbox'])
```

Notice how all the relevant data is in the *features* key, which is basically a list of the neighborhoods. So, let's define a new variable that includes this data.

```
In [5]: neighborhoods_data = newyork_data['features']
```

Let's take a look at the first item in this list.

In [6]:
```
neighborhoods_data[0]
```

Out[6]:
```
{'type': 'Feature',
 'id': 'nyu_2451_34572.1',
 'geometry': {'type': 'Point',
  'coordinates': [-73.84720052054902, 40.89470517661]},
 'geometry_name': 'geom',
 'properties': {'name': 'Wakefield',
  'stacked': 1,
  'annoline1': 'Wakefield',
  'annoline2': None,
  'annoline3': None,
  'annoangle': 0.0,
  'borough': 'Bronx',
  'bbox': [-73.84720052054902,
   40.89470517661,
   -73.84720052054902,
   40.89470517661]}}
```

As we can see, for each neighborhood, this dataset provides its name, borough, and location. We need to extract this information and convert them to a form that can be used in Python.

Tranform the data into a *pandas* dataframe

In [9]:
```
# define the dataframe columns
column_names = ['Borough', 'Neighborhood', 'Latitude', 'Longitude']

# instantiate the dataframe
NYC_neighborhoods = pd.DataFrame(columns=column_names)
for data in neighborhoods_data:
    borough = neighborhood_name = data['properties']['borough']
    neighborhood_name = data['properties']['name']

    neighborhood_latlon = data['geometry']['coordinates']
    neighborhood_lat = neighborhood_latlon[1]
    neighborhood_lon = neighborhood_latlon[0]

    NYC_neighborhoods = NYC_neighborhoods.append({'Borough': borough,
                                      'Neighborhood': neighborhood_name,
                                      'Latitude': neighborhood_lat,
                                      'Longitude': neighborhood_lon}, ignore_in
dex=True)
```

Let's look at the dataframe to confirm that it's correct.

In [10]:
```
NYC_neighborhoods.head()
```

Out[10]:

	Borough	Neighborhood	Latitude	Longitude
0	Bronx	Wakefield	40.894705	-73.847201
1	Bronx	Co-op City	40.874294	-73.829939
2	Bronx	Eastchester	40.887556	-73.827806
3	Bronx	Fieldston	40.895437	-73.905643
4	Bronx	Riverdale	40.890834	-73.912585

And make sure that the dataset has all 5 boroughs and 306 neighborhoods.

```
In [11]: print('The dataframe has {} boroughs and {} neighborhoods.'.format(
                 len(NYC_neighborhoods['Borough'].unique()),
                 NYC_neighborhoods.shape[0]
             )
         )
```

```
The dataframe has 5 boroughs and 306 neighborhoods.
```

Use geopy library to get the latitude and longitude values of New York City.

We will use this information to show the map of the new york city.

```
In [13]: address = 'New York City, NY'

         geolocator = Nominatim(user_agent="ny_explorer")
         location = geolocator.geocode(address)
         latitude = location.latitude
         longitude = location.longitude
         print('The geograpical coordinate of New York City are {}, {}.'.format(latitude, lo
         ngitude))
```

```
The geograpical coordinate of New York City are 40.7127281, -74.0060152.
```

Create a map of New York with neighborhoods superimposed on top.

To get a better understanding of the data, let's show an interactive map of the city with neighborhoods.

In [142]:
```
# create map of New York using latitude and longitude values
map_newyork = folium.Map(location=[latitude, longitude], zoom_start=10)

# add markers to map
for lat, lng, borough, neighborhood in zip(NYC_neighborhoods['Latitude'], NYC_neig
hborhoods['Longitude'], NYC_neighborhoods['Borough'], NYC_neighborhoods['Neighborh
ood']):
    label = '{}, {}'.format(neighborhood, borough)
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius=4,
        weight=1,
        popup=label,
        color='white',
        fill=True,
        fill_color='#3186cc',
        fill_opacity=0.7,
        parse_html=False).add_to(map_newyork)

map_newyork
```

Out[142]:

Next, we are going to start utilizing the Foursquare API to explore the neighborhoods and segment them.

Define Foursquare Credentials and Version

In [25]:
```
import os
from dotenv import load_dotenv
load_dotenv()

CLIENT_ID =  os.getenv('CLIENT_ID')
CLIENT_SECRET = os.getenv('CLIENT_SECRET')
VERSION=20200222
```

Let's explore the first neighborhood in our dataframe.

Let's show the workflow for a neighborhood.

In [140]:
```
print('The neighborhood is', NYC_neighborhoods.loc[0, 'Neighborhood'], 'in', NYC_n
eighborhoods.loc[0, 'Borough'],'.')
neighborhood_latitude = NYC_neighborhoods.loc[0, 'Latitude'] # neighborhood latitu
de value
neighborhood_longitude = NYC_neighborhoods.loc[0, 'Longitude'] # neighborhood long
itude value

neighborhood_name = NYC_neighborhoods.loc[0, 'Neighborhood'] # neighborhood name

print('It\'s latitude and longitude values are {}, {}.'.format(neighborhood_latitu
de,
                                                               neighborhood_longit
ude))
```

```
The neighborhood is Wakefield in Bronx .
It's latitude and longitude values are 40.89470517661, -73.84720052054902.
```

Now, let's get the top 200 venues that are in Wakefield within a radius of 500 meters.

In [32]:
```
# type your answer here
search_query=''
radius= 500
LIMIT= 200
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)
results = requests.get(url).json()
```

In [33]:
```
# function that extracts the category of the venue
def get_category_type(row):
    try:
        categories_list = row['categories']
    except:
        categories_list = row['venue.categories']

    if len(categories_list) == 0:
        return None
    else:
        return categories_list[0]['name']
```

In [141]:
```
venues = results['response']['groups'][0]['items']

nearby_venues = json_normalize(venues) # flatten JSON

# filter columns
filtered_columns = ['venue.name', 'venue.categories', 'venue.id', '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]

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

```
11 venues were returned by Foursquare.
```

Out[141]:

	name	categories	id	lat	lng
0	Lollipops Gelato	Dessert Shop	4c537892fd2ea593cb077a28	40.894123	-73.845892
1	Rite Aid	Pharmacy	4d6af9426107f04dedeb297a	40.896649	-73.844846
2	Carvel Ice Cream	Ice Cream Shop	4c783cef3badb1f7e4244b54	40.890487	-73.848568
3	Walgreens	Pharmacy	5d5f5044d0ae1c0008f043c3	40.896687	-73.844850
4	Dunkin'	Donut Shop	4c25c212f1272d7f836385c5	40.890459	-73.849089
5	Shell	Gas Station	4c81a91c51ada1cd87741510	40.894187	-73.845862
6	Cooler Runnings Jamaican Restaurant Inc	Caribbean Restaurant	508af256e4b0578944c87392	40.898083	-73.850259
7	SUBWAY	Sandwich Place	4d33665fb6093704b80001e0	40.890468	-73.849152
8	Central Deli	Deli / Bodega	4f32458019836c91c7c734ff	40.896728	-73.844387
9	Louis Pizza	Pizza Place	55aa92ac498e24734cd2e378	40.898399	-73.848810
10	Koss Quick Wash	Laundromat	5681717c498e9b9cf4d8c187	40.891281	-73.849904

In [ ]:

2. Explore Neighborhoods in New York City

Let's create a function to repeat the same process to all the neighborhoods

In [143]:
```
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_s
ecret={}&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']['id'],
            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 v
enue_list])
    nearby_venues.columns = ['Neighborhood',
                  'Neighborhood Latitude',
                  'Neighborhood Longitude',
                  'Venue',
                  'Venue id',
                  'Venue Latitude',
                  'Venue Longitude',
                  'Venue Category']

    return(nearby_venues)
```

In [60]:
```
NY_venues = getNearbyVenues(names=NYC_neighborhoods['Neighborhood'],
                                   latitudes=NYC_neighborhoods['Latitude'],
                                   longitudes=NYC_neighborhoods['Longitude']
                                  )


print('done!')
```

```
done!
```

Let's check the size of the resulting dataframe

In [146]:
```
print('In total',  NY_venues.shape[0], 'venues were found in New York')
NY_venues.head()
```

```
In total 10249 venues were found in New York
```

Out[146]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue id	Venue Latitude	Venue Longitude
0	Wakefield	40.894705	-73.847201	Lollipops Gelato	4c537892fd2ea593cb077a28	40.894123	-73.845892
1	Wakefield	40.894705	-73.847201	Rite Aid	4d6af9426107f04dedeb297a	40.896649	-73.844846
2	Wakefield	40.894705	-73.847201	Carvel Ice Cream	4c783cef3badb1f7e4244b54	40.890487	-73.848568
3	Wakefield	40.894705	-73.847201	Walgreens	5d5f5044d0ae1c0008f043c3	40.896687	-73.844850
4	Wakefield	40.894705	-73.847201	Dunkin'	4c25c212f1272d7f836385c5	40.890459	-73.849089

Let's check how many venues were returned for each neighborhood

In [159]:
```
Venue_count= NY_venues.groupby('Neighborhood').count()[['Venue']].sort_values('Ven
ue', ascending=False)
Venue_count.reset_index(inplace=True)
Venue_count.head(10)
```

Out[159]:

	Neighborhood	Venue
0	Murray Hill	147
1	Chelsea	105
2	Lenox Hill	100
3	Little Italy	100
4	Chinatown	100
5	Civic Center	100
6	Clinton	100
7	Downtown	100
8	East Village	100
9	Financial District	100

As we can see, the most venues were found in Murray Hill. Let's show the density of the venues in each neighborhood.

In [233]:
```
NY_geo = r'NTA.geojson' # geojson file

# create a plain world map
NY_map = folium.Map(location=[latitude, longitude], zoom_start=10)
# folium.Map(location=[latitude,longitude], zoom_start=11, tiles='Mapbox Bright')
```

In [234]:
```
# generate choropleth map using the total immigration of each country to Canada fr
om 1980 to 2013
NY_map.choropleth(
    geo_data=NY_geo,
    data=Venue_count,
    columns=['Neighborhood', 'Venue'],
    key_on='feature.properties.ntaname',
    fill_color='YlOrRd',
    fill_opacity=0.7,
    line_opacity=0.2,
    legend_name='Number of Venues'
)

# display map
NY_map
```

Out[234]:

Let's find out how many unique categories can be curated from all the returned venues

In [67]:
```
print('There are {} uniques categories.'.format(len(NY_venues['Venue Category'].uni
que())))
```

```
There are 433 uniques categories.
```

3. Analyze Each Neighborhood

Let's prepare the data for clustering. To do so, we would convert categories to columns where 1 means that category is in that neighborhood.

In [70]:
```
# one hot encoding
NY_onehot = pd.get_dummies(NY_venues[['Venue Category']], prefix="", prefix_sep="")

# add neighborhood column back to dataframe
NY_onehot['Neighborhood'] = NY_venues['Neighborhood']

# move neighborhood column to the first column
cols = list(NY_onehot)
cols.insert(0, cols.pop(cols.index('Neighborhood')))
NY_onehot = NY_onehot.loc[:, cols]

NY_onehot.head()
```

Out[70]:

	Neighborhood	Accessories Store	Adult Boutique	Afghan Restaurant	African Restaurant	Airport Terminal	American Restaurant	Antique Shop	Arcade	Rest
0	Wakefield	0	0	0	0	0	0	0	0	
1	Wakefield	0	0	0	0	0	0	0	0	
2	Wakefield	0	0	0	0	0	0	0	0	
3	Wakefield	0	0	0	0	0	0	0	0	
4	Wakefield	0	0	0	0	0	0	0	0	

And let's examine the new dataframe size.

In [72]:
```
NY_onehot.shape
```

Out[72]:
```
(10249, 433)
```

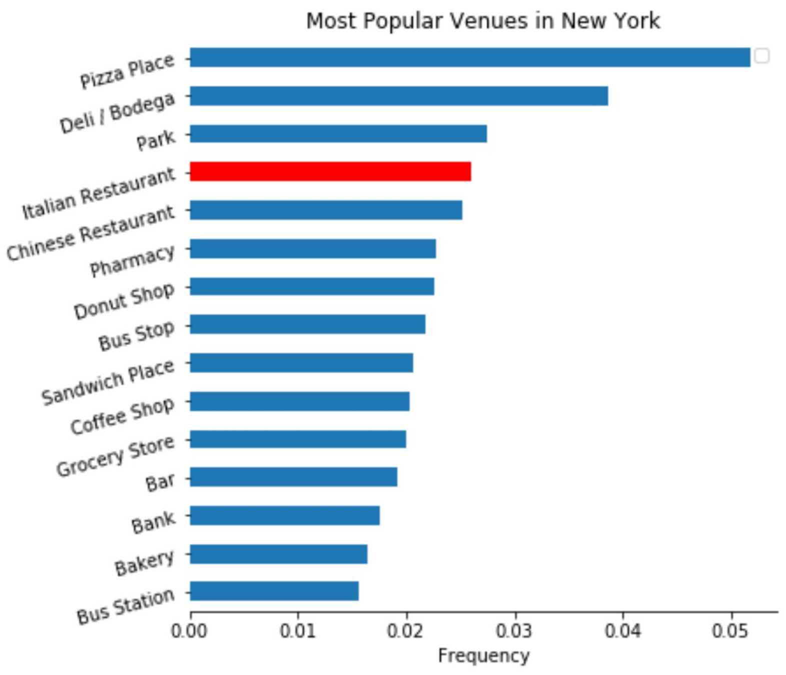
Next, let's group rows by neighborhood and by taking the mean of the frequency of occurrence of each category

In [74]:
```
NY_grouped = NY_onehot.groupby('Neighborhood').mean().reset_index()
NY_grouped.head()
```

Out[74]:

	Neighborhood	Accessories Store	Adult Boutique	Afghan Restaurant	African Restaurant	Airport Terminal	American Restaurant	Antique Shop	Arcade	Rest
0	Allerton	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1	Annadale	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2	Arden Heights	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
3	Arlington	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4	Arrochar	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

```
In [220]: sorted_venue= pd.DataFrame(NY_grouped.mean().sort_values(ascending=False)[0:15], c
          olumns=['frequency'])
          ax=sorted_venue.plot(kind='barh',rot=15,figsize=(6,6))
          ax.invert_yaxis()
          a = ax.barh(3, sorted_venue.iloc[3], height=0.5, color = 'red')
          ax.spines['top'].set_visible(False)
          ax.spines['right'].set_visible(False)
          ax.spines['left'].set_visible(False)
          plt.legend([])
          plt.xlabel('Frequency')
          plt.title('Most Popular Venues in New York')
          plt.show()
```



As we can see, Italian restaurant is the 4^{th} most popular venue in New York City.

Let's confirm the new size

```
In [75]: NY_grouped.shape

Out[75]: (301, 433)
```

Let's put that into a *pandas* dataframe

First, let's write a function to sort the venues in descending order.

```
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]
```

Now let's create the new dataframe and display the top 10 venues for each neighborhood.

In [84]:
```
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[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['Neighborhood'] = NY_grouped['Neighborhood']

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

neighborhoods_venues_sorted.head()
```

Out[84]:

	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 Most Common Venue	8th Most Common Venue
0	Allerton	Pizza Place	Deli / Bodega	Bus Station	Chinese Restaurant	Supermarket	Mexican Restaurant	Electronics Store	Martial Arts Dojo
1	Annadale	Pizza Place	Sports Bar	Diner	Train Station	Restaurant	Pub	Park	Event Service
2	Arden Heights	Pharmacy	Coffee Shop	Home Service	Pizza Place	Yoga Studio	Farmers Market	Ethiopian Restaurant	Event Service
3	Arlington	Bus Stop	Intersection	Deli / Bodega	Boat or Ferry	Grocery Store	Yoga Studio	Filipino Restaurant	Event Space
4	Arrochar	Bus Stop	Italian Restaurant	Deli / Bodega	Bagel Shop	Food Truck	Hotel	Middle Eastern Restaurant	Sandwich Place

4. Cluster Neighborhoods

Run *k*-means to cluster the neighborhood into 5 clusters.

In [89]:
```
# set number of clusters
kclusters = 7

NY_grouped_clustering = NY_grouped.drop('Neighborhood', 1)

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

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

Out[89]:
```
array([0, 0, 0, 6, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 3, 0, 1, 0, 1, 5, 0,
       1, 0, 1, 1, 1, 1, 0, 5, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 5, 0, 1,
       1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 3, 0, 1, 1, 0, 1, 0, 1, 0, 0, 5,
       0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1,
       1, 1, 0, 4, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 6, 0, 1, 0,
       1, 0, 0, 1, 1, 6, 1, 1, 0, 0, 1, 0, 1, 6, 1, 5, 0, 0, 0, 1, 1, 0,
       0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 5, 1, 1, 1, 1, 0,
       1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 6, 0, 0, 1, 5, 1, 1, 0,
       0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 5, 6, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0,
       1, 1, 1, 1, 0, 0, 1, 6, 1, 0, 1, 0, 0, 0, 0, 2, 1, 0, 0, 1, 1, 0,
       1, 0, 1, 6, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 5, 0, 1, 1, 0, 0, 5, 0,
       1, 0, 5, 1, 1, 1, 1, 3, 0, 5, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0,
       1, 0, 1, 1, 3, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1,
       1, 1, 0, 1, 1, 5, 1, 6, 1, 0, 0, 0, 0, 1, 1])
```

Let's create a new dataframe that includes the cluster as well as the top 10 venues for each neighborhood.

In [95]:
```
# add clustering labels
try:
    neighborhoods_venues_sorted.insert(0, 'Cluster Labels', kmeans.labels_)
except:
    neighborhoods_venues_sorted['Cluster Labels']=kmeans.labels_

NY_merged = NYC_neighborhoods

# merge toronto_grouped with toronto_data to add latitude/longitude for each neighb
orhood
NY_merged = NY_merged.join(neighborhoods_venues_sorted.set_index('Neighborhood'), o
n='Neighborhood')
NY_merged=NY_merged.dropna()
NY_merged['Cluster Labels']= NY_merged['Cluster Labels'].astype('int')
NY_merged.head() # check the last columns!
```

Out[95]:

	Borough	Neighborhood	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th M Comm Ve
0	Bronx	Wakefield	40.894705	-73.847201	0	Pharmacy	Laundromat	Dessert Shop	Sandwich Place	Caribb Restau
1	Bronx	Co-op City	40.874294	-73.829939	0	Discount Store	Baseball Field	Liquor Store	Pizza Place	Chir Restau
2	Bronx	Eastchester	40.887556	-73.827806	0	Caribbean Restaurant	Bus Station	Deli / Bodega	Diner	D S
3	Bronx	Fieldston	40.895437	-73.905643	1	Bus Station	River	Plaza	Yoga Studio	F
4	Bronx	Riverdale	40.890834	-73.912585	1	Park	Baseball Field	Food Truck	Bus Station	E

Finally, let's visualize the resulting clusters

In [101]:
```
# create map
map_clusters = folium.Map(location=[latitude, longitude], zoom_start=10)

# 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]

# add markers to the map
markers_colors = []
for lat, lon, poi, cluster in zip(NY_merged['Latitude'], NY_merged['Longitude'], N
Y_merged['Neighborhood'], NY_merged['Cluster Labels']):
    label = folium.Popup(str(poi) + ' Cluster ' + str(cluster), parse_html=True)
    folium.CircleMarker(
        [lat, lon],
        radius=3,
        popup=label,
        color=rainbow[cluster-1],
        weight=1,
        fill=True,
        fill_color=rainbow[cluster-1],
        fill_opacity=0.7).add_to(map_clusters)

map_clusters
```

Out[101]:

In [116]:
```
NY_merged.head()
```

Out[116]:

	Borough	Neighborhood	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th M Comm Ve
0	Bronx	Wakefield	40.894705	-73.847201	0	Pharmacy	Laundromat	Dessert Shop	Sandwich Place	Caribb Restau
1	Bronx	Co-op City	40.874294	-73.829939	0	Discount Store	Baseball Field	Liquor Store	Pizza Place	Chir Restau
2	Bronx	Eastchester	40.887556	-73.827806	0	Caribbean Restaurant	Bus Station	Deli / Bodega	Diner	D S
3	Bronx	Fieldston	40.895437	-73.905643	1	Bus Station	River	Plaza	Yoga Studio	F
4	Bronx	Riverdale	40.890834	-73.912585	1	Park	Baseball Field	Food Truck	Bus Station	E

5. Examine Clusters

Now, you can examine each cluster and determine the discriminating venue categories that distinguish each cluster.

Cluster 1

In [222]:
```
CL1=pd.DataFrame(NY_merged[NY_merged['Cluster Labels']==0].groupby(['1st Most Comm
on Venue']).count().sort_values('Neighborhood',ascending=False).iloc[:5,0])
CL1.rename(columns={'Borough':'Count'})
```

Out[222]:

	Count
1st Most Common Venue	
Pizza Place	35
Chinese Restaurant	10
Pharmacy	8
Caribbean Restaurant	7
Donut Shop	7

This cluster includes the most popular venue in New York; Pizza Place.

Cluster 2

In [225]:
```
CL2=pd.DataFrame(NY_merged[NY_merged['Cluster Labels']==1].groupby(['1st Most Comm
on Venue']).count().sort_values('Neighborhood',ascending=False).iloc[:5,0])
CL2.rename(columns={'Borough':'Count'})
```

Out[225]:

	Count
1st Most Common Venue	
Italian Restaurant	23
Coffee Shop	13
Bar	12
Park	9
Deli / Bodega	6

This is the cluster that we are interested in. The most popular venue in this cluster is the **Italian Restaurant**. Let's show this cluster on the map.

In [226]:
```
NY_map
```

Out[226]:

In [236]:
```
# 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]

# add markers to the map
markers_colors = []
for lat, lon, poi, cluster in zip(NY_merged[NY_merged['Cluster Labels']==1]['Latit
ude'], NY_merged[NY_merged['Cluster Labels']==1]['Longitude'], NY_merged[NY_merged
['Cluster Labels']==1]['Neighborhood'], NY_merged[NY_merged['Cluster Labels']==
1]['Cluster Labels']):
    label = folium.Popup(str(poi) + ' Cluster ' + str(cluster), parse_html=True)
    folium.CircleMarker(
        [lat, lon],
        radius=3,
        popup=label,
        color=rainbow[cluster-1],
        weight=1,
        fill=True,
        fill_color=rainbow[cluster-1],
        fill_opacity=0.7).add_to(NY_map)

NY_map
```

Out[236]:

The map shows the neighborhoods that are in this cluster on top of the venue density in New York. The best place to open a new italian restaurant is in these neighborhoods with denser venue numbers as more people go to these places.

Cluster 3

In [131]:
```
CL3=pd.DataFrame(NY_merged[NY_merged['Cluster Labels']==2].groupby(['1st Most Comm
on Venue']).count().sort_values('Neighborhood',ascending=False).iloc[:5,0])
CL3.rename(columns={'Borough':'Count'})
```

Out[131]:

	Count
1st Most Common Venue	
Bar	1

Cluster 4

In [132]:
```
CL4=pd.DataFrame(NY_merged[NY_merged['Cluster Labels']==3].groupby(['1st Most Comm
on Venue']).count().sort_values('Neighborhood',ascending=False).iloc[:5,0])
CL4.rename(columns={'Borough':'Count'})
```

Out[132]:

	Count
1st Most Common Venue	
Park	3
Playground	1

Cluster 5

In [133]:
```
CL5=pd.DataFrame(NY_merged[NY_merged['Cluster Labels']==4].groupby(['1st Most Comm
on Venue']).count().sort_values('Neighborhood',ascending=False).iloc[:5,0])
CL5.rename(columns={'Borough':'Count'})
```

Out[133]:

	Count
1st Most Common Venue	
Sculpture Garden	1

Cluster 6

In [133]:
```
CL6=pd.DataFrame(NY_merged[NY_merged['Cluster Labels']==5].groupby(['1st Most Comm
on Venue']).count().sort_values('Neighborhood',ascending=False).iloc[:5,0])
CL6.rename(columns={'Borough':'Count'})
```

Out[133]:

	Count
1st Most Common Venue	
Sculpture Garden	1

Cluster 7

In [134]:
```
CL7=pd.DataFrame(NY_merged[NY_merged['Cluster Labels']==6].groupby(['1st Most Comm
on Venue']).count().sort_values('Neighborhood',ascending=False).iloc[:5,0])
CL7.rename(columns={'Borough':'Count'})
```

Out[134]:

	Count
1st Most Common Venue	
Bus Stop	7
American Restaurant	1
Italian Restaurant	1