Capstone Project - The Battle of Neighborhoods

Introduction

Discussion of the problem:

After considering the data available, I have decided to consider the following problem: In Manhattan (NYC), if someone is looking to open a restaurant, where would you recommend that they open it? A recommendation will be made at the 'neighborhood-level'.

A client or a group of restaurateurs would be interested in this project because setting up at a good starting point and greatly improve the chances of their businesses surviving and/or even thriving in the competitive industry today.

Discussion of the background:

Since just simply reproducing what has already been taught in this course provides minimal value, I have searched the internet for more datasets that would be useful (discussed in more detail in the section on data). Besides looking at the number of existing potential competitors (as done in the course), I will also be using two additional datasets:

1) The historic NYPD shooting incident data

2) The population of each of the neighborhoods in each of the Borough in NYC (in years 2000 and 2010)

Reasons for the additional datasets and the additional value they provide:

1) The number of shooting incidents in a neighborhood is a telling sign of the stability and level of violence/ criminality in that particular neighborhood, and should also definitely be of concern to any business owner.

2) However, merely comparing the number of shooting incidents across the neighborhoods would be an unfair comparison, since one can expect the number of shooting incidents to be influenced by the size of the population. As such, the data on the population size of each neighborhood would be used to calculate the number of shooting incidents per capita for each of these neighborhoods.

3) While it will likely lead to an increase in shooting incidents, a greater population size will potentially result in a larger potential customer base.

Methodology:

The methodology used to determine the best neighborhood is a simple one that consists of several parts:

1) Which neighborhood balances the number of shooting incidents per capita (%) vs population size the best?

2) Are there potential competitors nearby in the neighborhood that will challenge a resturant business (use of foursquare API)?

Data section

1) Locations of neighborhoods in Manhattan

The procedures to do so follows what has been covered in the online lectures, and are shown below:

In [1]:

```
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 # uncomment this line if you haven't completed the Foursquare API lab
from geopy.geocoders import Nominatim # convert an address into latitude and longitude values

import requests # library to handle requests
from pandas.io.json 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 k-means from clustering stage
from sklearn.cluster import KMeans

#!conda install -c conda-forge folium=0.5.0 --yes # uncomment this line if you haven't completed the Foursquare API lab
import folium # map rendering library

print('Libraries imported.')
```

```
Libraries imported.
```

In [2]:

```
with open('nyu_2451_34572-geojson.json') as json_data:
    newyork_data = json.load(json_data)
```

In [3]:

```
# newyork_data
```

In [4]:

```
neighborhoods_data = newyork_data['features']
```

In [5]:

```
neighborhoods_data[0]
```

Out[5]:


```
{'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]}}
```


In [6]:

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

# instantiate the dataframe
neighborhoods = pd.DataFrame(columns=column_names)
```

In [7]:

```
neighborhoods
```

Out[7]:

Borough	Neighborhood	Latitude	Longitude

In [8]:

```
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]

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

In [9]:

```
neighborhoods.head()
```

Out[9]:

	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 [10]:

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

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

Since we are only interested in the neighborhoods in Manhattan, we slice the original dataframe and create a new dataframe of the Manhattan data.

In [11]:

```
manhattan_data = neighborhoods[neighborhoods['Borough'] == 'Manhattan'].reset_index(drop=True)
manhattan_data
```

Out[11]:

	Borough	Neighborhood	Latitude	Longitude
0	Manhattan	Marble Hill	40.876551	-73.910660
1	Manhattan	Chinatown	40.715618	-73.994279
2	Manhattan	Washington Heights	40.851903	-73.936900
3	Manhattan	Inwood	40.867684	-73.921210
4	Manhattan	Hamilton Heights	40.823604	-73.949688
5	Manhattan	Manhattanville	40.816934	-73.957385
6	Manhattan	Central Harlem	40.815976	-73.943211
7	Manhattan	East Harlem	40.792249	-73.944182
8	Manhattan	Upper East Side	40.775639	-73.960508
9	Manhattan	Yorkville	40.775930	-73.947118
10	Manhattan	Lenox Hill	40.768113	-73.958860
11	Manhattan	Roosevelt Island	40.762160	-73.949168
12	Manhattan	Upper West Side	40.787658	-73.977059
13	Manhattan	Lincoln Square	40.773529	-73.985338
14	Manhattan	Clinton	40.759101	-73.996119
15	Manhattan	Midtown	40.754691	-73.981669
16	Manhattan	Murray Hill	40.748303	-73.978332
17	Manhattan	Chelsea	40.744035	-74.003116
18	Manhattan	Greenwich Village	40.726933	-73.999914
19	Manhattan	East Village	40.727847	-73.982226
20	Manhattan	Lower East Side	40.717807	-73.980890
21	Manhattan	Tribeca	40.721522	-74.010683
22	Manhattan	Little Italy	40.719324	-73.997305
23	Manhattan	Soho	40.722184	-74.000657
24	Manhattan	West Village	40.734434	-74.006180
25	Manhattan	Manhattan Valley	40.797307	-73.964286
26	Manhattan	Morningside Heights	40.808000	-73.963896
27	Manhattan	Gramercy	40.737210	-73.981376
28	Manhattan	Battery Park City	40.711932	-74.016869
29	Manhattan	Financial District	40.707107	-74.010665
30	Manhattan	Carnegie Hill	40.782683	-73.953256
31	Manhattan	Noho	40.723259	-73.988434
32	Manhattan	Civic Center	40.715229	-74.005415
33	Manhattan	Midtown South	40.748510	-73.988713
34	Manhattan	Sutton Place	40.760280	-73.963556
35	Manhattan	Turtle Bay	40.752042	-73.967708
36	Manhattan	Tudor City	40.746917	-73.971219
37	Manhattan	Stuyvesant Town	40.731000	-73.974052
38	Manhattan	Flatiron	40.739673	-73.990947
39	Manhattan	Hudson Yards	40.756658	-74.000111

In [12]:

```
address = 'Manhattan, NY'

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

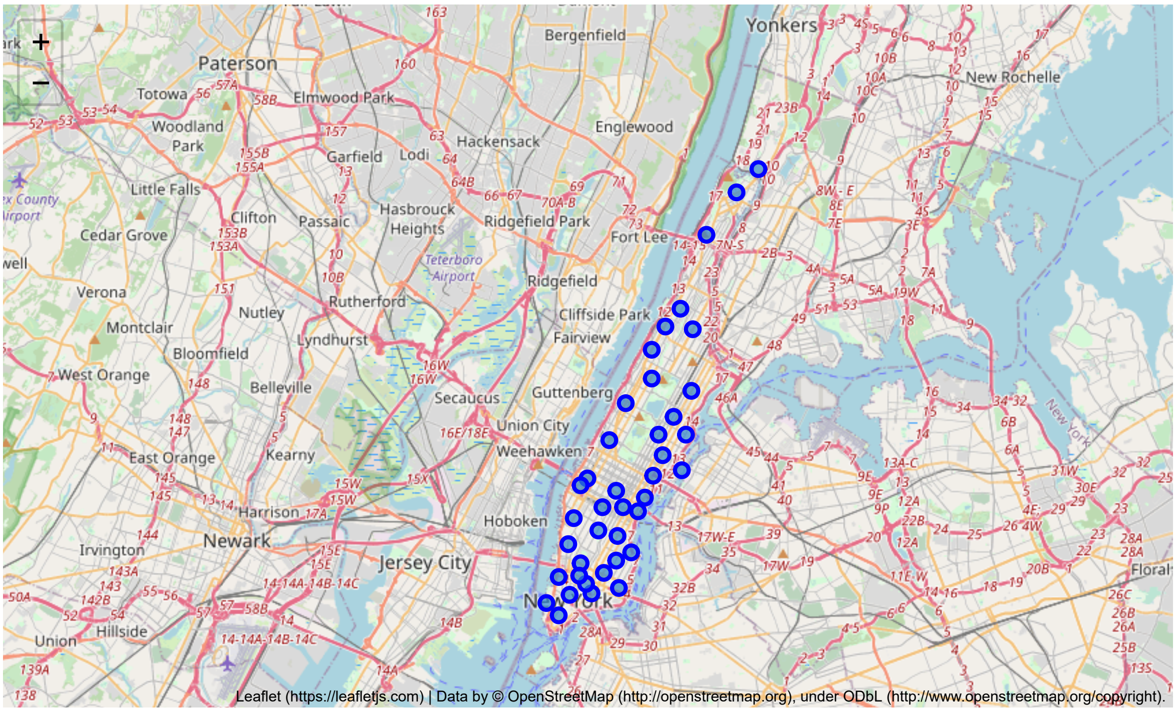
```
The geograpical coordinate of Manhattan are 40.7896239, -73.9598939.
```

Visualise the locations of the different neighborhoods on a map of Manhattan

In [13]:

```
# create map of Manhattan using latitude and longitude values
map_manhattan = folium.Map(location=[latitude, longitude], zoom_start=11)

# add markers to map
for lat, lng, label in zip(manhattan_data['Latitude'], manhattan_data['Longitude'], manhattan_data['Neighborhood']):
    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,
        parse_html=False).add_to(map_manhattan)

map_manhattan
```

Out[13]:



2) The number of venues nearby each neighborhood location (uses Foursquare API)

The procedures to do so follows what has been covered in the online lectures, and are shown below:

Define Foursquare Credentials and Version

In [14]:

```
CLIENT_ID = # 'your-client-ID' # your Foursquare ID
CLIENT_SECRET = # 'your-client-secret' # your Foursquare Secret
VERSION = '20180605' # Foursquare API version

# print('Your credentails:')
# print('CLIENT_ID: ' + CLIENT_ID)
# print('CLIENT_SECRET:' + CLIENT_SECRET)
```

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

In [15]:

```
def getNearbyVenues(names, latitudes, longitudes, radius=500):

    LIMIT = 100 # limit of number of venues returned by Foursquare API
    radius = 500 # define radius

    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={}'.form
            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)
```

Now write the code to run the above function on each neighborhood and create a new dataframe called *manhattan_venues*.

In [16]:

```
manhattan_venues = getNearbyVenues(names=manhattan_data['Neighborhood'],
                                   latitudes=manhattan_data['Latitude'],
                                   longitudes=manhattan_data['Longitude']
                                  )
```

Let's check the size of the resulting dataframe

In [17]:

```
print(manhattan_venues.shape)
manhattan_venues.head()
```

(2996, 7)

Out[17]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
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

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

In [18]:

```
manhattan_venues.groupby('Neighborhood').count()
```

Out[18]:

Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
Battery Park City	59	59	59	59	59	59
Carnegie Hill	85	85	85	85	85	85
Central Harlem	45	45	45	45	45	45
Chelsea	97	97	97	97	97	97
Chinatown	100	100	100	100	100	100
Civic Center	87	87	87	87	87	87
Clinton	100	100	100	100	100	100
East Harlem	43	43	43	43	43	43
East Village	100	100	100	100	100	100
Financial District	100	100	100	100	100	100
Flatiron	96	96	96	96	96	96
Gramercy	69	69	69	69	69	69
Greenwich Village	100	100	100	100	100	100
Hamilton Heights	61	61	61	61	61	61
Hudson Yards	53	53	53	53	53	53
Inwood	58	58	58	58	58	58
Lenox Hill	100	100	100	100	100	100
Lincoln Square	93	93	93	93	93	93
Little Italy	100	100	100	100	100	100
Lower East Side	40	40	40	40	40	40
Manhattan Valley	42	42	42	42	42	42
Manhattanville	43	43	43	43	43	43
Marble Hill	26	26	26	26	26	26
Midtown	100	100	100	100	100	100
Midtown South	93	93	93	93	93	93
Morningside Heights	39	39	39	39	39	39
Murray Hill	77	77	77	77	77	77
Noho	100	100	100	100	100	100
Roosevelt Island	24	24	24	24	24	24
Soho	71	71	71	71	71	71
Stuyvesant Town	17	17	17	17	17	17
Sutton Place	93	93	93	93	93	93
Tribeca	69	69	69	69	69	69
Tudor City	73	73	73	73	73	73
Turtle Bay	100	100	100	100	100	100
Upper East Side	85	85	85	85	85	85
Upper West Side	70	70	70	70	70	70
Washington Heights	88	88	88	88	88	88
West Village	100	100	100	100	100	100
Yorkville	100	100	100	100	100	100

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

In [19]:

```
print('There are {} uniques categories.'.format(len(manhattan_venues['Venue Category'].unique())))
```

```
There are 329 uniques categories.
```

We then use one hot encoding as a means to acquire the top 5 venues for each neighborhood.

In [20]:

```
# one hot encoding
manhattan_onehot = pd.get_dummies(manhattan_venues[['Venue Category']], prefix="", prefix_sep="")

# add neighborhood column back to dataframe
manhattan_onehot['Neighborhood'] = manhattan_venues['Neighborhood']

# move neighborhood column to the first column
fixed_columns = [manhattan_onehot.columns[-1]] + list(manhattan_onehot.columns[:-1])
manhattan_onehot = manhattan_onehot[fixed_columns]

manhattan_onehot.head()
```

Out[20]:

	Neighborhood	Accessories Store	Adult Boutique	Afghan Restaurant	African Restaurant	American Restaurant	Antique Shop	Arcade	Arepa Restaurant	Argentinian Restaurant	Art Gallery	Art Museum	Ar Cr S
0	Marble Hill	0	0	0	0	0	0	0	0	0	0	0	
1	Marble Hill	0	0	0	0	0	0	0	0	0	0	0	
2	Marble Hill	0	0	0	0	0	0	0	0	0	0	0	
3	Marble Hill	0	0	0	0	0	0	0	0	0	0	0	
4	Marble Hill	0	0	0	0	0	0	0	0	0	0	0	

And let's examine the new dataframe size.

In [21]:

```
manhattan_onehot.shape
```

Out[21]:

```
(2996, 330)
```

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

In [22]:

```
manhattan_grouped = manhattan_onehot.groupby('Neighborhood').mean().reset_index()
# manhattan_grouped
```

Let's confirm the new size

In [23]:

```
manhattan_grouped.shape
```

Out[23]:

```
(40, 330)
```

Let's print each neighborhood along with the top 5 most common venues

In [24]:

```
num_top_venues = 5

for hood in manhattan_grouped['Neighborhood']:
    print("----"+hood+"----")
    temp = manhattan_grouped[manhattan_grouped['Neighborhood'] == hood].T.reset_index()
    temp.columns = ['venue','freq']
    temp = temp.iloc[1:]
    temp['freq'] = temp['freq'].astype(float)
    temp = temp.round({'freq': 2})
    print(temp.sort_values('freq', ascending=False).reset_index(drop=True).head(num_top_venues))
    print('\n')
```

```
----Battery Park City----
           venue  freq
0           Park  0.10
1          Hotel  0.07
2  Memorial Site  0.05
3  Boat or Ferry  0.05
4            Gym  0.05


----Carnegie Hill----
           venue  freq
0    Coffee Shop  0.08
1    Yoga Studio  0.04
2      Bookstore  0.04
3    Pizza Place  0.04
4  Grocery Store  0.04


----Central Harlem----
                 venue  freq
0   African Restaurant  0.07
1                  Bar  0.04
2  American Restaurant  0.04
3   Seafood Restaurant  0.04
4   Chinese Restaurant  0.04


----Chelsea----
                venue  freq
0         Art Gallery  0.16
1         Coffee Shop  0.06
2      Ice Cream Shop  0.03
3  Italian Restaurant  0.03
4            Juice Bar  0.02


----Chinatown----
                 venue  freq
0   Chinese Restaurant  0.09
1               Bakery  0.04
2         Cocktail Bar  0.04
3                  Spa  0.03
4  American Restaurant  0.03


----Civic Center----
               venue  freq
0        Coffee Shop  0.06
1                Spa  0.05
2               Park  0.05
3              Hotel  0.05
4  French Restaurant  0.05


----Clinton----
                  venue  freq
0               Theater  0.08
1  Gym / Fitness Center  0.05
2           Coffee Shop  0.05
3                 Hotel  0.04
4             Wine Shop  0.04


----East Harlem----
                       venue  freq
0         Mexican Restaurant  0.12
1                     Bakery  0.09
2            Thai Restaurant  0.07
3              Deli / Bodega  0.05
4  Latin American Restaurant  0.05


----East Village----
                   venue  freq
0            Pizza Place  0.07
1               Wine Bar  0.04
```

```
1             Wine Bar  0.04
2          Coffee Shop  0.04
3         Cocktail Bar  0.04
4  Vietnamese Restaurant  0.03


----Financial District----
                venue  freq
0         Coffee Shop  0.08
1               Hotel  0.05
2         Pizza Place  0.04
3  American Restaurant  0.04
4                Café  0.04


----Flatiron----
                  venue  freq
0  Gym / Fitness Center  0.08
1    Italian Restaurant  0.05
2   American Restaurant  0.03
3             Wine Shop  0.03
4        Cosmetics Shop  0.03


----Gramercy----
               venue  freq
0  Italian Restaurant  0.06
1         Coffee Shop  0.06
2                 Bar  0.04
3          Bagel Shop  0.04
4  Mexican Restaurant  0.04


----Greenwich Village----
               venue  freq
0  Italian Restaurant  0.07
1         Pizza Place  0.04
2         Coffee Shop  0.04
3                 Gym  0.03
4            Wine Bar  0.03


----Hamilton Heights----
               venue  freq
0         Pizza Place  0.08
1         Coffee Shop  0.07
2                Café  0.07
3       Deli / Bodega  0.07
4  Mexican Restaurant  0.05


----Hudson Yards----
                  venue  freq
0    Italian Restaurant  0.06
1                 Hotel  0.06
2  Gym / Fitness Center  0.06
3   American Restaurant  0.06
4                  Park  0.04


----Inwood----
               venue  freq
0  Mexican Restaurant  0.07
1          Restaurant  0.05
2                Café  0.05
3              Lounge  0.05
4         Pizza Place  0.05


----Lenox Hill----
               venue  freq
0  Italian Restaurant  0.06
1         Pizza Place  0.05
2         Coffee Shop  0.05
3        Cocktail Bar  0.04
4                Café  0.04


----Lincoln Square----
               venue  freq
0               Plaza  0.05
1                Café  0.05
2  Italian Restaurant  0.05
3             Theater  0.04
4        Concert Hall  0.04


----Little Italy----
                     venue  freq
0       Chinese Restaurant  0.05
1                      Spa  0.05
```

```
1                     Spa  0.05
2                  Bakery  0.04
3      Italian Restaurant  0.04
4  Mediterranean Restaurant  0.04


----Lower East Side----
                venue  freq
0  Chinese Restaurant  0.08
1                Café  0.05
2         Art Gallery  0.05
3        Cocktail Bar  0.05
4         Yoga Studio  0.02


----Manhattan Valley----
                venue  freq
0         Coffee Shop  0.10
1  Mexican Restaurant  0.05
2                 Spa  0.05
3                 Bar  0.05
4         Pizza Place  0.05


----Manhattanville----
                venue  freq
0         Coffee Shop  0.07
1  Seafood Restaurant  0.07
2  Italian Restaurant  0.05
3  Mexican Restaurant  0.05
4                Park  0.05


----Marble Hill----
              venue  freq
0    Sandwich Place  0.12
1               Gym  0.08
2       Coffee Shop  0.08
3       Yoga Studio  0.04
4  Department Store  0.04


----Midtown----
            venue  freq
0     Coffee Shop  0.06
1           Hotel  0.05
2  Clothing Store  0.05
3         Theater  0.04
4             Spa  0.03


----Midtown South----
                 venue  freq
0    Korean Restaurant  0.15
1                Hotel  0.06
2  Japanese Restaurant  0.05
3         Dessert Shop  0.04
4         Burger Joint  0.04


----Morningside Heights----
                 venue  freq
0                 Park  0.10
1  American Restaurant  0.08
2          Coffee Shop  0.08
3            Bookstore  0.08
4         Deli / Bodega  0.05


----Murray Hill----
                 venue  freq
0          Coffee Shop  0.05
1       Sandwich Place  0.05
2                Hotel  0.05
3          Pizza Place  0.04
4  Japanese Restaurant  0.04


----Noho----
                venue  freq
0  Italian Restaurant  0.05
1         Coffee Shop  0.05
2       Grocery Store  0.04
3      Sandwich Place  0.04
4           Wine Shop  0.03


----Roosevelt Island----
                   venue  freq
0          Metro Station  0.04
1  Outdoors & Recreation  0.04
```

```
1  Outdoors & Recreation  0.04
2                    Gym  0.04
3            Coffee Shop  0.04
4         Farmers Market  0.04


----Soho----
                      venue  freq
0       Italian Restaurant  0.08
1  Mediterranean Restaurant  0.06
2               Coffee Shop  0.04
3                       Gym  0.04
4            Clothing Store  0.04


----Stuyvesant Town----
               venue  freq
0      Boat or Ferry  0.12
1               Park  0.12
2        Coffee Shop  0.06
3     Farmers Market  0.06
4  German Restaurant  0.06


----Sutton Place----
                  venue  freq
0    Italian Restaurant  0.06
1  Gym / Fitness Center  0.05
2                  Park  0.04
3           Coffee Shop  0.03
4            Bagel Shop  0.03


----Tribeca----
                venue  freq
0                Park  0.07
1  Italian Restaurant  0.07
2            Wine Bar  0.04
3                 Spa  0.04
4                Café  0.04


----Tudor City----
                venue  freq
0                Café  0.07
1                Park  0.07
2  Mexican Restaurant  0.05
3       Deli / Bodega  0.04
4         Pizza Place  0.04


----Turtle Bay----
                venue  freq
0                Café  0.05
1         Coffee Shop  0.05
2                Park  0.04
3  Italian Restaurant  0.04
4            Wine Bar  0.04


----Upper East Side----
                  venue  freq
0    Italian Restaurant  0.08
1  Gym / Fitness Center  0.05
2             Juice Bar  0.05
3                Bakery  0.05
4           Yoga Studio  0.04


----Upper West Side----
                venue  freq
0  Italian Restaurant  0.06
1         Coffee Shop  0.04
2              Bakery  0.04
3      Ice Cream Shop  0.03
4                 Bar  0.03


----Washington Heights----
                venue  freq
0                Café  0.06
1              Bakery  0.05
2  Chinese Restaurant  0.03
3         Pizza Place  0.03
4   Mobile Phone Shop  0.03


----West Village----
                 venue  freq
0   Italian Restaurant  0.08
1             Wine Bar  0.06
```

```
1            Wine Bar  0.06
2         Coffee Shop  0.06
3  American Restaurant  0.05
4           Jazz Club  0.04


----Yorkville----
                venue  freq
0         Coffee Shop  0.08
1  Italian Restaurant  0.07
2                 Gym  0.05
3                 Bar  0.04
4       Deli / Bodega  0.04
```

Let's put that into a *pandas* dataframe

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

In [25]:

```
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 5 venues for each neighborhood.

In [26]:

```
num_top_venues = 5

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'] = manhattan_grouped['Neighborhood']

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

neighborhoods_venues_sorted.head()
```

Out[26]:

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue
0	Battery Park City	Park	Hotel	Gym	Boat or Ferry	Memorial Site
1	Carnegie Hill	Coffee Shop	Yoga Studio	Wine Shop	Pizza Place	Japanese Restaurant
2	Central Harlem	African Restaurant	Chinese Restaurant	French Restaurant	Cosmetics Shop	Seafood Restaurant
3	Chelsea	Art Gallery	Coffee Shop	Italian Restaurant	Ice Cream Shop	Hotel
4	Chinatown	Chinese Restaurant	Bakery	Cocktail Bar	Optical Shop	Salon / Barbershop

3) Acquring NYPD Shooting Incident Data (Historic)

In [27]:

```
from sodapy import Socrata
import pickle
```

In [28]:

```
# Unauthenticated client only works with public data sets. Note 'None'
# in place of application token, and no username or password:
client = Socrata("data.cityofnewyork.us", None)

# Example authenticated client (needed for non-public datasets):
# client = Socrata(data.cityofnewyork.us,
#                  MyAppToken,
#                  userame="user@example.com",
#                  password="AFakePassword")

# First 20000 results, returned as JSON from API / converted to Python list of
# dictionaries by sodapy.
results = client.get("833y-fsy8", limit=22000)

# Convert to pandas DataFrame
incident_df = pd.DataFrame.from_records(results)

# Check shape
incident_df.shape
```

```
WARNING:root:Requests made without an app_token will be subject to strict throttling limits.
```

Out[28]:

```
(22000, 18)
```

Extract the data we are interested in

In [29]:

```
incident_df = incident_df[incident_df.boro == 'MANHATTAN']
incident_df = incident_df[['boro','latitude','longitude']]
incident_df.drop(columns=['boro'], inplace = True)
incident_df.columns = ['Latitude', 'Longitude']
incident_df.reset_index(drop=True, inplace=True)
```

In [30]:

```
incident_df.head()
```

Out[30]:

	Latitude	Longitude
0	40.80024432600004	-73.95339008999997
1	40.79415025600008	-73.93986908699996
2	40.80186520800004	-73.95723931799995
3	40.80941319900006	-73.94436716399997
4	40.79501283900004	-73.93613752499994

In [31]:

```
incident_df.shape
```

Out[31]:

```
(2705, 2)
```

Plotting the location heatmap for the locations of shooting incidents

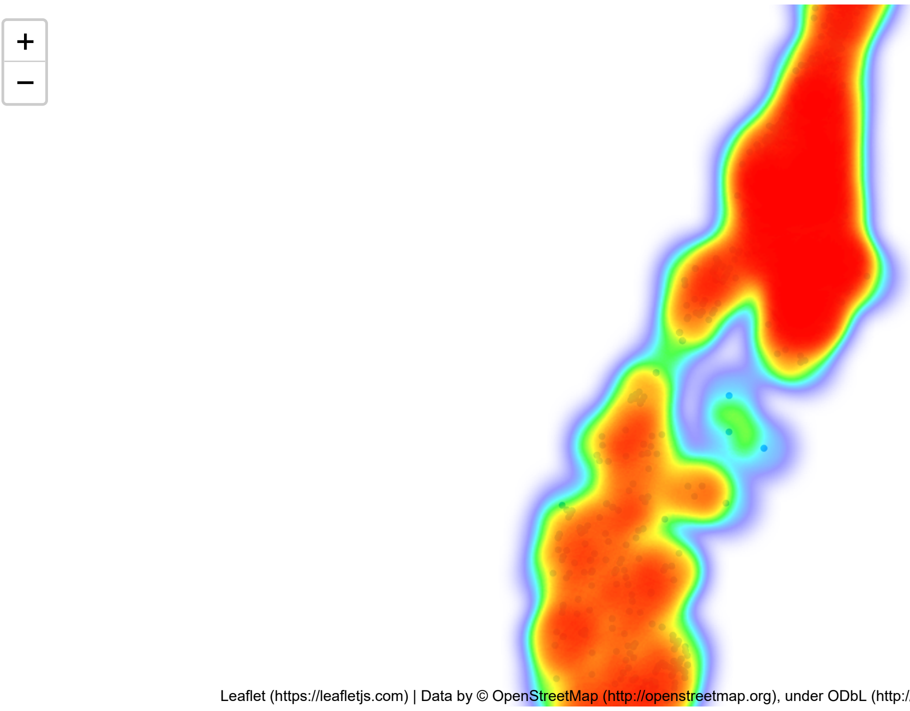
In [32]:

```
import folium
from folium import plugins
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [33]:

```
shooting_map = folium.Map(location=[40.773529, -73.985338], zoom_start=11.5, prefer_canvas = True)

# mark each station as a point
for index, row in incident_df.iterrows():
    folium.Circle([row['Latitude'], row['Longitude']],
                        radius=15,
                        fill_color="#3db7e4", # divvy color
                       ).add_to(shooting_map)

# convert to (n, 2) nd-array format for heatmap
incidentArr = incident_df[['Latitude', 'Longitude']].values

# plot heatmap
shooting_map.add_child(plugins.HeatMap(incidentArr))

shooting_map
```

Out[33]:



Leaflet (https://leafletjs.com) | Data by © OpenStreetMap (http://openstreetmap.org), under ODbL (http://www.openstreetmap.org/copyright).

Besides storing the NYPD Shooting Incident Data into a dataframe, the algorithm below assigns each shooting incident to a neighborhood; based on the latitude-longitude of the neighborhood location the incident is closest to.

In [34]:

```
# vectorized haversine function
def haversine(lat1, lon1, manhattan_data, to_radians, earth_radius):
    """
    slightly modified version: of http://stackoverflow.com/a/29546836/2901002

    Calculate the great circle distance between two points
    on the earth (specified in decimal degrees or in radians)

    All (lat, lon) coordinates must have numeric dtypes and be of equal length.

    """

    # convert type of columns as required
    manhattan_data = manhattan_data.astype({"Latitude":'float64', "Longitude":'float64'})

    # initialise large minimum distance to be overwritten:
    minimum_distance = 1000000
    neighborhood_assigned = 'to-be-assigned'

    for n_lat, n_lng, neighborhood in zip(manhattan_data['Latitude'],
                                                  manhattan_data['Longitude'],
                                                  manhattan_data['Neighborhood']):

        if to_radians:
            lat1, lon1, n_lat, n_lng = np.radians([lat1, lon1, n_lat, n_lng])

        a = np.sin((n_lat-lat1)/2.0)**2 + np.cos(lat1) * np.cos(n_lat) * np.sin((n_lng-lon1)/2.0)**2

        distance = earth_radius * 2 * np.arcsin(np.sqrt(a))

        if distance < minimum_distance:
            minimum_distance = distance
            neighborhood_assigned = neighborhood

    return minimum_distance, neighborhood_assigned
```

In [35]:

```
incident_distance = []
incident_neighborhood = []

for i_lat, i_lng in zip(incident_df['Latitude'], incident_df['Longitude']):
    incident_distance_i, incident_neighborhood_i = haversine(lat1 = float(i_lat), lon1 = float(i_lng),
                                                              manhattan_data = manhattan_data,
                                                              to_radians = False, earth_radius = 6371)
    incident_distance.append(incident_distance_i)
    incident_neighborhood.append(incident_neighborhood_i)

incident_distances_df = pd.DataFrame(incident_distance, incident_neighborhood)
incident_distances_df.reset_index(drop=False, inplace = True)
incident_distances_df.columns = ['Neighborhood','Incident_distance']

print(incident_distances_df.shape)
```

(2705, 2)

In [36]:

```
incident_distances_df = pd.DataFrame(incident_distances_df.Neighborhood.value_counts())
incident_distances_df = incident_distances_df.reset_index(drop = False, inplace = False)
incident_distances_df.columns = ['Neighborhood', 'Count']
incident_distances_df = pd.merge(manhattan_data, incident_distances_df, how='left', on=['Neighborhood'])
incident_distances_df = incident_distances_df.fillna(0)
incident_distances_df = incident_distances_df.astype({"Count":'int32'})
incident_distances_df.drop(columns=['Borough', 'Latitude', 'Longitude'], inplace = True)
```

4) Acquire the population sizes of different neighborhoods

In [37]:

```
# Unauthenticated client only works with public data sets. Note 'None'
# in place of application token, and no username or password:
client = Socrata("data.cityofnewyork.us", None)

# Example authenticated client (needed for non-public datasets):
# client = Socrata(data.cityofnewyork.us,
#                  MyAppToken,
#                  userame="user@example.com",
#                  password="AFakePassword")

# First 20000 results, returned as JSON from API / converted to Python list of
# dictionaries by sodapy.
results = client.get("swpk-hqdp", limit=390)

# Convert to pandas DataFrame
population_df = pd.DataFrame.from_records(results)

# Check shape
population_df.shape
```

```
WARNING:root:Requests made without an app_token will be subject to strict throttling limits.
```

Out[37]:

```
(390, 6)
```

In [38]:

```
population_df.head()
```

Out[38]:

	borough	fips_county_code	nta_code	nta_name	population	year
0	Bronx	005	BX01	Claremont-Bathgate	28149	2000
1	Bronx	005	BX03	Eastchester-Edenwald-Baychester	35422	2000
2	Bronx	005	BX05	Bedford Park-Fordham North	55329	2000
3	Bronx	005	BX06	Belmont	25967	2000
4	Bronx	005	BX07	Bronxdale	34309	2000

Extract the data we are interested in and clean it

In [39]:

```
population_df = population_df[population_df.borough == 'Manhattan']
population_df = population_df[population_df.year == '2010']
population_df = population_df[['borough','nta_name','population','year']]

temp_nta = population_df['nta_name'].str.split('-').apply(pd.Series, 1).stack()
temp_nta.index = temp_nta.index.droplevel(-1) # to line up with df's index
temp_nta.name = 'nta_name' # needs a name to join
del population_df['nta_name']
population_df = population_df.join(temp_nta)
population_df.columns = ['Borough','Population','Year','Neighborhood']
```

In [40]:

```
population_df.shape
```

Out[40]:

```
(47, 4)
```

In [41]:

```
population_df.head()
```

Out[41]:

	Borough	Population	Year	Neighborhood
284	Manhattan	46746	2010	Marble Hill
284	Manhattan	46746	2010	Inwood
285	Manhattan	75282	2010	Central Harlem North
285	Manhattan	75282	2010	Polo Grounds
286	Manhattan	48520	2010	Hamilton Heights

Merge dataframes. This dataframe will serve as the 'base' dataframe other dataframes will merge with later.

In [42]:

```
add_dataframe = pd.merge(manhattan_data, population_df, how='left', on=['Borough','Neighborhood'])
add_dataframe.dropna(how='any', thresh=None, subset=None, inplace=True)
add_dataframe.reset_index(drop=True, inplace=True)
add_dataframe.drop(columns=['Year'], inplace = True)
```

In [43]:

```
add_dataframe
```

Out[43]:

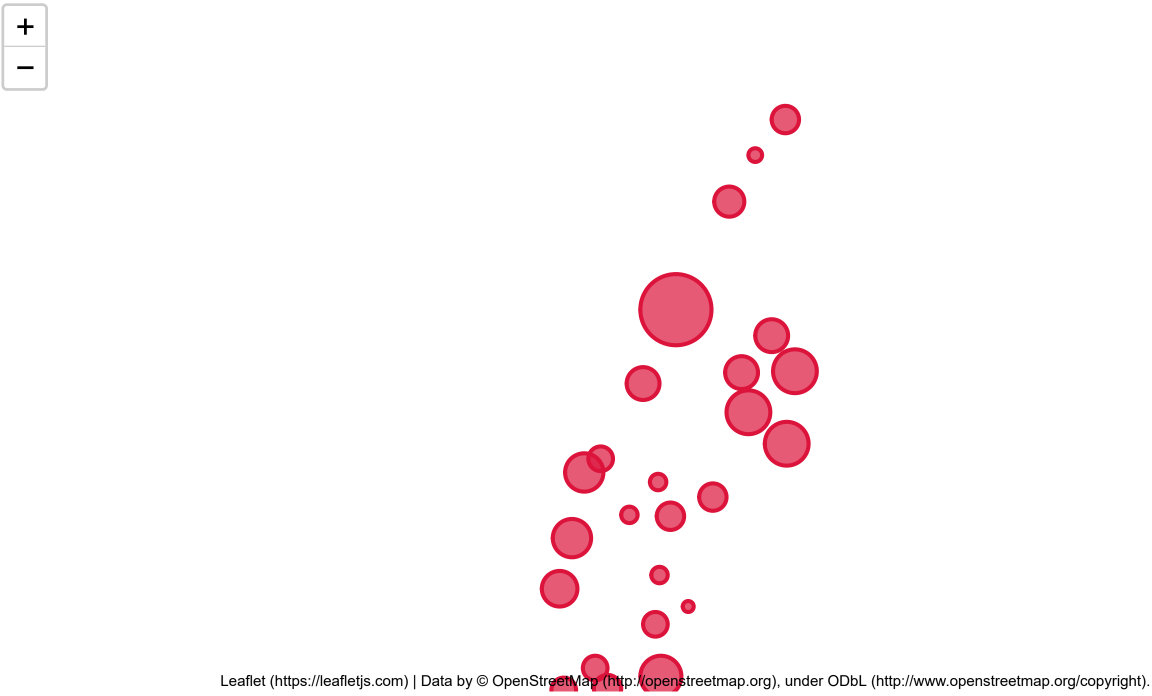
	Borough	Neighborhood	Latitude	Longitude	Population
0	Manhattan	Marble Hill	40.876551	-73.910660	46746
1	Manhattan	Chinatown	40.715618	-73.994279	47844
2	Manhattan	Inwood	40.867684	-73.921210	46746
3	Manhattan	Hamilton Heights	40.823604	-73.949688	48520
4	Manhattan	Manhattanville	40.816934	-73.957385	22950
5	Manhattan	Upper East Side	40.775639	-73.960508	61207
6	Manhattan	Yorkville	40.775930	-73.947118	77942
7	Manhattan	Lenox Hill	40.768113	-73.958860	80771
8	Manhattan	Roosevelt Island	40.762160	-73.949168	80771
9	Manhattan	Upper West Side	40.787658	-73.977059	132378
10	Manhattan	Lincoln Square	40.773529	-73.985338	61489
11	Manhattan	Clinton	40.759101	-73.996119	45884
12	Manhattan	Midtown	40.754691	-73.981669	28630
13	Manhattan	Murray Hill	40.748303	-73.978332	50742
14	Manhattan	Chelsea	40.744035	-74.003116	70150
15	Manhattan	East Village	40.727847	-73.982226	44136
16	Manhattan	Lower East Side	40.717807	-73.980890	72957
17	Manhattan	Little Italy	40.719324	-73.997305	42742
18	Manhattan	West Village	40.734434	-74.006180	66880
19	Manhattan	Morningside Heights	40.808000	-73.963896	55929
20	Manhattan	Gramercy	40.737210	-73.981376	27988
21	Manhattan	Battery Park City	40.711932	-74.016869	39699
22	Manhattan	Carnegie Hill	40.782683	-73.953256	61207
23	Manhattan	Civic Center	40.715229	-74.005415	42742
24	Manhattan	Midtown South	40.748510	-73.988713	28630
25	Manhattan	Turtle Bay	40.752042	-73.967708	51231
26	Manhattan	Stuyvesant Town	40.731000	-73.974052	21049
27	Manhattan	Hudson Yards	40.756658	-74.000111	70150

Plotting the populations on a folium map, where larger circles symbolise locations with a larger population.

In [44]:

```
# create map of Manhattan using latitude and longitude values
manhattan_population = folium.Map(location=[40.773529, -73.985338], zoom_start=11.5)

# add markers to map
for lat, lng, neighborhood, population in zip(add_dataframe['Latitude'],
                                              add_dataframe['Longitude'],
                                              add_dataframe['Neighborhood'],
                                              add_dataframe['Population'].astype(int)):

    label = '{}, {}'.format(neighborhood, population)
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius=population / 5000,
        popup=label,
        color='crimson',
        fill=True,
        fill_color='crimson',
        fill_opacity=0.7,
        parse_html=False).add_to(manhattan_population)

manhattan_population
```

Out[44]:



Results

The cell below describes the distribution of the two key variables; the Population and the Incidents per capita (%)

In [45]:

```
results_dataframe = pd.merge(add_dataframe, incident_distances_df, how='left', on=['Neighborhood'])

results_dataframe = results_dataframe.astype({"Latitude":'str',
                                              "Longitude":'str',
                                              "Count":"int",
                                              "Population":'int'})

results_dataframe['Incidents per capita (%)'] = results_dataframe['Count'] / results_dataframe['Population']
results_dataframe = pd.merge(results_dataframe, neighborhoods_venues_sorted, how='left', on=['Neighborhood'])
results_dataframe.describe()
```

Out[45]:

	Population	Count	Incidents per capita (%)
count	28.000000	28.000000	28.000000
mean	54575.357143	36.678571	0.000880
std	22914.833829	62.470173	0.001746
min	21049.000000	0.000000	0.000000
25%	42742.000000	3.750000	0.000068
50%	49631.000000	13.500000	0.000252
75%	67697.500000	34.500000	0.000703
max	132378.000000	280.000000	0.007625

The cell below sorts and displays the 10 neighborhoods with the smallest Incidents per capita (%), a desirable trait.

In [46]:

```
results_dataframe = results_dataframe.sort_values(by = ['Incidents per capita (%)'])
results_dataframe.head(10)
```

Out[46]:

	Borough	Neighborhood	Latitude	Longitude	Population	Count	Incidents per capita (%)	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue
0	Manhattan	Marble Hill	40.87655077879964	-73.91065965862981	46746	0	0.000000	Sandwich Place	Gym	Coffee Shop
21	Manhattan	Battery Park City	40.71193198394565	-74.01686930508617	39699	0	0.000000	Park	Hotel	Gym
8	Manhattan	Roosevelt Island	40.76215960576283	-73.94916769227953	80771	0	0.000000	Pizza Place	Scenic Lookout	Coffee Shop
5	Manhattan	Upper East Side	40.775638573301805	-73.96050763135	61207	1	0.000016	Italian Restaurant	Bakery	Gym / Fitness Center
7	Manhattan	Lenox Hill	40.76811265828733	-73.9588596881376	80771	3	0.000037	Italian Restaurant	Coffee Shop	Pizza Place
6	Manhattan	Yorkville	40.775929849884875	-73.94711784471826	77942	3	0.000038	Coffee Shop	Italian Restaurant	Gym
13	Manhattan	Murray Hill	40.748303077252174	-73.97833207924127	50742	2	0.000039	Sandwich Place	Hotel	Coffee Shop
25	Manhattan	Turtle Bay	40.75204236950722	-73.96770824581834	51231	4	0.000078	Café	Coffee Shop	Wine Bar
18	Manhattan	West Village	40.73443393572434	-74.00617998126812	66880	8	0.000120	Italian Restaurant	Wine Bar	Coffee Shop
17	Manhattan	Little Italy	40.71932379395907	-73.99730467208073	42742	6	0.000140	Spa	Chinese Restaurant	Mediterranean Restaurant

The cell below sorts and displays the 10 neighborhoods with the largest population sizes, a desirable trait.

In [47]:

```
results_dataframe = results_dataframe.sort_values(by = ['Population'], ascending = False)
results_dataframe.head(10)
```

Out[47]:

	Borough	Neighborhood	Latitude	Longitude	Population	Count	Incidents per capita (%)	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4 C
9	Manhattan	Upper West Side	40.787657998534854	-73.97705923630603	132378	32	0.000242	Italian Restaurant	Bakery	Coffee Shop	
8	Manhattan	Roosevelt Island	40.76215960576283	-73.94916769227953	80771	0	0.000000	Pizza Place	Scenic Lookout	Coffee Shop	
7	Manhattan	Lenox Hill	40.76811265828733	-73.9588596881376	80771	3	0.000037	Italian Restaurant	Coffee Shop	Pizza Place	Res
6	Manhattan	Yorkville	40.775929849884875	-73.94711784471826	77942	3	0.000038	Coffee Shop	Italian Restaurant	Gym	Res
16	Manhattan	Lower East Side	40.71780674892765	-73.98089031999291	72957	94	0.001288	Chinese Restaurant	Café	Cocktail Bar	Art
14	Manhattan	Chelsea	40.744034706747975	-74.00311633472813	70150	48	0.000684	Art Gallery	Coffee Shop	Italian Restaurant	Ice
27	Manhattan	Hudson Yards	40.75665808227519	-74.00011136202637	70150	10	0.000143	Hotel	Italian Restaurant	Gym / Fitness Center	Ar Res
18	Manhattan	West Village	40.73443393572434	-74.00617998126812	66880	8	0.000120	Italian Restaurant	Wine Bar	Coffee Shop	Ar Res
10	Manhattan	Lincoln Square	40.77352888942166	-73.98533777001262	61489	26	0.000423	Italian Restaurant	Café	Plaza	(
5	Manhattan	Upper East Side	40.775638573301805	-73.96050763135	61207	1	0.000016	Italian Restaurant	Bakery	Gym / Fitness Center	Ju

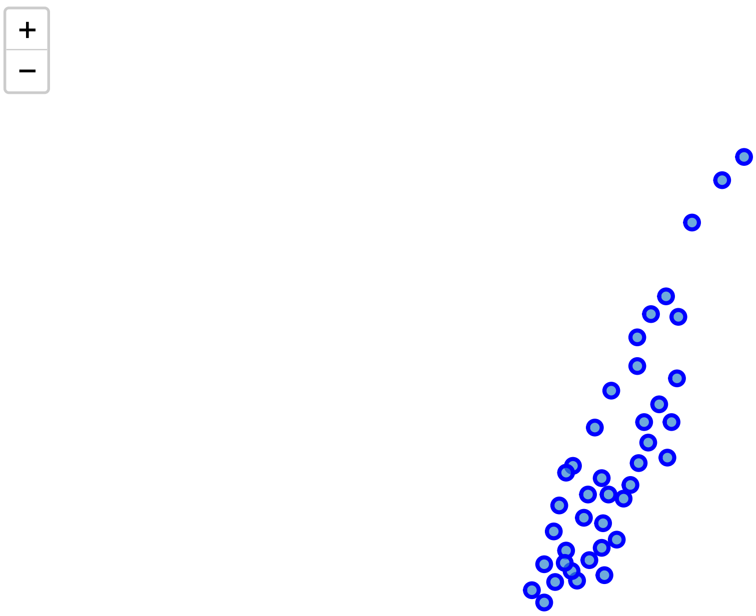
The folium maps rendered in this notebook are consolidated below

A map of the locations of the different neighborhoods in Manhattan

In [48]:

```
map_manhattan
```

Out[48]:



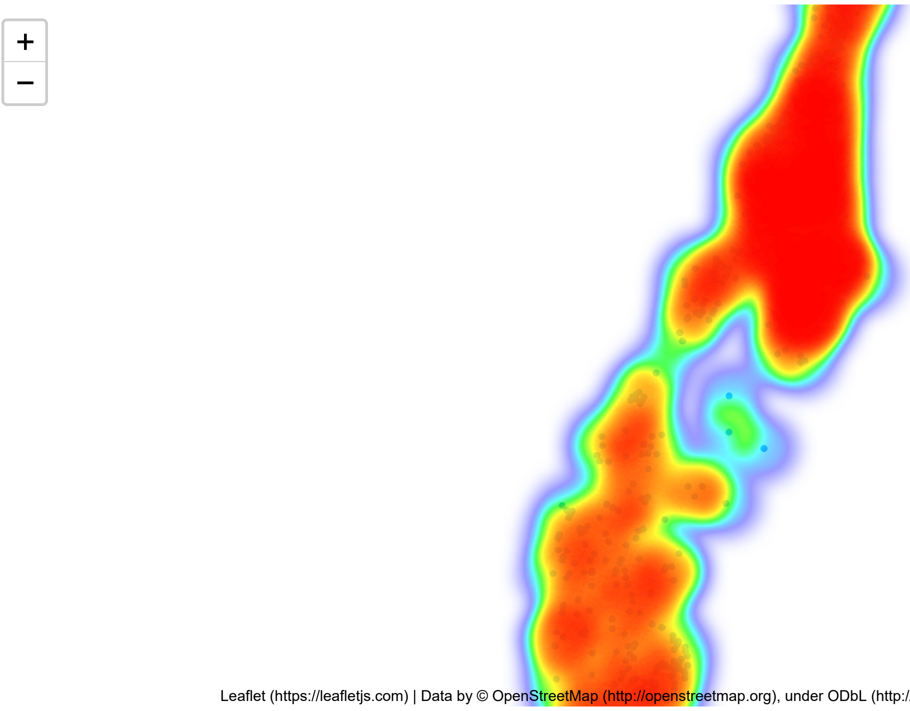
Leaflet (https://leafletjs.com) | Data by © OpenStreetMap (http://openstreetmap.org), under ODbL (http://www.openstreetmap.org/copyright).

A heatmap of the shooting incidents in Manhattan

In [49]:

```
shooting_map
```

Out[49]:



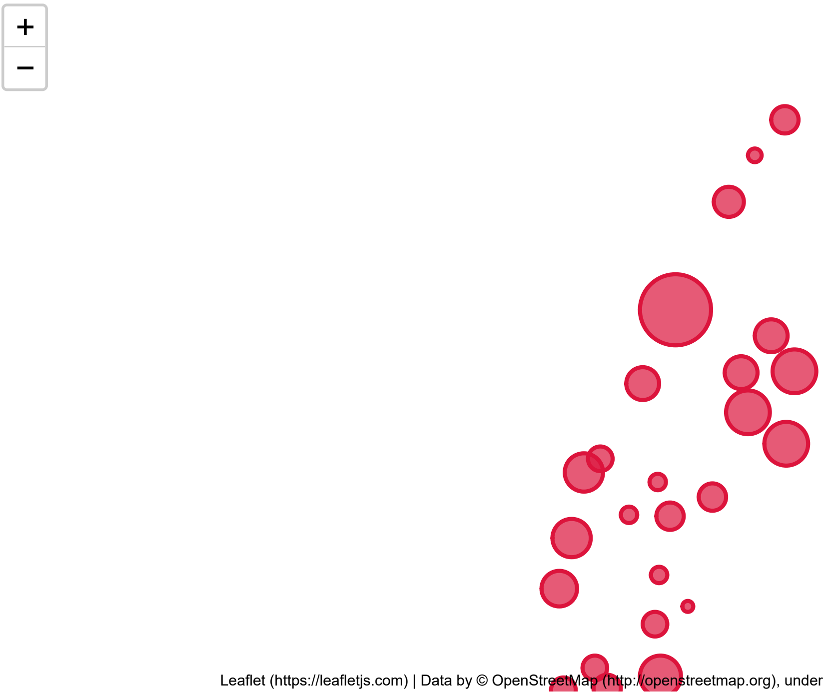
Leaflet (https://leafletjs.com) | Data by © OpenStreetMap (http://openstreetmap.org), under ODbL (http://www.openstreetmap.org/copyright).

The population sizes of the different neighborhoods in Manhattan

In [50]:

```
manhattan_population
```

Out[50]:



Leaflet (https://leafletjs.com) | Data by © OpenStreetMap (http://openstreetmap.org), under ODbL (http://www.openstreetmap.org/copyright).

Conclusions

1) By observation, the best neighborhood for a restaurateur to set up shop in Manhattan is Roosevelt Island. By comparing the two tables produced in the results section, Roosevelt Island has the best balance between having a large population size and a low crime of shooting (0).

2) However, due to the limited data available online, it can be observed that the number of features used for comparison between the neighborhoods are very limited. Given the chance, more data points and features of greater variety should be used for analysis.