Capstone Project - The Battle of Neighborhoods

Jawharah Almulhim

10 June -2020

1. **Introduction:**

1.1 Background

New York city also called NYC , it is one of the largest city on USA with huge number of population and diversity of people. The NYC is a main distance for visitors from all over the world . The idea of this project is to explore Manhattan which is often referred to by residents of the New York City area as the City, it is one of the most densely populated of the five boroughs of New York City. Exploring neighborhoods and venues of this borough is handled on this project.

1.2 Problem

The idea of this project suggest a Higley rated parks on Manhattan since mostly, visitors are willing to visit different places to enjoy themselves and parks are one of these places.
Also , ending up with clustering different parks of Manhattan into different clusters with similar features.

1.3 Audience

- 1-Vistors who are willing to visit highly rated parks
- 2-Governemnt agencies when they planning to open a new park ,so new location of park could be close to a highly rated one.

2. **Data acquisition and cleaning**

2.1 Data sources

To accomplish this project , different data sources are used:

- **New York City data** that contains list Boroughs, Neighborhoods along with their latitude and longitude. Data source : https://cocl.us/new_york_dataset Explanation: the above data set is available for free and it contains main data of NYC like latitude, longitude , boroughs and neighborhoods.
- **Foursquare API service:** using API calls to get neighborhoods and venues of the selected borough as well as detailed information about venues such as tips, likes, rating and more. Such information is necessary for clustering.
- **Pandas data frames** is used to store the results of the API calls and do the operations
- **Geopy** is client which is used to locate the coordinates of addresses using third-party geocoders
- **K-mean clustering :**machine learning tool to cluster the parks on different cluster based on similarities

2.2 Data cleaning

First, data downloaded from https://cocl.us/new_york_dataset in JSON format and the required features are fetched which are brought, neighborhood, latitude and longitude. Then, this data is stored in pandas dataframe.

	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
5	Bronx	Kingsbridge	40.881687	-73.902818
6	Manhattan	Marble Hill	40.876551	-73.910660
7	Bronx	Woodlawn	40.898273	-73.867315
8	Bronx	Norwood	40.877224	-73.879391

Figure 1:Dataframe of New York Data

Also, Multiple operations on dataframe are performed to filter rows to reach the target data which Manhattan's neighborhoods.

Later ,Foursquare API used to make RESTful API calls to retrieve data about venues of the selected borough , Venues retrieved from all the neighborhoods along with their category, latitude , longitude and name.

An extract of an API call is as follows:


```
{'meta': {'code': 200, 'requestId': '5ede334c949393001cde5537'},
 'response': {'suggestedFilters': {'header': 'Tap to show:',
   'filters': [{'name': 'Open now', 'key': 'openNow'}]},
  'headerLocation': 'Marble Hill',
  'headerFullLocation': 'Marble Hill, New York',
  'headerLocationGranularity': 'neighborhood',
  'totalResults': 26,
  'suggestedBounds': {'ne': {'lat': 40.88105078329964,
    'lng': -73.90471933917806},
   'sw': {'lat': 40.87205077429964, 'lng': -73.91659997808156}},
  'groups': [{'type': 'Recommended Places',
    'name': 'recommended',
    'items': [{'reasons': {'count': 0,
       'items': [{'summary': 'This spot is popular',
         'type': 'general',
         'reasonName': 'globalInteractionReason'}]},
      'venue': {'id': '4b4429abf964a52037f225e3',
       'name': "Arturo's",
       'location': {'address': '5198 Broadway',
        'crossStreet': 'at 225th St.',
        'lat': 40.87441177110231,
        'lng': -73.91027100981574,
        'labeledLatLngs': [{'label': 'display',
          'lat': 40.87441177110231,
          'lng': -73.91027100981574},
```


Figure 2 Response from Foursquare API call

Lastly, K-Mean clustering was applied on numerical data after it's being normalized , since data normalization helps to interpret features with different magnitudes and distributions equally.

	ID	Lat	Lan	Likes	Rating	Tips	Cluster
0	4a5a4eb2f964a52021ba1fe3	40.792027	-73.959853	109	9.0	6	3
1	4f3c0584e4b0f7c8c775c07e	40.789188	-73.957867	8	8.0	0	4
2	4c841c2ed8086dcb246f8652	40.787786	-73.955924	25	8.5	3	0
3	4b67aad0f964a520265a2be3	40.791591	-73.964795	33	8.6	2	0
4	4d6331414554a0934064afaa	40.788791	-73.955232	16	7.8	1	4

Figure 3 Original Data

```
array([[ 0.87946493,  0.54902537, -0.30578036,  0.7570733 , -0.31312928],
       [ 0.22156757,  0.9432769 , -0.32212336, -0.8758299 , -0.32470717],
       [-0.10341175,  1.32893295, -0.31937256, -0.0593783 , -0.31891823],
       [ 0.77845491, -0.43207911, -0.31807806,  0.10391202, -0.32084787],
       [ 0.12964712,  1.4663409 , -0.32082886, -1.20241054, -0.32277752],
       [-0.96133935, -0.44372359,  3.16217223,  1.90010554,  3.16216719],
       [-0.6523277 , -1.3700646 , -0.29429171,  1.24694426, -0.2938328 ],
       [-0.37045159, -1.15678125, -0.31937256, -0.38595894, -0.31119963],
       [ 2.31841576,  0.5399093 , -0.31969618,  0.10391202, -0.31312928],
       [-0.82579961, -1.41498389, -0.32099067,  0.10391202, -0.32470717],
       [-1.41422029, -0.00985296, -0.32163792, -1.6922815 , -0.31891823]])
```

Figure 4 Normalized Data

3. Exploratory Data Analysis

Install and import all required decencies and packages at first. Below list shows the main ones:

- Pandas and NumPy for handling data
- Request to create foursquare API calls
- Foluim to create map of New York city and visualize clusters
- Geopy to generate coordinate of the New York city
- Sklearn to apply k-means clustering

```
In [2]: %pip install numpy
        %pip install pandas
        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

        %pip install requests
        import requests # library to handle requests
        from pandas.io.json import json_normalize # tranform JSON file into a pandas dataframe
        %pip install ipython

        %pip install folium
        %pip install geopy
        # import k-means from clustering stage
        %pip install sklearn
        from sklearn.cluster import KMeans
        import folium
        from geopy.geocoders import Nominatim

        print('Libraries imported.')
```

Now, define the function that will be used to generate the address(Latitude and Longitude) of New York city

Define function to get address , to explore venues

```
In [3]: def geo_location(address):
            # get geo location of address
            geolocator = Nominatim(user_agent="ny_explorer")
            location = geolocator.geocode(address)
            latitude = location.latitude
            longitude = location.longitude
            return latitude,longitude
```

Similiraly, define the function that retrive the venues details(rating, tips,likes) for a given venue id by using FourSquare API

```
In [4]: def get_venue_details(venue_id):
            radius=1000
            LIMIT=400
            CLIENT_ID = 'GLX05FUL3DLD2BPRN2CZPCRTJT1FE22A43NTZGHRYV4WG1ST' # your Foursquare ID
            CLIENT_SECRET = '3FRSXZKOPLBJCRLSEQ5DMNEJVB3LGWTYW2BRROXRUIOWUZVP' # your Foursquare Secret
            VERSION = '20180605' # Foursquare API version
            url = 'https://api.foursquare.com/v2/venues/{}?&client_id={}&client_secret={}&v={}'.format(
                        venue_id,
                        CLIENT_ID,
                        CLIENT_SECRET,
                        VERSION)
                # get all the data
            results = requests.get(url).json()
            venue_data=results['response']['venue']
            venue_details=[]
            for row in venue_data:
                try:
                    venue_id=venue_data['id']
                    venue_name=venue_data['name']
                    venue_likes=venue_data['likes']['count']
                    venue_rating=venue_data['rating']
                    venue_tips=venue_data['tips']['count']
                    venue_details.append([venue_id,venue_name,venue_likes,venue_rating,venue_tips])
                except KeyError:
                        pass
            column_names=['ID','Name','Likes','Rating','Tips']
            df9 = pd.DataFrame(venue_details,columns=column_names)
            return df9
```

Now define the function that will retrieve the New York data from the JSON file

```
In [5]: def get_new_york_data():
    url='https://cocl.us/new_york_dataset'
    resp=requests.get(url).json()
    # all data is present in features label
    features=resp['features']

    # define the dataframe columns
    column_names = ['Borough', 'Neighborhood', 'Latitude', 'Longitude']
    # instantiate the dataframe
    new_york_data = pd.DataFrame(columns=column_names)

    for data in features:
        borough = data['properties']['borough']
        neighborhood_name = data['properties']['name']

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

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

    return new_york_data
```

Calling the above function to get New York data and store them in dataframe

```
In [6]: # get new york data
new_york_data=get_new_york_data()
```

```
In [97]: new_york_data.head(20)
```

Out[97]:

	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
5	Bronx	Kingsbridge	40.881687	-73.902818
6	Manhattan	Marble Hill	40.876551	-73.910660
7	Bronx	Woodlawn	40.898273	-73.867315
8	Bronx	Norwood	40.877224	-73.879391
9	Bronx	Williamsbridge	40.881039	-73.857446
10	Bronx	Baychester	40.866858	-73.835798
11	Bronx	Pelham Parkway	40.857413	-73.854756
12	Bronx	City Island	40.847247	-73.786488

Filter the result to get specific rows (Manhattan only) , we have 40 different neighborhoods on Manhattan boroughs.

now filter the result to select Manhatten data only

```
In [8]: manhatten_df=new_york_data[new_york_data.Borough=="Manhattan"]
manhatten_df.head()
```

Out[8]:

	Borough	Neighborhood	Latitude	Longitude
6	Manhattan	Marble Hill	40.876551	-73.910660
100	Manhattan	Chinatown	40.715618	-73.994279
101	Manhattan	Washington Heights	40.851903	-73.936900
102	Manhattan	Inwood	40.867684	-73.921210
103	Manhattan	Hamilton Heights	40.823604	-73.949688

```
In [9]: manhatten_df.shape
```

```
Out[9]: (40, 4)
```

Next, creating the map of New York city with neighborhoods on the pervious dataframe on top.

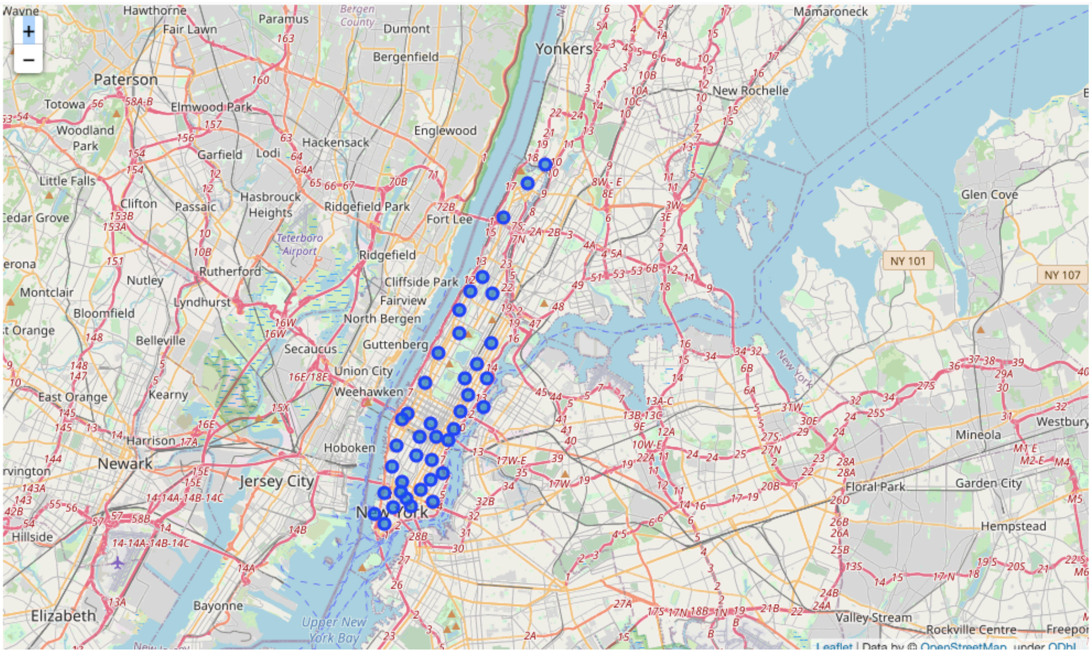
Create map of NYC and display Manhatten neighborhoods

```
In [11]: #map of new your with 2 boroghts
         map_newyork = folium.Map(location=geo_location("New York City, NY"), zoom_start=10)

         # add markers to map
         for lat, lng, borough, neighborhood in zip(manhatten_df['Latitude'], manhatten_df['Longitude'], manhatten_df['Borough']
             label = '{}, {}'.format(neighborhood, borough)
             label = folium.Popup(label, parse_html=True)
             folium.CircleMarker(
                 [lat, lng],
                 radius=5,
                 popup=label,
                 color='blue',
                 fill=True,
                 fill_color='#3186cc',
                 fill_opacity=0.7,
                 parse_html=False).add_to(map_newyork)

         map_newyork
```

Out[11]:



Then, getting all venues on all neighborhoods of Manhattan using Foursquare API

NOW we get the venues on Manahtten using FourSqaure API Calls

In [13]:
```
# manhatten venues
radius=1000
LIMIT=400
CLIENT_ID = 'GLX05FUL3DLD2BPRN2CZPCRTJT1FE22A43NTZGHRYV4WG1ST' # your Foursquare ID
CLIENT_SECRET = '3FRSXZKOPLBJCRLSEQ5DMNEJVB3LGWTYW2BRROXRUIOWUZVP' # your Foursquare Secret
VERSION = '20180605' # Foursquare API version
url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={},{}&radius={}&limit={}'.fc
            CLIENT_ID,
            CLIENT_SECRET,
            VERSION,
            address2[0],
            address2[1],
            radius,
            LIMIT)

    # get all the data
results = requests.get(url).json()

venue_data=results["response"]['groups'][0]['items']
venue_details=[]
for row in venue_data:
    try:
            venue_id=row['venue']['id']
            venue_name=row['venue']['name']
            venue_category=row['venue']['categories'][0]['name']
            venue_lat=row['venue']['location']['lat']
            ven_lan=row['venue']['location']['lng']
            venue_details.append([venue_id,venue_name,venue_category,venue_lat,ven_lan])
    except KeyError:
            pass
column_names=['ID','Name','Category','Lat','Lan']
df = pd.DataFrame(venue_details,columns=column_names)
```

Result is a dataframe with all venues

In [14]:
```
df.head()
```

Out[14]:

	ID	Name	Category	Lat	Lan
0	4a78425df964a52053e51fe3	Central Park Tennis Center	Tennis Court	40.789313	-73.961862
1	4a5a4eb2f964a52021ba1fe3	North Meadow	Park	40.792027	-73.959853
2	4ba233dbf964a5206fe337e3	East Meadow	Field	40.790160	-73.955498
3	4f3c0584e4b0f7c8c775c07e	Oldest Tree in Central Park	Park	40.789188	-73.957867
4	4c841c2ed8086dcb246f8652	Central Park - Woodman's Gate	Park	40.787786	-73.955924

Find the number of unique category on the venues dataframe

let's find unique catagory

In [17]:
```
print('There are {} uniques categories.'.format(len(df['Category'].unique())))
df.groupby('Category')['Category'].count().sort_values(ascending=False)
```

```
There are 48 uniques categories.
```

Out[17]:
```
Category
Park                   11
Playground              8
Café                    6
Art Museum              4
Grocery Store           4
Baseball Field          4
Wine Shop               4
Fountain                3
Garden                  3
Scenic Lookout          3
Pizza Place             3
Gym                     2
History Museum          2
Italian Restaurant      2
Coffee Shop             2
Gym / Fitness Center    2
Plaza                   2
```

Now get the parks of Manhattan on separate dataframe from the venues dataframe. As it is shown on the above output, we have 11 parks of different neighborhoods.

```
In [19]: manhatten_parks=df[df.Category=="Park"]
         manhatten_parks.head()
```

Out[19]:

	ID	Name	Category	Lat	Lan
1	4a5a4eb2f964a52021ba1fe3	North Meadow	Park	40.792027	-73.959853
3	4f3c0584e4b0f7c8c775c07e	Oldest Tree in Central Park	Park	40.789188	-73.957867
4	4c841c2ed8086dcb246f8652	Central Park - Woodman's Gate	Park	40.787786	-73.955924
7	4b67aad0f964a520265a2be3	Central Park - Gate Of All Saints	Park	40.791591	-73.964795
17	4d6331414554a0934064afaa	Central Park - 99th & 5th Ave	Park	40.788791	-73.955232

```
In [20]: manhatten_parks.shape

Out[20]: (11, 5)
```

Find more details about each park such as tips, ratings and likes by calling function defined above get_venue_details and passing the ID .

Now getting the venues details by FourSquare

```
In [22]: colomns=['ID','Name','Likes','Rating','Tips']
         all_details_df=pd.DataFrame(columns=colomns)
         for data in manhatten_parks.values.tolist():

             ID,Name,Category,Lat,Lan=data

             details= get_venue_details(ID)
             id,name,likes,rating,tips=details.values.tolist()[0]

             all_details_df=all_details_df.append({'ID':id,
                                                   'Name':Name,
                                                   'Likes':likes,
                                                   'Rating':rating,
                                                   'Tips':tips},ignore_index=True)
```

Display the details dataframe

```
In [23]: all_details_df.head(20)
```

Out[23]:

	ID	Name	Likes	Rating	Tips
0	4a5a4eb2f964a52021ba1fe3	North Meadow	109	9.0	6
1	4f3c0584e4b0f7c8c775c07e	Oldest Tree in Central Park	8	8.0	0
2	4c841c2ed8086dcb246f8652	Central Park - Woodman's Gate	25	8.5	3
3	4b67aad0f964a520265a2be3	Central Park - Gate Of All Saints	33	8.6	2
4	4d6331414554a0934064afaa	Central Park - 99th & 5th Ave	16	7.8	1
5	412d2800f964a520df0c1fe3	Central Park	21541	9.7	1807
6	4c531929479fc9282b90ee90	Central Park West - W 86th St	180	9.3	16
7	4bb622b846d4a59345bdc5c0	Central Park West - W 88th St	25	8.3	7
8	4b59c10af964a520469628e3	Central Park - Strangers' Gate	23	8.6	6
9	4c3f76b63735be9abe6a15a4	Central Park - Mariners' Gate	15	8.6	0
10	4de802a8b0fbcb7c9ab8c21f	Central Park - South Gate House	11	7.5	3

Now let's sort the above dataframe by rating

Sort the result

In [24]:
```
all_details_df.sort_values(by=['Rating'], inplace=True,ascending=False)
```

In [25]:
```
all_details_df.head(10)
```

Out[25]:

	ID	Name	Likes	Rating	Tips
5	412d2800f964a520df0c1fe3	Central Park	21541	9.7	1807
6	4c531929479fc9282b90ee90	Central Park West - W 86th St	180	9.3	16
0	4a5a4eb2f964a52021ba1fe3	North Meadow	109	9.0	6
3	4b67aad0f964a520265a2be3	Central Park - Gate Of All Saints	33	8.6	2
8	4b59c10af964a520469628e3	Central Park - Strangers' Gate	23	8.6	6
9	4c3f76b63735be9abe6a15a4	Central Park - Mariners' Gate	15	8.6	0
2	4c841c2ed8086dcb246f8652	Central Park - Woodman's Gate	25	8.5	3
7	4bb622b846d4a59345bdc5c0	Central Park West - W 88th St	25	8.3	7
1	4f3c0584e4b0f7c8c775c07e	Oldest Tree in Central Park	8	8.0	0
4	4d6331414554a0934064afaa	Central Park - 99th & 5th Ave	16	7.8	1

Merge the sorted dataframe to get Latitude and Longitude of each park

Now let's join the two dataframes

In [30]:
```
#join dataframes

result_df = pd.merge(manhatten_parks, all_details_df,how='inner', on=['ID','Name'])
result_df
```

Out[30]:

	ID	Name	Category	Lat	Lan	Likes	Rating	Tips
0	4a5a4eb2f964a52021ba1fe3	North Meadow	Park	40.792027	-73.959853	109	9.0	6
1	4f3c0584e4b0f7c8c775c07e	Oldest Tree in Central Park	Park	40.789188	-73.957867	8	8.0	0
2	4c841c2ed8086dcb246f8652	Central Park - Woodman's Gate	Park	40.787786	-73.955924	25	8.5	3
3	4b67aad0f964a520265a2be3	Central Park - Gate Of All Saints	Park	40.791591	-73.964795	33	8.6	2
4	4d6331414554a0934064afaa	Central Park - 99th & 5th Ave	Park	40.788791	-73.955232	16	7.8	1
5	412d2800f964a520df0c1fe3	Central Park	Park	40.784083	-73.964853	21541	9.7	1807
6	4c531929479fc9282b90ee90	Central Park West - W 86th St	Park	40.785417	-73.969519	180	9.3	16
7	4bb622b846d4a59345bdc5c0	Central Park West - W 88th St	Park	40.786633	-73.968445	25	8.3	7
8	4b59c10af964a520469628e3	Central Park - Strangers' Gate	Park	40.798237	-73.959899	23	8.6	6
9	4c3f76b63735be9abe6a15a4	Central Park - Mariners' Gate	Park	40.784668	-73.969746	15	8.6	0
10	4de802a8b0fbcb7c9ab8c21f	Central Park - South Gate House	Park	40.782129	-73.962668	11	7.5	3

Now let's prepare the above dataframe for clustering by dropping all categorical columns

Now let's drop the catagory,name colomns to apply K-mean clustering

In [59]:
```
cluster_df=result_df.drop('Name',axis=1)
cluster_df.head()
```

Out[59]:

	ID	Category	Lat	Lan	Likes	Rating	Tips
0	4a5a4eb2f964a52021ba1fe3	Park	40.792027	-73.959853	109	9.0	6
1	4f3c0584e4b0f7c8c775c07e	Park	40.789188	-73.957867	8	8.0	0
2	4c841c2ed8086dcb246f8652	Park	40.787786	-73.955924	25	8.5	3
3	4b67aad0f964a520265a2be3	Park	40.791591	-73.964795	33	8.6	2
4	4d6331414554a0934064afaa	Park	40.788791	-73.955232	16	7.8	1

In [64]:
```
cluster_df=cluster_df.drop('Category',axis=1)
cluster_df.head()
```

Out[64]:

	ID	Lat	Lan	Likes	Rating	Tips	Cluster
0	4a5a4eb2f964a52021ba1fe3	40.792027	-73.959853	109	9.0	6	3
1	4f3c0584e4b0f7c8c775c07e	40.789188	-73.957867	8	8.0	0	4
2	4c841c2ed8086dcb246f8652	40.787786	-73.955924	25	8.5	3	0
3	4b67aad0f964a520265a2be3	40.791591	-73.964795	33	8.6	2	0
4	4d6331414554a0934064afaa	40.788791	-73.955232	16	7.8	1	4

Then , normalize the numerical columns using StandardScaler function

Pre-process the data to apply k-mean clustering on Numerical features

In [67]:
```
from sklearn.preprocessing import StandardScaler
X = cluster_df.values[:,1:]
X = np.nan_to_num(X)
Clus_dataSet = StandardScaler().fit_transform(X)
Clus_dataSet
```

Out[67]:
```
array([[ 0.87946493,  0.54902537, -0.30578036,  0.7570733 , -0.31312928],
       [ 0.22156757,  0.9432769 , -0.32212336, -0.8758299 , -0.32470717],
       [-0.10341175,  1.32893295, -0.31937256, -0.0593783 , -0.31891823],
       [ 0.77845491, -0.43207911, -0.31807806,  0.10391202, -0.32084787],
       [ 0.12964712,  1.4663409 , -0.32082886, -1.20241054, -0.32277752],
       [-0.96133935, -0.44372359,  3.16217223,  1.90010554,  3.16216719],
       [-0.6523277 , -1.3700646 , -0.29429171,  1.24694426, -0.2938328 ],
       [-0.37045159, -1.15678125, -0.31937256, -0.38595894, -0.31119963],
       [ 2.31841576,  0.5399093 , -0.31969618,  0.10391202, -0.31312928],
       [-0.82579961, -1.41498389, -0.32099067,  0.10391202, -0.32470717],
       [-1.41422029, -0.00985296, -0.32163792, -1.6922815 , -0.31891823]])
```

Start K-Mean clustering on the normalized data to group the parks with similar properties on clusters and then apply cluster labels to all parks

In [68]:
```
clusterNum = 5
k_means = KMeans(init = "k-means++", n_clusters = clusterNum, n_init = 12)
k_means.fit(X)
labels = k_means.labels_
print(labels)
```

```
[3 0 4 4 0 1 2 4 4 0 0]
```

In [69]:
```
cluster_df['Cluster'] = labels
cluster_df.head(10)
```

Out[69]:

	ID	Lat	Lan	Likes	Rating	Tips	Cluster
0	4a5a4eb2f964a52021ba1fe3	40.792027	-73.959853	109	9.0	6	3
1	4f3c0584e4b0f7c8c775c07e	40.789188	-73.957867	8	8.0	0	0
2	4c841c2ed8086dcb246f8652	40.787786	-73.955924	25	8.5	3	4
3	4b67aad0f964a520265a2be3	40.791591	-73.964795	33	8.6	2	4
4	4d6331414554a0934064afaa	40.788791	-73.955232	16	7.8	1	0
5	412d2800f964a520df0c1fe3	40.784083	-73.964853	21541	9.7	1807	1
6	4c531929479fc9282b90ee90	40.785417	-73.969519	180	9.3	16	2
7	4bb622b846d4a59345bdc5c0	40.786633	-73.968445	25	8.3	7	4
8	4b59c10af964a520469628e3	40.798237	-73.959899	23	8.6	6	4
9	4c3f76b63735be9abe6a15a4	40.784668	-73.969746	15	8.6	0	0

Finally , lets visualize the clustering result

In [82]:

```
# create map
map_clusters = folium.Map(location=[address2[0], address2[1]], zoom_start=10)

# set color scheme for the clusters
colors_array = cm.rainbow(np.linspace(0, 1, clusterNum))
rainbow = [colors.rgb2hex(i) for i in colors_array]

# add markers to the map
markers_colors = []
for id,lat, lon, cluster in zip(cluster_df['ID'],cluster_df['Lat'], cluster_df['Lan'],cluster_df['Cluster']):
    label = folium.Popup(str(id) + ' Cluster ' + str(cluster), parse_html=True)
    folium.CircleMarker(
        [lat, lon],
        radius=5,
        popup=label,
        color=rainbow[cluster-1],
        fill=True,
        fill_color=rainbow[cluster-1],
        fill_opacity=0.7).add_to(map_clusters)

map_clusters
```

Out[82]:

4. Result and Discussion

From the analysis , we notice that parks are the most frequent type of venues on Manhattan. As well as, Central Park is the best park on Manhattan which located on Harlem neighborhood and it should be a distance for the neighborhood visitors.
From the clustering , it is clear that Central park is the only one on its cluster so the recommendation for government agency is to open a similar park on other neighborhood with low rated parks.

5. Conclusion

In conclusion, this report is written to explain the complete project on examining Manhattan neighborhoods and clustering its highly frequent venues which are parks. The result of this study would be useful for NYC-Manhattan visitors and government when they plan to open a new park on Manhattan.