Segmenting and Clustering Neighborhoods in New York City & London

In [1]:
```
#install packages
!pip install numpy
!pip install pandas
!pip install requests
!pip install bs4
!pip install plotly
!conda install -c conda-forge geopy --yes
!conda install -c conda-forge folium=0.5.0 --yes
!pip install html5lib
!pip install OSGridConverter
print('Packages installed.')
```

```
Requirement already satisfied: requests in /srv/conda/envs/notebook/lib/python3.6/site-packages (2.25.
1)
Requirement already satisfied: certifi>=2017.4.17 in /srv/conda/envs/notebook/lib/python3.6/site-packa
ges (from requests) (2020.12.5)
Requirement already satisfied: chardet<5,>=3.0.2 in /srv/conda/envs/notebook/lib/python3.6/site-packag
es (from requests) (4.0.0)
Requirement already satisfied: idna<3,>=2.5 in /srv/conda/envs/notebook/lib/python3.6/site-packages (f
rom requests) (2.10)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /srv/conda/envs/notebook/lib/python3.6/site-pa
ckages (from requests) (1.26.3)
Collecting bs4
  Downloading bs4-0.0.1.tar.gz (1.1 kB)
Collecting beautifulsoup4
  Downloading beautifulsoup4-4.9.3-py3-none-any.whl (115 kB)
     |████████████████████████████████| 115 kB 4.5 MB/s eta 0:00:01

Collecting soupsieve>1.2
  Downloading soupsieve-2.2.1-py3-none-any.whl (33 kB)
Building wheels for collected packages: bs4
  Building wheel for bs4 (setup.py) ... done
```

In [1]:
```
#import libraries
import numpy as np
import pandas as pd
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
import json
import requests
from pandas.io.json import json_normalize
from bs4 import BeautifulSoup
import plotly.graph_objects as go
from plotly.subplots import make_subplots
from geopy.geocoders import Nominatim
from OSGridConverter import grid2latlong
import folium
import matplotlib.cm as cm
import matplotlib.colors as colors
from sklearn.cluster import KMeans
print('Libraries imported.')
```

```
Libraries imported.
```

Set Up Foursquare Credentials and Venue Functions

In [2]:
```
#input Foursquare credentials
CLIENT_ID = 'YTE1O5DN3TRLBFJR2V4CMOU0SNG2AF3XQA0CGFA4KA4SFAOK' # your Foursquare ID
CLIENT_SECRET = 'SRMIKFL2VBEBVO3ADRTIEDTRAB5OPAS4HOPTF00PXTOVXWT3' # your Foursquare Secret
VERSION = '20180605' # Foursquare API version
LIMIT = 100 # A default Foursquare API limit value
```

In [8]:
```
ction to get the top 100 venues that for given neighborhood within a radius of 750 meters

getNearbyVenues(names, latitudes, longitudes, radius= 750):

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={},{}&rad
        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)
```

In [9]:
```
#function to sort the venues in descending order
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]
```

NYC : Download Data from IBM Server and Transform into Pandas Dataframe

In [10]:
```
ver
ps://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DS0701EN-Skill
```

In [11]:
```
#load data
with open('newyork_data.json') as json_data:
    newyork_data = json.load(json_data)
```

In [12]:
```
#get a feel for how the data is structured
newyork_data
```

Out[12]:

```
{'type': 'FeatureCollection',
 'totalFeatures': 306,
 'features': [{'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]}},
  {'type': 'Feature',
```

In [13]:
```
#all the relevant data is located within "features"
nyc_neighborhoods_data = newyork_data['features']
```

In [14]:
```
# define the dataframe columns
column_names = ['City','Borough', 'Neighborhood', 'Latitude', 'Longitude']

# instantiate the dataframe (creating an empty dataframe)
nyc_neighborhoods = pd.DataFrame(columns=column_names)
```

In [15]:
```
#transform data into pandas dataframe
for data in nyc_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]
    city = 'New York City'

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

In [16]:
```
#populated dataframe with nyc data
nyc_neighborhoods.head()
```

Out[16]:

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

In [17]:
```
#merge neighborhood and borough into one column
nyc_neighborhoods['Neighborhood'] = nyc_neighborhoods['Neighborhood'] + " (" + nyc_neighborhoods['Borou
nyc_neighborhoods.drop(columns = 'Borough', inplace=True)
nyc_neighborhoods.head()
```

Out[17]:

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

In [18]:
```
#number of columns and rows
nyc_neighborhoods.shape
```

Out[18]:
```
(306, 4)
```

London: Webscrape Data from Wiki Page and Transform into Pandas Dataframe

In [19]:
```
#download webpage and save text in the html_data variable
london_url = "https://en.wikipedia.org/wiki/List_of_areas_of_London"
html_data = requests.get(london_url).text
```

In [20]:
```
#parse through the html data
soup = BeautifulSoup(html_data,"html.parser")
```

In [21]:
```
#find all tables
tables = soup.find_all('table')
```

In [22]:
```
reate empty list to store cleansed data
ble_contents=[]

xtract the html data and assign it to the corresponding column
w_num = 0
r row in tables[1].find("tbody").find_all("tr"):
  if row_num > 0: #want to skip the first row as that contains a heading (refine later so that there's n
      cell = {} #create a dictionary to hold all record values
      col = row.find_all('td')
      remove_tail_borough = col[1].text.split('[')
      cell['Borough']= remove_tail_borough[0]
      cell['Neighborhood'] = col[0].text
      cell['PostTown'] = col[2].text
      cell['OSGridRef'] = col[5].text.replace('\n','')
      if cell['OSGridRef'] == '': #use the OS grid to find the coordinates
          cell['Latitude'] = 0
          cell['Longitude'] = 0
      else:
          l=grid2latlong(cell['OSGridRef'])
          cell['Latitude'] = l.latitude
          cell['Longitude'] = l.longitude
      table_contents.append(cell) #consolidate all dictionaries into a list
  row_num = row_num + 1
=pd.DataFrame(table_contents) #convert list into dataframe

rint dataframe
.head()
```

Out[22]:

	Borough	Neighborhood	PostTown	OSGridRef	Latitude	Longitude
0	Bexley, Greenwich	Abbey Wood	LONDON	TQ465785	51.486484	0.109318
1	Ealing, Hammersmith and Fulham	Acton	LONDON	TQ205805	51.510591	-0.264585
2	Croydon	Addington	CROYDON	TQ375645	51.362934	-0.025780
3	Croydon	Addiscombe	CROYDON	TQ345665	51.381625	-0.068126
4	Bexley	Albany Park	BEXLEY, SIDCUP	TQ478728	51.434929	0.125663

In [23]:
```
#number of columns and rows
df.shape
```

Out[23]: (531, 6)

In [24]:
```
#filter results for PostTown with London only (records with multiple PostTowns will NOT be included)
london_neighborhoods = df[df['PostTown'] == 'LONDON'].reset_index(drop=True)
london_neighborhoods.head()
```

Out[24]:

	Borough	Neighborhood	PostTown	OSGridRef	Latitude	Longitude
0	Bexley, Greenwich	Abbey Wood	LONDON	TQ465785	51.486484	0.109318
1	Ealing, Hammersmith and Fulham	Acton	LONDON	TQ205805	51.510591	-0.264585
2	City	Aldgate	LONDON	TQ334813	51.514885	-0.078356
3	Westminster	Aldwych	LONDON	TQ307810	51.512819	-0.117388
4	Bromley	Anerley	LONDON	TQ345695	51.408585	-0.066989

In [25]:
```
#number of columns and rows
london_neighborhoods.shape
```

Out[25]: (297, 6)

In [26]:
```
#format the London dataframe to match the NYC dataframe so that we can merge them

london_neighborhoods.drop(['PostTown'], axis=1, inplace = True)#drop PostTown column since all of them a
london_neighborhoods.drop(['OSGridRef'], axis=1, inplace = True)#drop OSGridRef since we already extract
london_neighborhoods['City'] = 'London' #add city column to help distinguish between NYC and London
cols = london_neighborhoods.columns.tolist() #current order of columns
cols = cols[-1:] + cols[:-1] #move City column to the front
london_neighborhoods[cols].head()
```

Out[26]:

	City	Borough	Neighborhood	Latitude	Longitude
0	London	Bexley, Greenwich	Abbey Wood	51.486484	0.109318
1	London	Ealing, Hammersmith and Fulham	Acton	51.510591	-0.264585
2	London	City	Aldgate	51.514885	-0.078356
3	London	Westminster	Aldwych	51.512819	-0.117388
4	London	Bromley	Anerley	51.408585	-0.066989

In [27]:
```
#merge neighborhood and borough into one column
london_neighborhoods['Neighborhood'] = london_neighborhoods['Neighborhood'] + " (" + london_neighborhoods
london_neighborhoods.drop(columns = 'Borough', inplace=True)
london_neighborhoods.head()
```

Out[27]:

	Neighborhood	Latitude	Longitude	City
0	Abbey Wood (Bexley, Greenwich)	51.486484	0.109318	London
1	Acton (Ealing, Hammersmith and Fulham)	51.510591	-0.264585	London
2	Aldgate (City)	51.514885	-0.078356	London
3	Aldwych (Westminster)	51.512819	-0.117388	London
4	Anerley (Bromley)	51.408585	-0.066989	London

In [28]:
```
#verify the number of columns and rows
london_neighborhoods.shape
```

Out[28]: (297, 4)

In [29]:
```
#merge the NYC dataframe and London dataframe
nyc_london_neighborhoods = pd.concat([nyc_neighborhoods, london_neighborhoods]).reset_index(drop=True)
nyc_london_neighborhoods.tail()
```

Out[29]:

	City	Neighborhood	Latitude	Longitude
598	London	Wood Green (Haringey)	51.598237	-0.116745
599	London	Woodford (Redbridge)	51.604820	0.028068
600	London	Woodside Park (Barnet)	51.617324	-0.186791
601	London	Woolwich (Greenwich)	51.496238	0.066504
602	London	Wormwood Scrubs (Hammersmith and Fulham)	51.519148	-0.235411

In [30]:
```
#verify the number of columns and rows
nyc_london_neighborhoods.shape
```

Out[30]:
```
(603, 4)
```

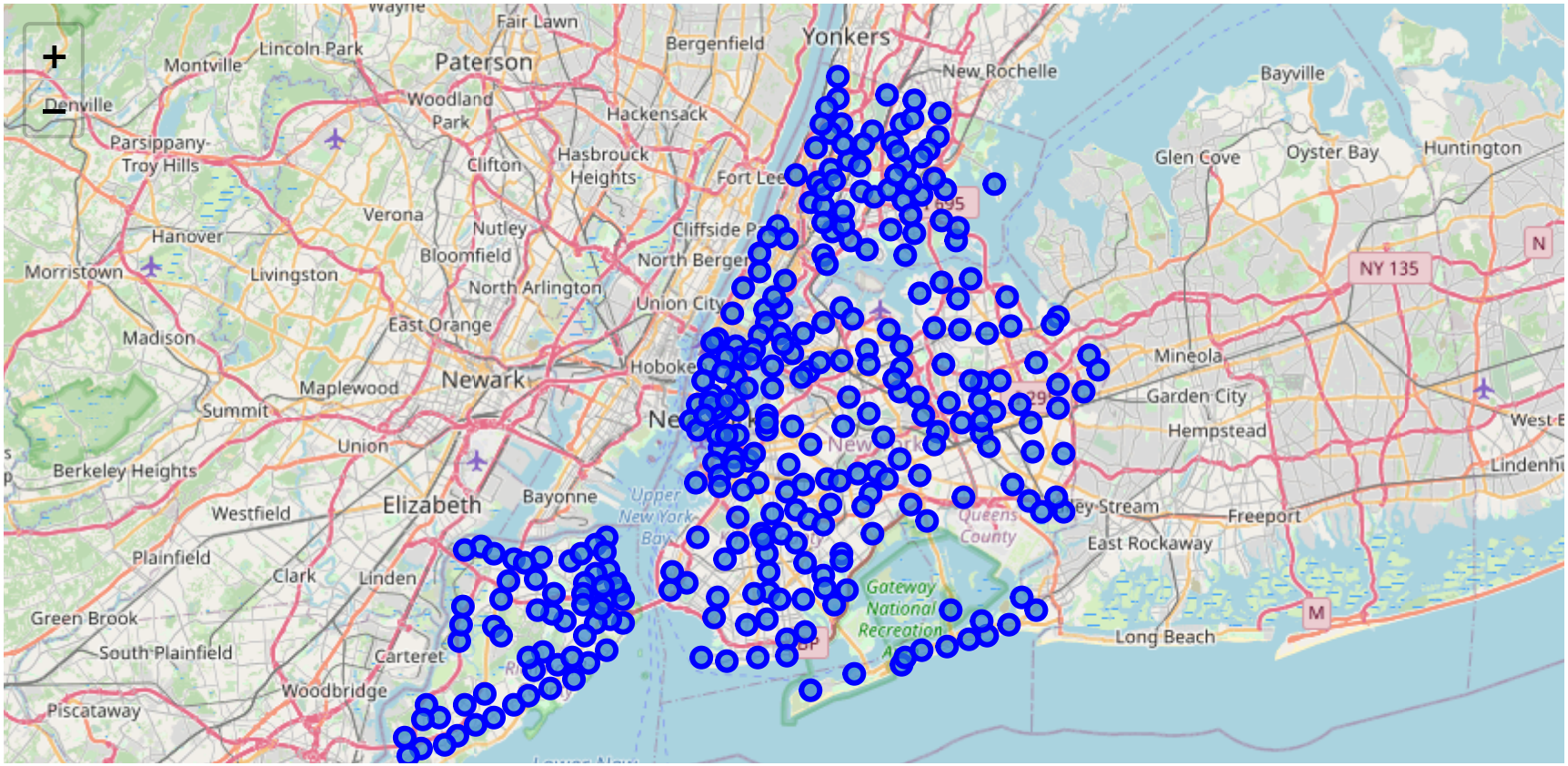
Use Folium Map to Visualize Neighborhoods in NYC and London

In [31]:
```
#obtain geographic coordinates of NYC
nyc_address = 'New York City, NY'

nyc_geolocator = Nominatim(user_agent="ny_explorer")
nyc_location = nyc_geolocator.geocode(nyc_address)
nyc_latitude = nyc_location.latitude
nyc_longitude = nyc_location.longitude
print('The geograpical coordinate of New York City are {}, {}.'.format(nyc_latitude, nyc_longitude))
```

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

In [37]:
```
# create map of NYC
map_nyc = folium.Map(location=[nyc_latitude, nyc_longitude], zoom_start=10)

# add markers to map
for lat, lng, neighborhood in zip(nyc_london_neighborhoods['Latitude'], nyc_london_neighborhoods['Longit
    label = '{}'.format(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_nyc)

map_nyc
```

Out[37]:

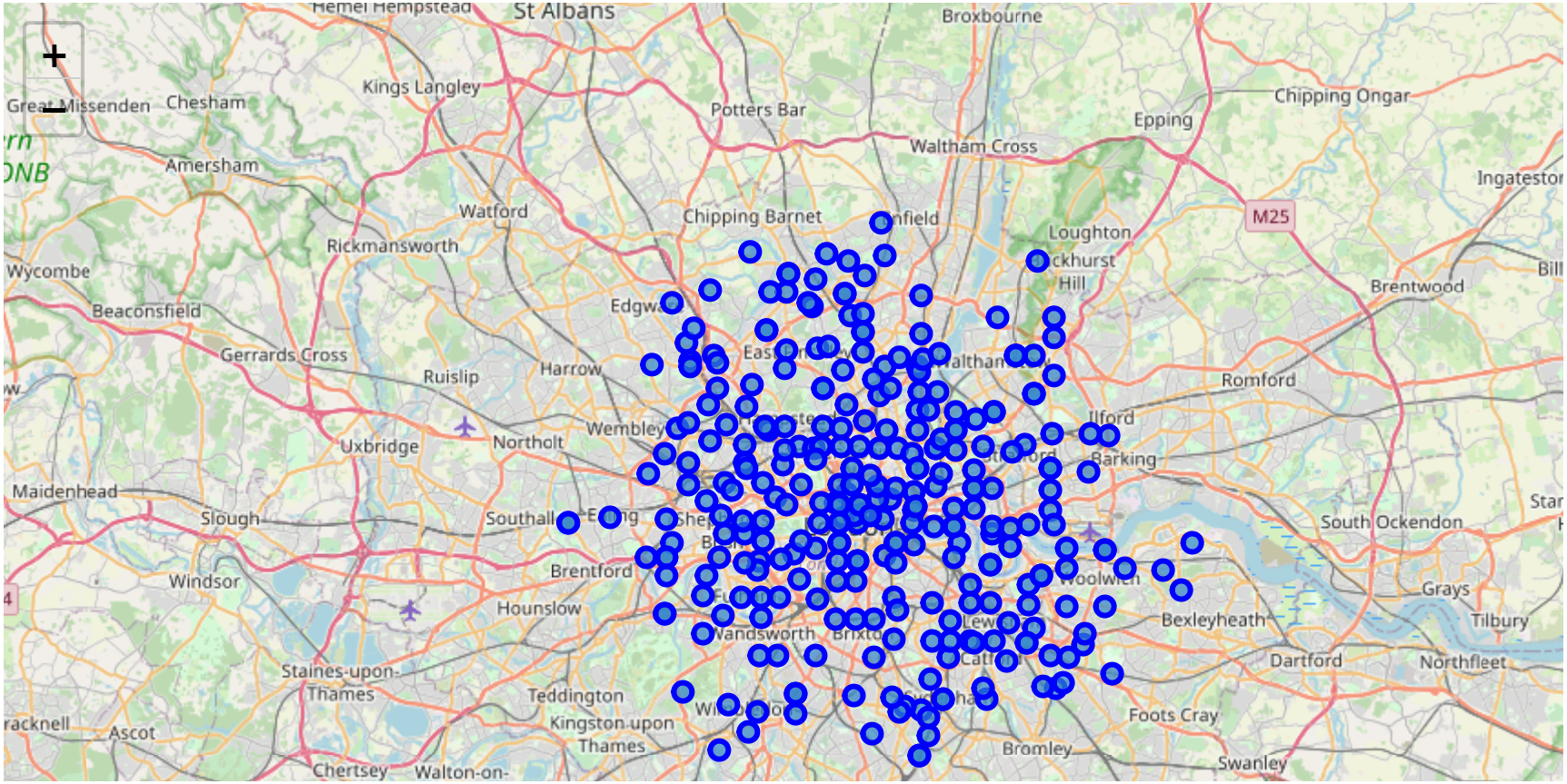


In [32]:
```
#obtain geographic coordinates of London
london_address = 'London, United Kingdom'

london_geolocator = Nominatim(user_agent="uk_explorer")
london_location = london_geolocator.geocode(london_address)
london_latitude = london_location.latitude
london_longitude = london_location.longitude
print('The geograpical coordinate of London are {}, {}.'.format(london_latitude, london_longitude))
```

```
The geograpical coordinate of London are 51.5073219, -0.1276474.
```

In [38]:
```
# create map of London
map_london = folium.Map(location=[london_latitude, london_longitude], zoom_start=10)

# add markers to map
for lat, lng, neighborhood in zip(nyc_london_neighborhoods['Latitude'], nyc_london_neighborhoods['Longit
    label = '{}'.format(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_london)

map_london
```

Out[38]:



Use Foursquare to Identify Top Venues in Each Neighborhood

In [41]:
```
#run the above function on each neighborhood
nyc_london_venues = getNearbyVenues(names=nyc_london_neighborhoods['Neighborhood'],
                                    latitudes=nyc_london_neighborhoods['Latitude'],
                                    longitudes=nyc_london_neighborhoods['Longitude']
                                   )
```

```
Wakefield (Bronx)
Co-op City (Bronx)
Eastchester (Bronx)
Fieldston (Bronx)
Riverdale (Bronx)
Kingsbridge (Bronx)
Marble Hill (Manhattan)
Woodlawn (Bronx)
Norwood (Bronx)
Williamsbridge (Bronx)
Baychester (Bronx)
Pelham Parkway (Bronx)
City Island (Bronx)
Bedford Park (Bronx)
University Heights (Bronx)
Morris Heights (Bronx)
Fordham (Bronx)
East Tremont (Bronx)
West Farms (Bronx)
```

In [42]:
```
nyc_london_venues.shape
```

Out[42]:
```
(29507, 7)
```

In [43]:
```
#group the venues by neighborhood
nyc_london_venues.groupby('Neighborhood').count()
```

Out[43]:

Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
Abbey Wood (Bexley, Greenwich)	4	4	4	4	4	4
Acton (Ealing, Hammersmith and Fulham)	30	30	30	30	30	30
Aldgate (City)	100	100	100	100	100	100
Aldwych (Westminster)	100	100	100	100	100	100
Allerton (Bronx)	36	36	36	36	36	36
Anerley (Bromley)	12	12	12	12	12	12
Angel (Islington)	17	17	17	17	17	17
Annadale (Staten Island)	13	13	13	13	13	13
Archway (Islington)	33	33	33	33	33	33

In [44]:
```
#unique categories for venues
print('There are {} uniques categories.'.format(len(nyc_london_venues['Venue Category'].unique())))
```

```
There are 525 uniques categories.
```

In [45]:
```
#Analyze Each Neighborhood

# one hot encoding
nyc_london_onehot = pd.get_dummies(nyc_london_venues[['Venue Category']], prefix="", prefix_sep="")

# add neighborhood column back to dataframe
nyc_london_onehot['Neighborhood']= nyc_london_venues['Neighborhood']
neighborhood_col_index = nyc_london_onehot.columns.get_loc("Neighborhood")

#move neighborhood column to the first column
fixed_columns = [nyc_london_onehot.columns[neighborhood_col_index]] + list(nyc_london_onehot.columns[0:r
nyc_london_onehot = nyc_london_onehot[fixed_columns]


print("The unique venue categories have now become columns.")
print("Therefore the column count should equal the number of unique categories:", len(nyc_london_onehot.

nyc_london_onehot.head()
```

```
The unique venue categories have now become columns.
Therefore the column count should equal the number of unique categories: 525
```

Out[45]:

	Neighborhood	ATM	Accessories Store	Adult Boutique	Afghan Restaurant	African Restaurant	Airport Lounge	Airport Service	American Restaurant	Animal Shelter	Antique Shop	Aquarium	Arcade
0	Wakefield (Bronx)	0	0	0	0	0	0	0	0	0	0	0	0
1	Wakefield (Bronx)	0	0	0	0	0	0	0	0	0	0	0	0
2	Wakefield (Bronx)	0	0	0	0	0	0	0	0	0	0	0	0
3	Wakefield (Bronx)	0	0	0	0	0	0	0	0	0	0	0	0
4	Wakefield (Bronx)	0	0	0	0	0	0	0	0	0	0	0	0

In [46]:
```
#the venue dataframe is now expanded to include the different categories
nyc_london_onehot.shape
```

Out[46]:
```
(29507, 525)
```

In [47]:
```
#group rows by neighborhood and by taking the mean of the frequency of occurrence of each category
nyc_london_grouped = nyc_london_onehot.groupby('Neighborhood').mean().reset_index()
nyc_london_grouped.head()
```

Out[47]:

	Neighborhood	ATM	Accessories Store	Adult Boutique	Afghan Restaurant	African Restaurant	Airport Lounge	Airport Service	American Restaurant	Animal Shelter	Antique Shop	Aquarium	Arcade
0	Abbey Wood (Bexley, Greenwich)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.0
1	Acton (Ealing, Hammersmith and Fulham)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.0
2	Aldgate (City)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.0
3	Aldwych (Westminster)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.01	0.0	0.0	0.0	0.0
4	Allerton (Bronx)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.0

In [48]:
```
nyc_london_grouped.shape
```

Out[48]:
```
(603, 525)
```

In [49]:
```
#create the new dataframe and display the top 10 venues for each neighborhood.
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
nyc_london_neighborhoods_venues_sorted = pd.DataFrame(columns=columns)
nyc_london_neighborhoods_venues_sorted['Neighborhood'] = nyc_london_grouped['Neighborhood']

for ind in np.arange(nyc_london_grouped.shape[0]):
    nyc_london_neighborhoods_venues_sorted.iloc[ind, 1:] = return_most_common_venues(nyc_london_grouped.

nyc_london_neighborhoods_venues_sorted.head()
```

Out[49]:

	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	9th Most Common Venue	10th Most Common Venue
0	Abbey Wood (Bexley, Greenwich)	Playground	Grocery Store	Campground	Indian Restaurant	Fabric Shop	Factory	Falafel Restaurant	Farm	Financial or Legal Service	Farmers Market
1	Acton (Ealing, Hammersmith and Fulham)	Gym / Fitness Center	Pub	Grocery Store	Indian Restaurant	Train Station	Park	Fast Food Restaurant	Supermarket	Chinese Restaurant	Bakery
2	Aldgate (City)	Coffee Shop	Hotel	Gym / Fitness Center	Restaurant	Middle Eastern Restaurant	Cocktail Bar	Café	Italian Restaurant	Food Truck	French Restaurant
3	Aldwych (Westminster)	Hotel	Theater	Coffee Shop	Restaurant	Café	Ice Cream Shop	Bakery	Steakhouse	Museum	History Museum
4	Allerton (Bronx)	Donut Shop	Pizza Place	Sandwich Place	Supermarket	Food	Fast Food Restaurant	Bus Station	Pharmacy	Discount Store	Gas Station

In [51]:
```
nyc_london_neighborhoods_venues_sorted.shape
```

Out[51]:
```
(603, 11)
```

Use K-Means to Cluster Neighborhoods Across NYC and London

In [50]:
```
# set number of clusters
kclusters = 5

nyc_london_grouped_clustering = nyc_london_grouped.drop('Neighborhood', 1)

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

# check cluster labels generated for each row in the dataframe
kmeans.labels_[0:10]
```

Out[50]:
```
array([4, 1, 3, 3, 0, 4, 4, 0, 1, 0], dtype=int32)
```

In [52]:
```
#quick visual on how the clusters are distributed
print(kmeans.labels_)
```

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

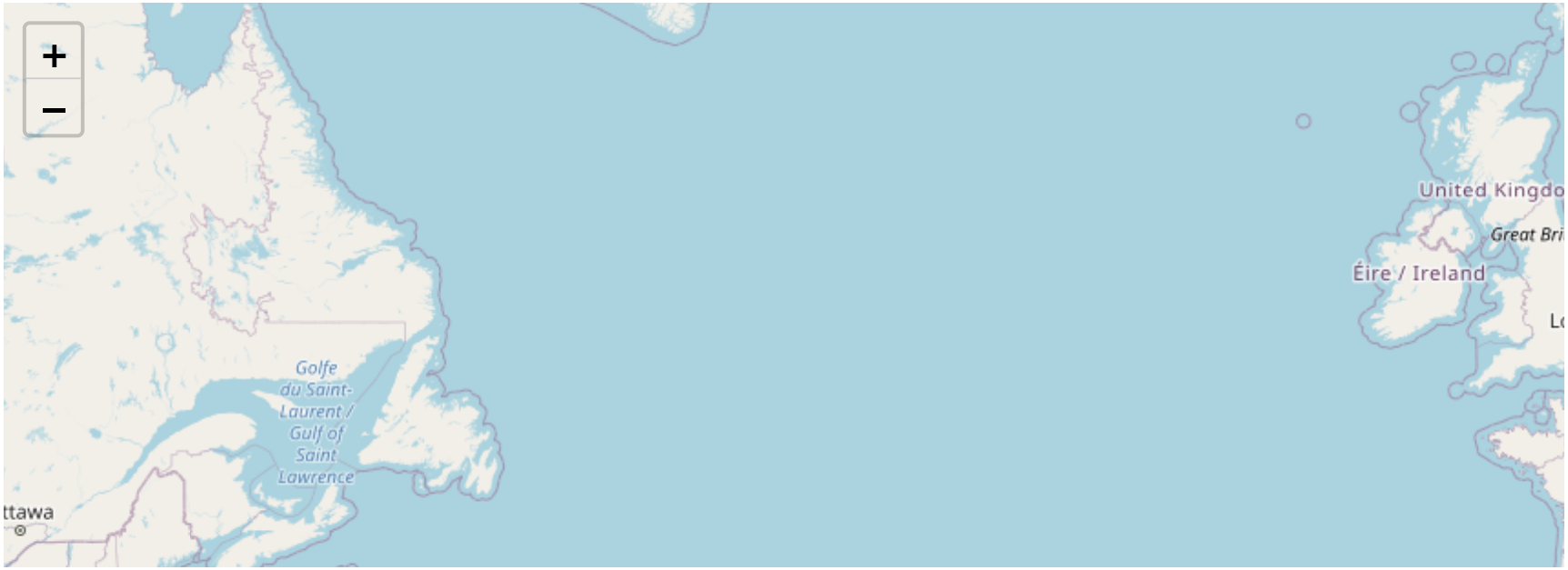
In [53]:
```
# add clustering labels
nyc_london_neighborhoods_venues_sorted.insert(0, 'Cluster Labels', kmeans.labels_)

merged = nyc_london_neighborhoods

merged = merged.join(nyc_london_neighborhoods_venues_sorted.set_index('Neighborhood'), on='Neighborhood'

merged.head() # check the last columns!
```

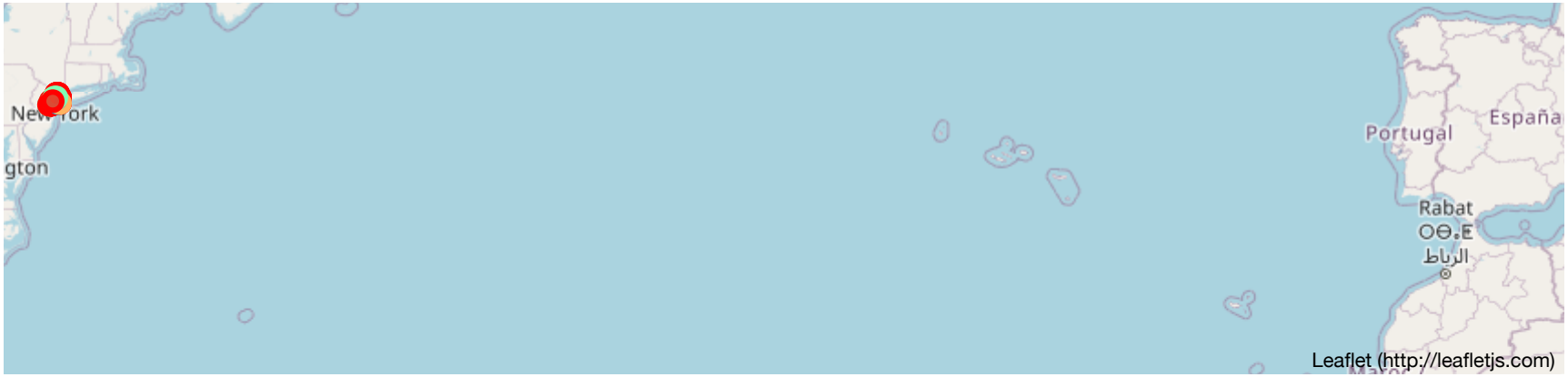
Out[53]:

	City	Neighborhood	Latitude	Longitude	Cluster Labels	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
0	New York City	Wakefield (Bronx)	40.894705	-73.847201	0	Pharmacy	Caribbean Restaurant	Supermarket	Fast Food Restaurant	Gas Station	Bagel Shop	Ice Cream Shop
1	New York City	Co-op City (Bronx)	40.874294	-73.829939	0	Mattress Store	Accessories Store	Pizza Place	Fast Food Restaurant	Shopping Mall	Pharmacy	Bakery
2	New York City	Eastchester (Bronx)	40.887556	-73.827806	0	Caribbean Restaurant	Fast Food Restaurant	Diner	Burger Joint	Shopping Mall	Grocery Store	Cocktail Bar
3	New York City	Fieldston (Bronx)	40.895437	-73.905643	4	Bus Station	Park	Plaza	Coffee Shop	River	Playground	Café
4	New York City	Riverdale (Bronx)	40.890834	-73.912585	0	Bank	Sandwich Place	Bar	Medical Supply Store	Mexican Restaurant	Pharmacy	Pizza Place

In [54]:
```
# create map
map_clusters = folium.Map(location=[avg_latitude, avg_longitude], zoom_start=4)

# 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(merged['Latitude'], merged['Longitude'], merged['Neighborhood'], merge
    label = folium.Popup(str(poi) + ' 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[54]:





Examine Clusters

In [46]:
```
merged.loc[merged['Cluster Labels'] == 0, merged.columns[[1] + list(range(5, merged.shape[1]))]]
```

133	Howard Beach (Queens)	Pharmacy	Italian Restaurant	Park	Fast Food Restaurant	Clothing Store	Chinese Restaurant	Sushi Restaurant	Bank	Café
134	Corona (Queens)	Science Museum	Playground	Mexican Restaurant	Ice Cream Shop	Donut Shop	Convenience Store	Park	Food Truck	Deli / Bodega
136	Kew Gardens (Queens)	Chinese Restaurant	Deli / Bodega	Pizza Place	Cosmetics Shop	Supermarket	Lounge	Bar	Bank	Donut Shop
137	Richmond Hill (Queens)	Pizza Place	Indian Restaurant	Deli / Bodega	Bank	Lounge	Diner	Latin American Restaurant	Costume Shop	Spanish Restaurant
138	Flushing (Queens)	Bubble Tea Shop	Chinese Restaurant	Hotpot Restaurant	Bakery	Mobile Phone Shop	Korean Restaurant	Ice Cream Shop	Dumpling Restaurant	Sandwich Place
140	Sunnyside (Queens)	Pizza Place	Bakery	Diner	Bar	Sandwich Place	Peruvian Restaurant	Coffee Shop	Turkish Restaurant	Donut Shop
141	East Elmhurst (Queens)	Rental Car Location	Deli / Bodega	Ice Cream Shop	Donut Shop	American Restaurant	Hotel Bar	Fried Chicken Joint	Fast Food Restaurant	Gas Station
142	Maspeth (Queens)	Deli / Bodega	Pizza Place	Diner	Chinese Restaurant	Park	Grocery Store	Mobile Phone Shop	Bank	Sandwich Place

In [52]:
```
merged.loc[merged['Cluster Labels'] == 2, merged.columns[[1] + list(range(5, merged.shape[1]))]]
```

Out[52]:

	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	9th Most Common Venue	10th Most Common Venue
85	Sea Gate (Brooklyn)	Beach	Supermarket	Paper / Office Supplies Store	Park	Construction & Landscaping	Spa	Falafel Restaurant	Farm	Factory	Financial or Legal Service
178	Rockaway Beach (Queens)	Beach	Ice Cream Shop	Latin American Restaurant	Bar	Bagel Shop	Food Stand	BBQ Joint	Restaurant	Pharmacy	Hotel
179	Neponsit (Queens)	Beach	Park	Bus Stop	Fast Food Restaurant	Financial or Legal Service	Film Studio	Filipino Restaurant	Field	Zoo Exhibit	Fish Market
190	Belle Harbor (Queens)	Beach	Spa	Boutique	Italian Restaurant	Deli / Bodega	Pub	Event Space	Chinese Restaurant	Bakery	Bagel Shop
191	Rockaway Park (Queens)	Beach	Italian Restaurant	Spa	Bagel Shop	Pizza Place	Donut Shop	Deli / Bodega	Bar	Pub	Boutique
204	South Beach (Staten Island)	Beach	Pier	Athletics & Sports	Theme Park	Skate Park	Deli / Bodega	Soccer Field	American Restaurant	BBQ Joint	Food
232	Midland Beach (Staten Island)	Baseball Field	Beach	Other Great Outdoors	Basketball Court	Food	Bookstore	Chinese Restaurant	Bus Stop	Bagel Shop	Deli / Bodega
302	Hammels (Queens)	Beach	Taco Place	Supermarket	Wine Shop	Fried Chicken Joint	Gym / Fitness Center	Farmers Market	Fast Food Restaurant	Bakery	Pharmacy

In [53]:
```
merged.loc[merged['Cluster Labels'] == 3, merged.columns[[1] + list(range(5, merged.shape[1]))]]
```

		Venue	Venue	Venue	Venue	Venue	Venue	Venue	Venue	Venue
521	Bush (Hammersmith and Fulham)	Coffee Shop	Clothing Store	Grocery Store	Bakery	Hotel	Pizza Place	Supermarket	Thai Restaurant	Sandwich Place
523	Shoreditch (Hackney)	Coffee Shop	Italian Restaurant	Gym / Fitness Center	Café	Restaurant	Food Truck	Hotel	Indian Restaurant	Bagel Shop
525	Snaresbrook (Redbridge, Waltham Forest)	Lake	Lounge	English Restaurant	Hotel	Gym Pool	Bus Stop	Zoo Exhibit	Film Studio	Filipino Restaurant
526	Soho (Westminster)	Coffee Shop	Tapas Restaurant	Restaurant	Pizza Place	Liquor Store	Clothing Store	Cocktail Bar	Indian Restaurant	Bookstore
527	Somerstown (Camden)	Coffee Shop	Café	Hotel	Hotel Bar	Italian Restaurant	Bar	Bakery	Indian Restaurant	Bookstore
529	South Kensington (Kensington and Chelsea)	Hotel	Italian Restaurant	Bakery	Burger Joint	French Restaurant	Science Museum	Sandwich Place	Japanese Restaurant	Exhibit
	South									

In [55]:
```
merged.loc[merged['Cluster Labels'] == 4, merged.columns[[1] + list(range(5, merged.shape[1]))]]
```

	Lewisham)	Venue	Venue	Venue	Venue	Venue	Venue	Venue	Venue	Venu
454	Lea Bridge (Hackney)	Park	Nature Preserve	Bus Stop	Tennis Court	Skating Rink	Café	Gym / Fitness Center	Farm	Intersectic
458	Leyton (Waltham Forest)	Grocery Store	Park	Café	Fried Chicken Joint	Pub	Mediterranean Restaurant	Chinese Restaurant	Gym / Fitness Center	Pharma
467	Manor Park (Newham)	Restaurant	Train Station	Park	Gym / Fitness Center	Indian Restaurant	Hotel	Field	Fast Food Restaurant	Farme Mark
472	Merton Park (Merton)	Park	Grocery Store	Pub	Cricket Ground	Indian Restaurant	Train Station	Pizza Place	Coffee Shop	Din
479	Mottingham (Bromley)	Gym / Fitness Center	Gym	Motorcycle Shop	Park	Zoo Exhibit	Fish & Chips Shop	Fabric Shop	Factory	Falaf Restaura
484	New Southgate (Barnet)	Pizza Place	Convenience Store	Metro Station	Park	Grocery Store	Beer Bar	Bus Stop	Scenic Lookout	Chines Restaura
487	Norbury (Croydon)	Park	River	Forest	Mediterranean Restaurant	Hotel	Convenience Store	Grocery Store	Gym	Farme Mark

In [ ]: