Lab 5: Geospatial Clustering with Flickr Photo Locations

Overview

In this lab, we will work with a dataset of geotagged Flickr photos in New York City (NYC). Flickr is an online photo sharing platform, and geotagged Flickr photos are photos shared on Flickr which have latitudes and longitudes attached. Many of our today's photos are automatically associated with locations thanks to the GPS receivers embedded in smartphones. When people take photos using smartphones, these photos are often geotagged. In this lab, our goal is to extract the areas of interest (AOI) in NYC where many Flickr users took photos. We can convert this goal into a geospatial clustering problem because fundamentally we are trying to identify clusters based on the locations of the photos. This lab is based on the publication below, and you can find more details about this study in the paper.

Hu, Y., Gao, S., Janowicz, K., Yu, B., Li, W. and Prasad, S., 2015. Extracting and understanding urban areas of interest using geotagged photos. Computers, Environment and Urban Systems, 54, 240-254.

Visualizing geographic data

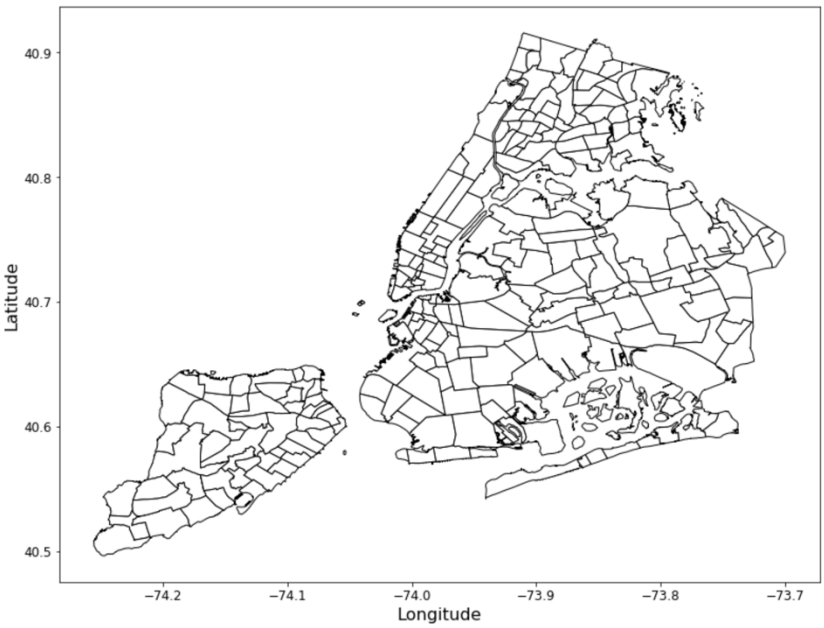
For the first step, we will use GeoPandas to load a NYC shapefile. You should be able to download the shapefile.

```
import geopandas as gpd

NYC_shp = gpd.read_file("Data/Neighborhood.shp")
print (NYC_shp)
```

Then, we will draw this shapefile using GeoPandas as a map:

```
ax = NYC_shp.plot(color='white', edgecolor='black', figsize=(15, 10))
ax.set_xlabel("Longitude", fontsize=16)
ax.set_ylabel("Latitude", fontsize=16)
```

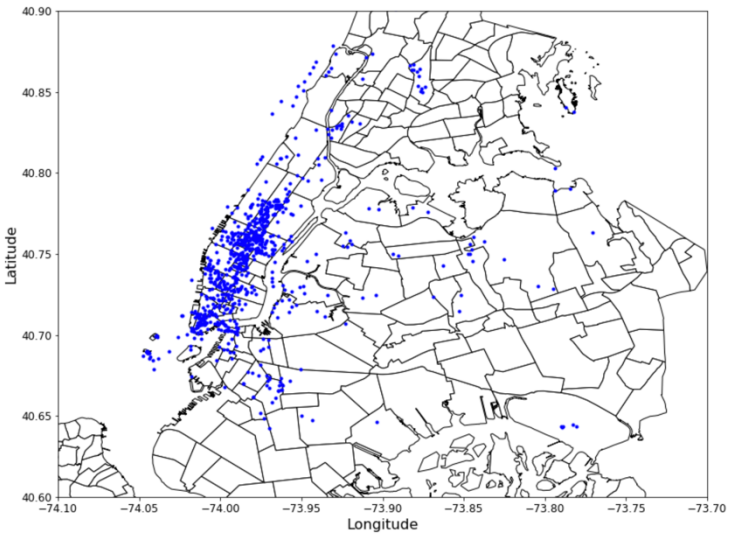
You should be able to see a figure like below:



This is only a background figure. Let's plot the Flickr photo locations on this map. The locations of Flickr data are in the file "Flickr_NewYork.csv" under the "Data" folder. In the following, we use Pandas to load the csv file, and then plot the Flickr photo locations on the map.

```
import pandas as pd
import matplotlib.pyplot as plt

flickr_data = pd.read_csv("Data/Flickr_NewYork.csv", header=None)
flickr_lat = flickr_data[2]
flickr_lng = flickr_data[3]

def plot_NYC_map():
    ax = NYC_shp.plot(color='white', edgecolor='black', figsize=(15, 10))
    ax.set_xlabel("Longitude", fontsize=16)
    ax.set_ylabel("Latitude", fontsize=16)
    plt.axis([-74.1,-73.7,40.6,40.9])

plot_NYC_map()
plt.plot(flickr_lng, flickr_lat, "b.")
```

The plotted figure should look like below:

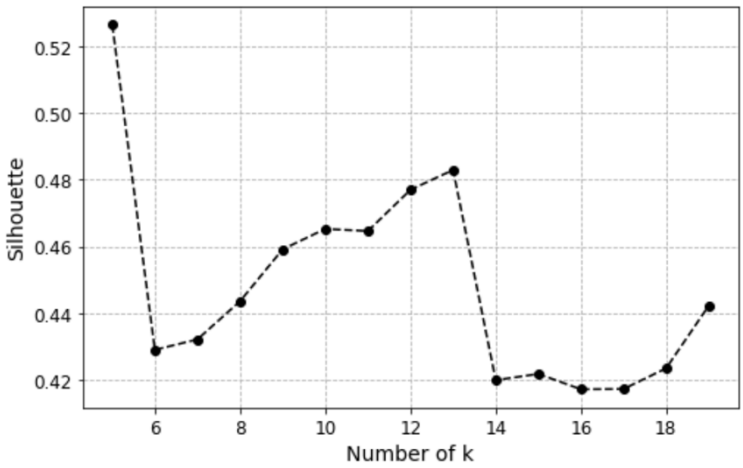


Clustering geotagged photos with K-means and DBSCAN

In our first attempt, we will try to cluster the locations of Flickr photos using K-means. Since we don't know how many clusters there are in NYC, we will first use silhouette to determine the k in a data-driven manner:

```
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
import numpy as np

np.random.seed(42)

flickr_loc = np.c_[flickr_lng,flickr_lat]
k_range = range(5,20)
silu_score = []
for k in k_range:
    kmeans_model = KMeans(n_clusters=k, n_init=10, random_state=42)
    cluster_label = kmeans_model.fit_predict(flickr_loc)
    silu_score.append(silhouette_score(flickr_loc,cluster_label))

plt.figure(figsize=(8,5))
plt.plot(k_range, silu_score, "ko--")
plt.xlabel("Number of k")
plt.ylabel("Silhouette")
plt.grid(linestyle='--')
```

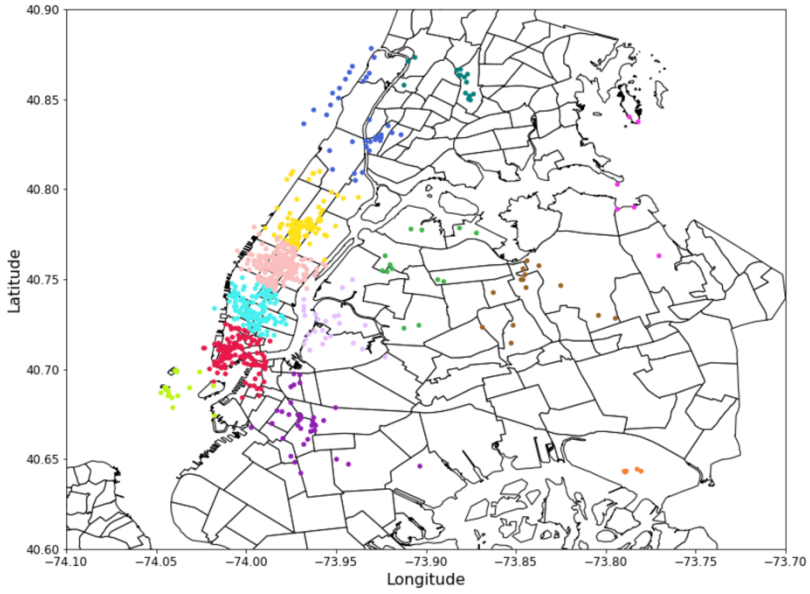
You should get a figure like below:



The result suggest maybe k=13 is a good value, so let's set k=13:

```
kmeans_model = KMeans(n_clusters=13, n_init=10, random_state=42)
cluster_label = kmeans_model.fit_predict(flickr_loc)
color_array = ['#e6194b', '#3cb44b', '#ffe119', '#4363d8', '#f58231', '#911eb4', '#46f0f0',
'#f032e6', '#bcf60c', '#fabebe', '#008080', '#e6beff', '#9a6324', '#fffac8', '#800000',
'#aaffc3', '#808000', '#ffd8b1', '#000075']


plot_NYC_map()
for cluster_index in range(13):
    plt.scatter(flickr_loc[:,0][cluster_label==cluster_index],
flickr_loc[:,1][cluster_label==cluster_index], c = color_array[cluster_index], s=15)
```

You should get a result looks like below:

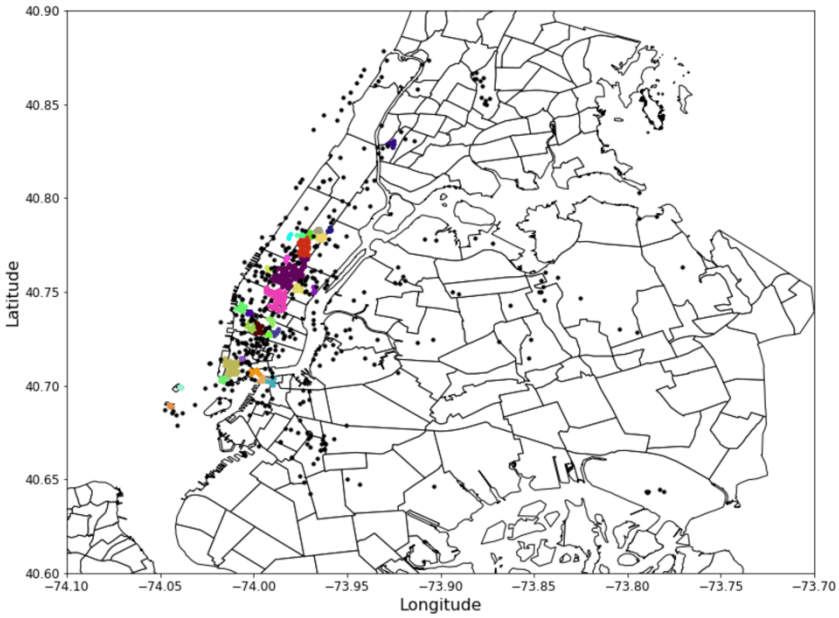


This result looks OK. However, if you recall our initial goal of finding areas of interest in NYC which attract a lot of Flickr users, this result does not seem to achieve our goal. K-means basically divides Manhattan into lower, middle, and upper parts and also include a lot of noise points in other areas. Although we have used a data-driven approach to automatically select the "best" parameters, the result does not seem to fit what we need.

Now, assume that a domain scientist tells you that such areas of interest can be defined generally based on the fact that at least 5 people have taken photos within a radius of 200 meteres (about 0.002 degrees). We can then make use of this information by using DBSCAN and setting its parameters to 0.002 for Eps and 5 for MinPts. We can run the following codes:

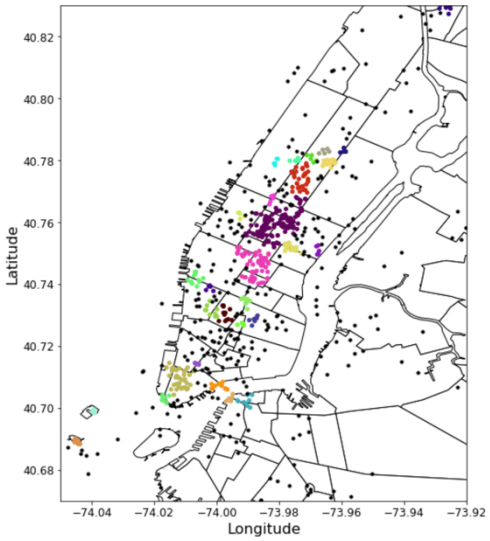
```
from sklearn.cluster import DBSCAN

dbscan = DBSCAN(eps=0.002, min_samples=5)
dbscan_label = dbscan.fit_predict(flickr_loc)
cluster_count = len(set(dbscan_label)) - 1


np.random.seed(42)
color_array_rand = []
for i in range(cluster_count):
    color_array_rand.append('#'+'%06X' % np.random.randint(0, 0xFFFFFF))


plot_NYC_map()
for cluster_index in range(cluster_count):
    plt.scatter(flickr_loc[:,0][dbscan_label==cluster_index],
flickr_loc[:,1][dbscan_label==cluster_index],  c  =  color_array_rand[cluster_index],
s=15)

# plot noise points
plt.scatter(flickr_loc[:,0][dbscan_label==-1],  flickr_loc[:,1][dbscan_label==-1],  c  =
"black", s=10)
```

You should see a figure like below:



We can further zoom to Manhatton using the following code:

```
plot_NYC_map()
for cluster_index in range(cluster_count):
    plt.scatter(flickr_loc[:,0][dbscan_label==cluster_index],
flickr_loc[:,1][dbscan_label==cluster_index],  c  =  color_array_rand[cluster_index],
s=15)

# plot noise points
plt.scatter(flickr_loc[:,0][dbscan_label==-1],  flickr_loc[:,1][dbscan_label==-1],  c  =
"black", s=10)

plt.axis([-74.05,-73.92,40.67,40.83])
```



The result now seems to make more sense with sub regions such as Time Square and Wall Street identified. Please note that from a pure clustering perspective, we cannot claim that this clustering result is better than the previous result from K-Means. However, from an application perspective, this result is based on domain knowledge and therefore it has a valid meaning for each identified cluster (namely we can guarantee that each cluster is endorsed by at least 5 people within a radius of 200 meters).

To submit:

- Please submit your Jupyter Notebook “Lab5_First_Last.ipynb”