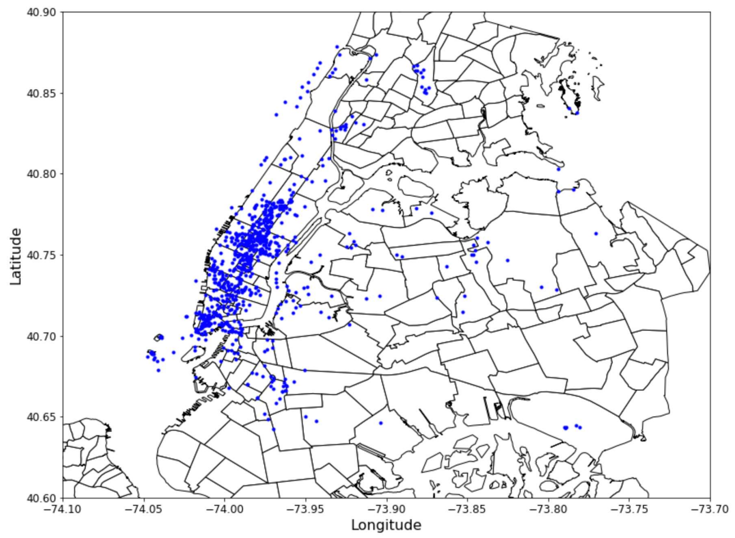
Lab 6: Geospatial Clustering with Geotagged Photo Location Data

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 learn how a geospatial machine learning paper may look like from it.

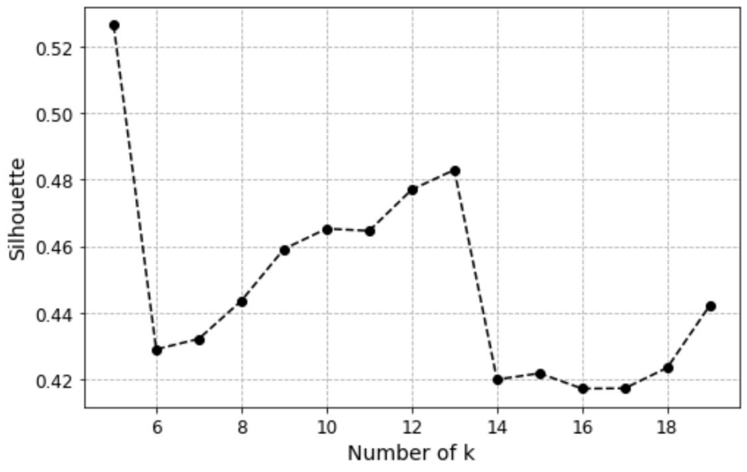
Hu, Y., Gao, S., Janowicz, K., Yu, B., Li, W. and Prasad, S., 2015. Extracting and understanding urban areas of interest using geotagged photos. Computers, Environment and Urban Systems, 54, 240-254.

Task 1 (20 pts): Explore and visualize NYC boundary and Flickr photo location data. geographic data. Use GeoPandas to load the NYC "Neighborhood.shp" shapefile. You should be able to download the shapefile. The locations of Flickr data are in the file "Flickr_NewYork.csv" under the "data" folder. Use Pandas to load the csv file, and then plot the Flickr photo locations on the NYC map. Your figure should look like below:



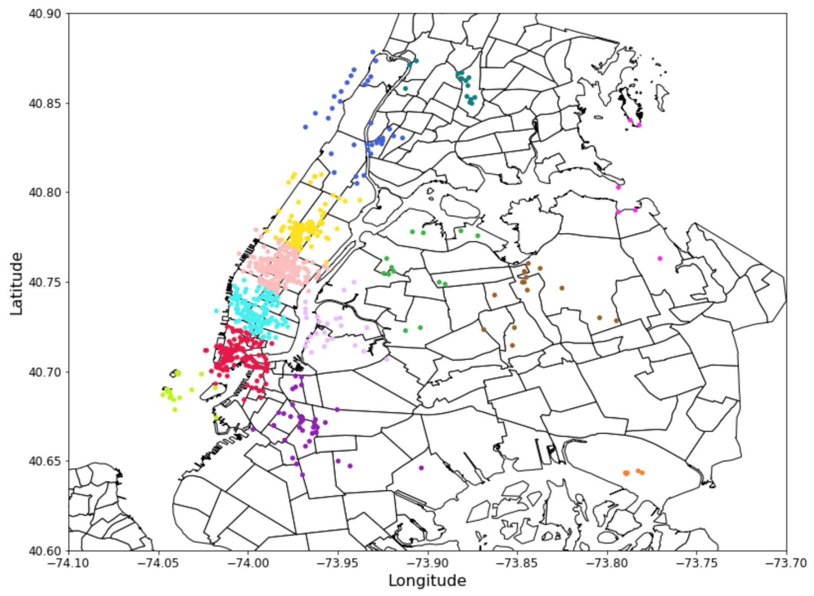
Hint: To visualize the shapefile as the background, you may want to set *color='white', edgecolor='black',* **when calling the plot function.**

Task 2 (30 pts): Use silhouette to determine the *k* for K-means clustering. 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. Test *k* from 5 to 20, and plot the silhouette figure. Your figure should look like below. Note that you must initialize your kmeans model by setting *random_state=42* **in order to get the same figure.**



Task 3 (25 pts): K-means clustering. The result suggests that maybe k=13 is a good value, so let's set k=13 for the K-means model. Create your model and use it to cluster the photo locations, and visualize the result. Note that you still need to use the same random parameters in order to ensure that the K-means model behaves as we expected: *kmeans_model = KMeans(n_clusters=13, random_state=42)*
In addition, to visualize different clusters using different colors, you will need to define a color array. You can use the color array below:
color_array = ['#e6194b', '#3cb44b', '#ffe119', '#4363d8', '#f58231', '#911eb4', '#46f0f0', '#f032e6', '#bcf60c', '#fabebe', '#008080', '#e6beff', '#9a6324', '#fffac8', '#800000', '#aaffc3', '#808000', '#ffd8b1', '#000075']

You should get a result looks like below:



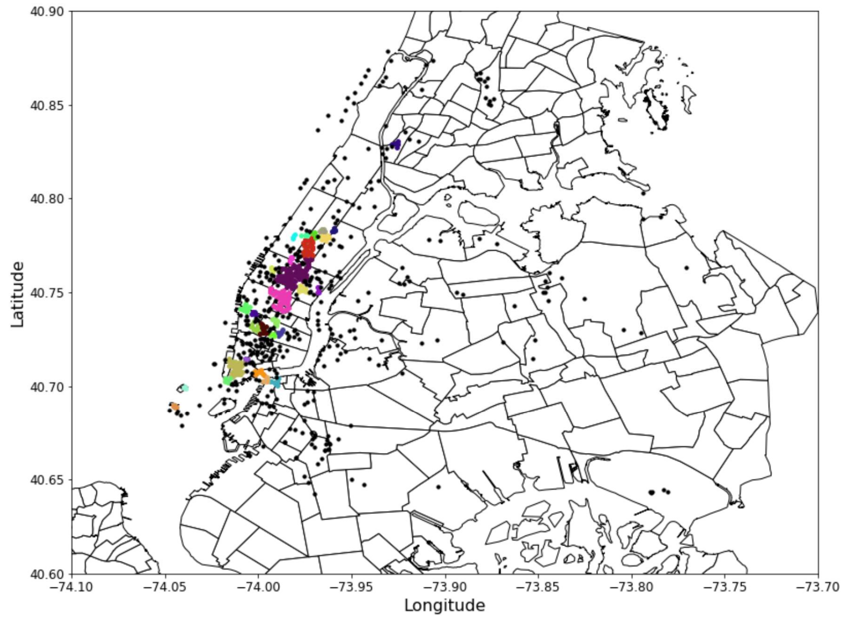
The result of K-means clustering looks OK. However, if you recall our initial goal of finding areas of interest in NYC which attract a lot of Flickr users, this result does not seem to achieve our goal. K-means basically divides Manhattan into lower, middle, and upper parts and also include a lot of noise points in other areas (because every point must be included in a cluster as required by K-means). 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 expert 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.

Task 4 (25 pts): DBSCAN clustering. Use 0.002 for Eps and 5 for MinPts, and create a DBSCAN model to cluster the location points.
Hint1: DBSCAN can identify many clusters from the point locations, and we don't know the number of clusters beforehand. In order to visualize the different clusters using different colors, you will need to define a color array based on the number of clusters identified. You can use:

```
dbscan_label = dbscan.fit_predict(flickr_loc)    # predict clusters
cluster_count = len(set(dbscan_label)) – 1     # get the number of clusters
# the following code define the color array based on the number of clusters
np.random.seed(42)
color_array_rand = []
for i in range(cluster_count):
    color_array_rand.append('#'+'%06X' % np.random.randint(0, 0xFFFFFF))
```

You should see a figure like below:

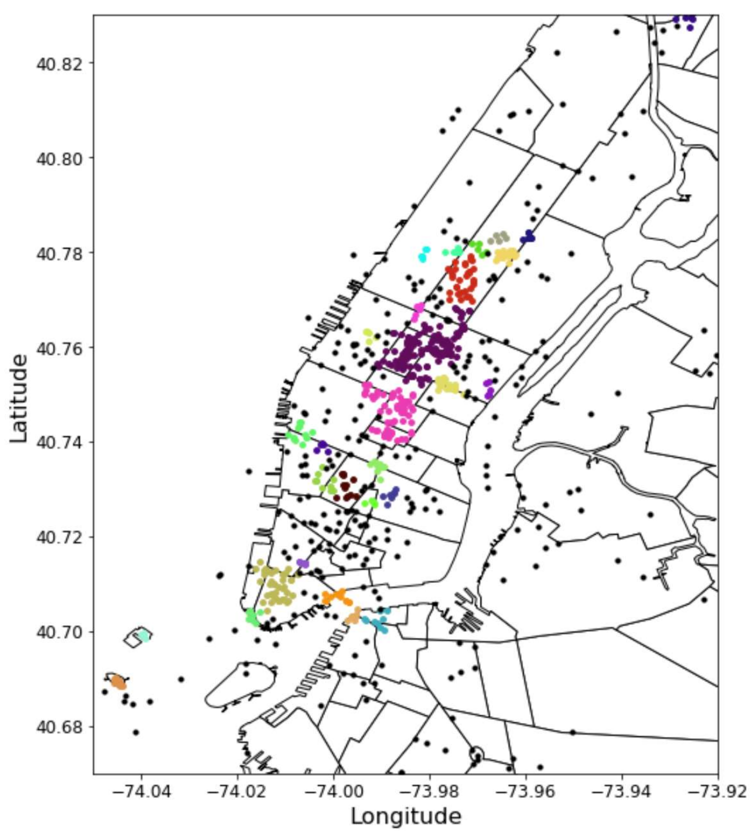


The result now seems to make more sense with sub regions such as Time Square and Wall Street identified. Please note that from a pure clustering perspective, we cannot state that this clustering result is better than the previous result from K-Means. However, from an application perspective, this result 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).

Optional step: We can further zoom to Manhatton by setting the axis to *[-74.05,-73.92,40.67,40.83]* after the map is plotted.

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

You should see something like below:



To submit:

- Please submit your Jupyter Notebook "Lab6_First_Last.ipynb"