

## **Introduction**

When people visit other cities for some reasons (traveling, attending conference, visiting a friends, etc.), many of them are eager to explore the city they are not familiar with. It would be helpful to suggest some places for them to go based on their only interests. For example, if a person is enthusiastic about museum exhibitions, it would be good to recommend her to a neighborhood that many local museums are located; if someone loves French cuisine, it is preferable to suggest a place with many such restaurants for him to choose. A person who have being living in a city for years may be able to offer useful advices regarding above scenarios, however, identifying such a person is not easy and it could be time-consuming, especially for strangers of this place. This report is aimed to solve such a problem by leveraging the ability of machine learning algorithms, punctuated by the help of various python libraries and Foursquare API, to identify places of interests for people based on their needs. To be more specific, in the jupyter notebook of this report, a person could provide a place that she favors (i.e., a place she is already familiar with, such as an art block in her city, a beach park nearby, a place with some of her most favorite restaurants, etc.), and the program will find the cluster of places that the proposed place belongs to in Toronto (the city to be visited in this report) and recommend five most similar places in the city to explore. In short, the problem to be solved in this report is:

Find the closest cluster of the place provided by the user in Toronto and recommend five most similar places in the city based on the proposed location.

## **Data**

To solve the problem described in the introduction, several sources of data will be considered.

First, it is important to find a reference point of the user-proposed place in order to calculate relevant venues around the point and then generate a vector for cluster assignment and comparison. Geocoder, a python library to convert a manually input address into coordinates, is therefore used in this report. Figure 1 shows a screenshot of how a Geocoder module could be used to find the coordinates given a sample address.

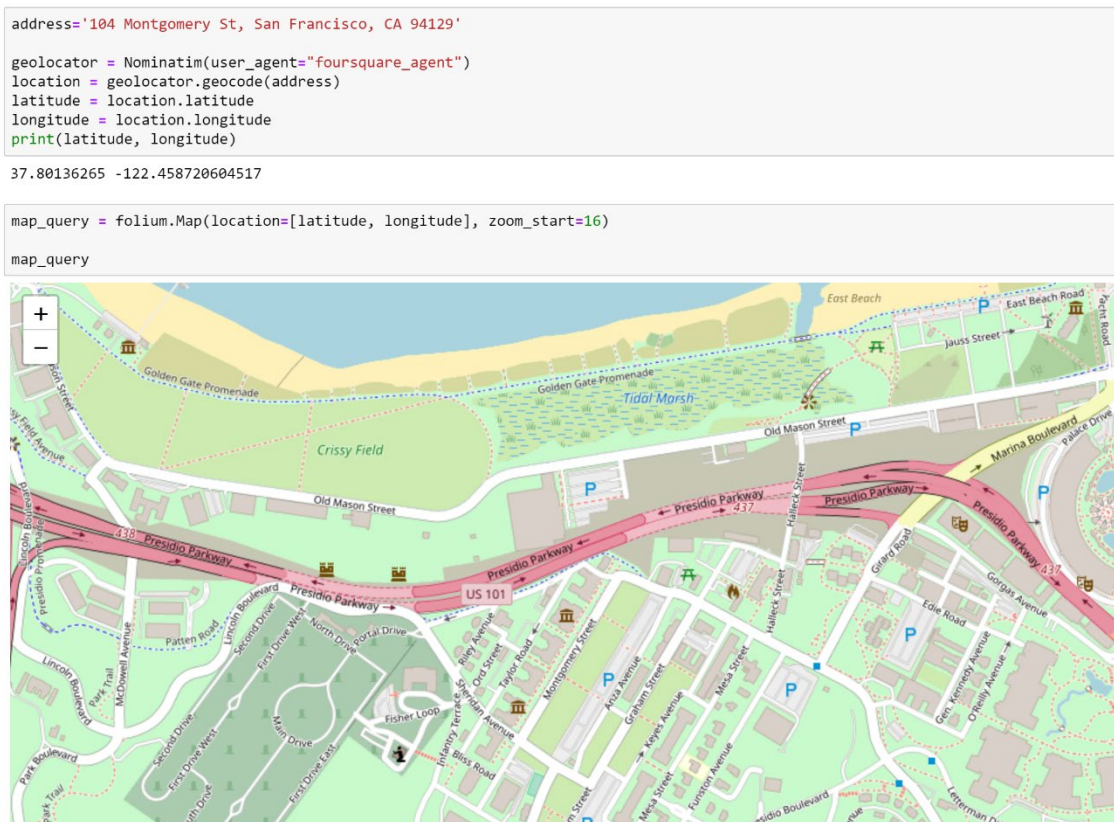
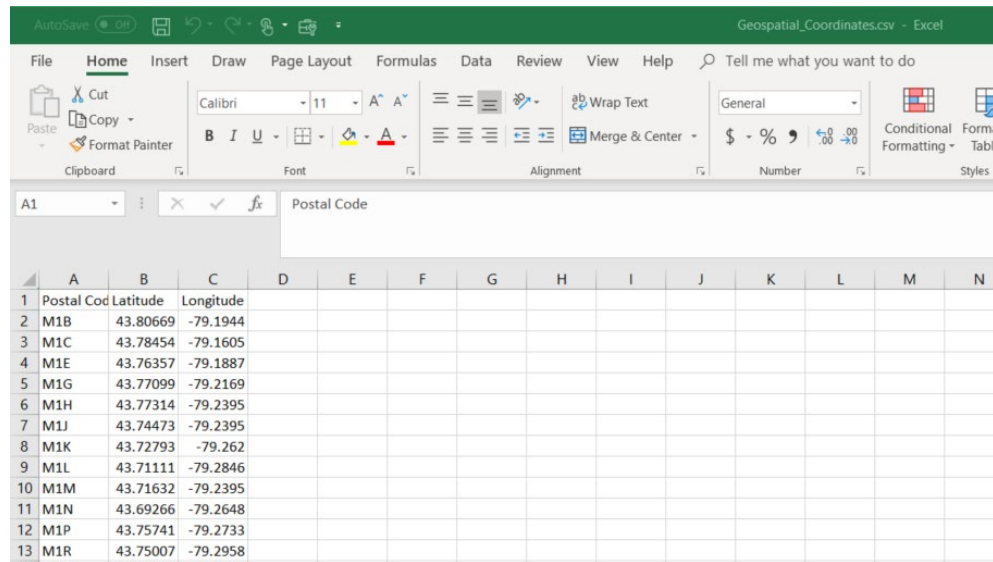


Figure 1. Code snippets of the application of Geocoder to find coordinates.

Second, since this report uses Toronto as the target city to perform the task, data regarding its neighborhoods is necessary in order to run clustering algorithm. Luckily, such kind of data is readily available from previous modules of the same course. To be specific, boroughs and neighborhoods names as well as postal codes could be found from the Wikipedia page and coordinates of each neighborhood could be provided by either the Geocoder library or the

“Geospatial Coordinates.csv” file prepared by the course instructor. Figure 2 shows a screenshot of part of the Geospatial Coordinates file which lists coordinates and their corresponding postal codes.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Postal Cod	Latitude	Longitude											
2	M1B	43.80669	-79.1944											
3	M1C	43.78454	-79.1605											
4	M1E	43.76357	-79.1887											
5	M1G	43.77099	-79.2169											
6	M1H	43.77314	-79.2395											
7	M1J	43.74473	-79.2395											
8	M1K	43.72793	-79.262											
9	M1L	43.71111	-79.2846											
10	M1M	43.71632	-79.2395											
11	M1N	43.69266	-79.2648											
12	M1P	43.75741	-79.2733											
13	M1R	43.75007	-79.2958											

Figure 2. The Geospatial Coordinates file lists postal codes of Toronto neighborhoods and their coordinates.

Third, in order to find the venues and their respective categories of the favorite place proposed by the user and neighborhoods in the target city (Toronto), Foursquare API is applied.

Foursquare is a third-party API that could be used to query specified venues around a given location. Figure 3 shows a search query using Foursquare to identify presumably French restaurants/bakeries within 1000 meters of a location in New York City by setting the query string to “French”. The result is in JSON format which could be handled to extract useful information later.

```

search_query = 'French'
radius = 1000
url = 'https://api.foursquare.com/v2/venues/search?client_id={}&client_secret={}&ll={}&v={}&query={}&radius={}&limit={}'.format(CLIENT_ID, CLIENT_SECRET, LAT, LONG, SEARCH_QUERY, RADIUS, LIMIT)
results = requests.get(url).json()
results

{
  'meta': {
    'code': 200,
    'requestId': '5c43b711dd57975fd5219e8c'
  },
  'response': {
    'venues': [
      {
        'id': '4e4e4a47bd4101d0d7a6f34b',
        'name': 'Le Croissant Shop French Bakery',
        'location': {
          'address': '10 Ave. of the Americas',
          'lat': 40.71799133506263,
          'lng': -74.00949587210722,
          'labeledLatLngs': [
            {
              'label': 'display',
              'lat': 40.71799133506263,
              'lng': -74.00949587210722
            }
          ],
          'distance': 597,
          'postalCode': '10013',
          'cc': 'US',
          'city': 'New York',
          'state': 'NY',
          'country': 'United States',
          'formattedAddress': [
            '10 Ave. of the Americas',
            'New York, NY 10013',
            'United States'
          ],
          'categories': [
            {
              'id': '4bf58dd8d48988d16a941735',
              'name': 'Bakery',
              'pluralName': 'Bakeries',
              'shortName': 'Bakery',
              'icon': {
                'prefix': 'https://ss3.4sqi.net/img/categories_v2/food/bakery_'
              }
            }
          ]
        }
      }
    ]
  }
}

```

Figure 3. A sample query using Foursquare and its returned results in JSON format.