Neighborood recommender

Business problem

When you rent a place for vacations, or for an internship, in another city that you don't know, it can be difficult to choose a neighborood. Some neighborood are quiet with parks, others have a lot of restaurants...

The aim of this recommender is to provide a list of neighborood similar to a neighborood that you like, in targeted cities.

You enter a neighborood in a city that you know and like, the name of the targeted city, and the recommender will provide to you neighborood similar to the one that you enjoy in the targeted cities. It could help people to choose a place for vacations or living when they don't have the possibility to visit the targeted city before (for example, interns, students in international exchanges...).

Data

For the scope of this projects, I will start with three cities: Paris, New York and Singapore.

For Paris, I will generate the districts numbers. For Singapore and New York, I will use data from wikipedia: Singapore:https://en.wikipedia.org/wiki/Postal_codes_in_Singapore New York: https://fr.wikipedia.org/wiki/Liste_des_quartiers_de_New_York.

The result is a list of cities and neighboroods.

```
['Singapore,Upper Thomson',
 'Singapore, Springleaf',
 'Singapore,Yishun',
 'Singapore, Sembawang',
 'Singapore,Seletar']…
```

I will then use these names to localize each neighborood using geolocator. The result is a dataframe with neighboroods localized.

	City	Latitude	Longitude	Neighborood	full_adress
275	New York	43.521737	-75.965196	Lighthouse Hill	New York,Lighthouse Hill
276	New York	40.634548	-74.112087	West New Brighton	New York,West New Brighton
277	New York	40.642326	-74.092919	New Brighton	New York,New Brighton
278	New York	40.643963	-74.073442	St. George	New York,St. George
279	New York	40.621215	-74.131809	Westerleigh	New York,Westerleigh

Geolocator provided to me some wrong locations: I have dropped them.

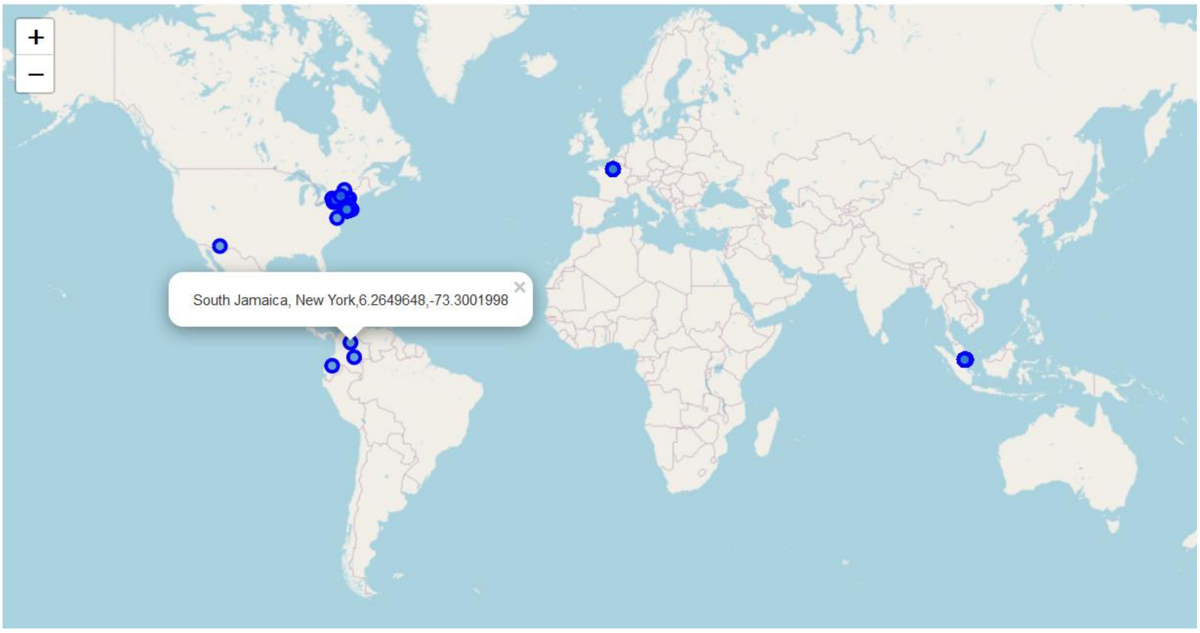


Figure 1-Wrong location provided by geolocator

In order to assign features to each neighborood, I will use the foursquare API (https://developer.foursquare.com/docs/places-api/) in order to retrieve the places and their features for each neighborood. Foursquare provides venues (restaurants, parks, yoga studios etc... within a determined radius around a point.

Foursquares provides data in json format.


```
{⊟
    "meta": {⊞ ⋯},
    "notifications": [⊞ ⋯],
    "response": {⊟
        "venues": [⊟
            {⊟
                "id": "42cc7080f964a520e9251fe3",
                "name": "Apple Store",
                "contact": {⊞ ⋯},
                "location": {⊞ ⋯},
                "canonicalUrl": "https://foursquare.com/v/apple-store/42cc7080f964a520e9251fe3",
                "categories": [⊟
                    {⊟
                        "id": "4bf58dd8d48988d122951735",
                        "name": "Electronics Store",
                        "pluralName": "Electronics Stores",
                        "shortName": "Electronics",
                        "icon": {⊟
                            "prefix": "https://foursquare.com/img/categories_v2/shops/technology_",
                            "suffix": ".png"
                        },
                        "primary": true
                    }
                ],
                "verified": true,
                "stats": {⊞ ⋯},
                "url": "http://www.apple.com/retail/sanfrancisco",
                "specials": {⊞ ⋯},
                "hereNow": {⊞ ⋯},
                "referralId": "v-1368218103"
```


Figure 2- Results of a call to the Foursquare API

We can easily retrieve the venues from this result:

```
results['response']['groups'][0]['items']
```

After grouped venues by neighborhoods, I obtained a dataframe with 264 neighborhoods, and 389 venues.

Out[29]:

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue
259	Woodhaven	Deli / Bodega	Pharmacy	Bank	Sandwich Place	Dive Bar
260	Woodlawn	Deli / Bodega	Pub	Bank	Track	Discount Store
261	Woodrow	Cosmetics Shop	Garden Center	Falafel Restaurant	Farm	Farmers Market
262	Woodside	Bar	Pub	Bakery	Grocery Store	Thai Restaurant
263	Yishun	Food Court	Chinese Restaurant	Park	Fried Chicken Joint	Hainan Restaurant

Methodology

I have worked on data retrieved from Foursquare (one-hot encoding then average by neighboroods to sort top venues by neighborood.

:

	Neighborhood	Alsatian Restaurant	Art Gallery	Art Museum	Arts & Crafts Store	Auvergne Restaurant	Bakery	Beer Store	Bistro	Bookstore	...	Scenic Lookout	Sculpture Garden	Seafood Restaurant	Snack Place	Souvenir Shop	Res
0	75013	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
1	75013	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
2	75013	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
3	75013	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
4	75013	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	

Figure 3- One-hot encoding

	Neighborhood	Alsatian Restaurant	Art Gallery	Art Museum	Arts & Crafts Store	Auvergne Restaurant	Bakery	Beer Store	Bistro	Bookstore	...	Scenic Lookout	Sculpture Garden	Seafood Restaurant	Snack Place	Souvenir Shop	Res
0	75013	0.01	0.03	0.01	0.01	0.01	0.02	0.01	0.01	0.02	...	0.01	0.01	0.01	0.01	0.01	

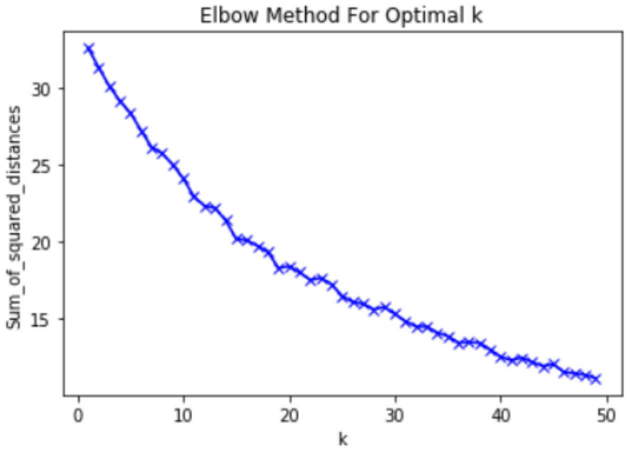
Figure 4- Mean by neighborood

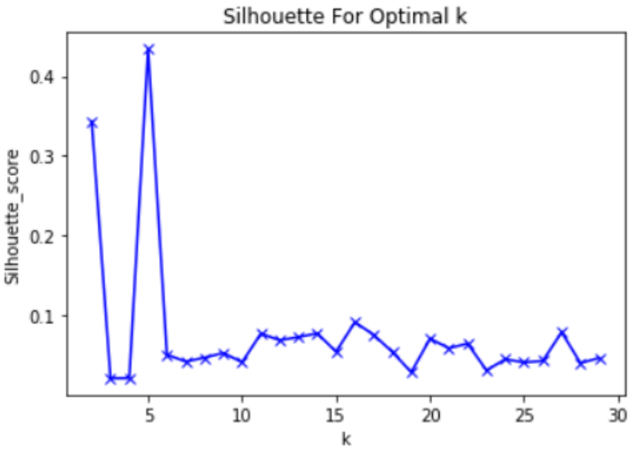
	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue	9th Most Common Venue	10th Most Common Venue
0	75013	French Restaurant	Ice Cream Shop	Plaza	Historic Site	Creperie	Art Gallery	Park	Wine Bar	Hotel	Italian Restaurant

Figure 5- Sorted features by neighborood.

I have then tried to cluster neighboroods, using a k-means and DBscan algorithms.

I first tried Kmeans:





I was not happy with the results. No real Elbow, and very unbalanced clusters if I wanted more than three clusters.

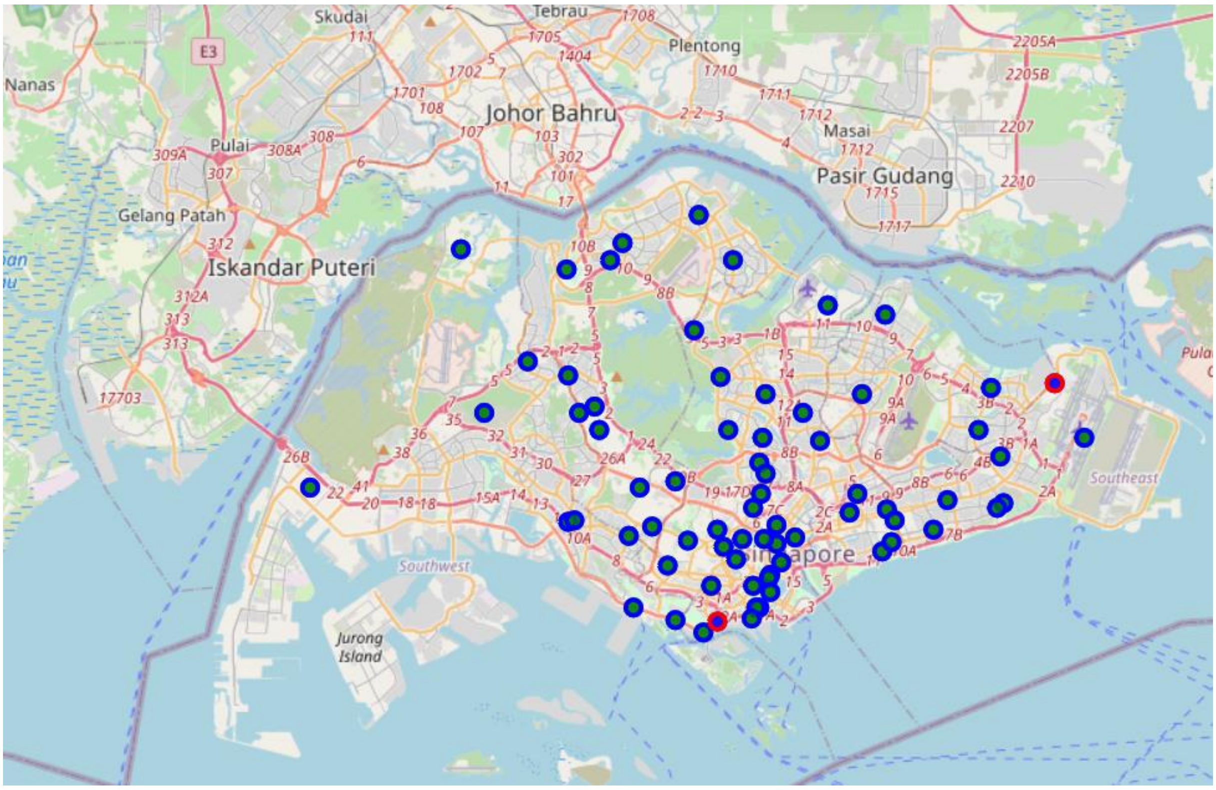


Figure 6- Very unbalanced clusters - it's a very bad result.

I tried DBscan: it returns three clusters with 82 outliers with the best epsilon. It was not usable.

I assumed that the problem was that my data were very sparsed: I have 264 rows and 389 features.

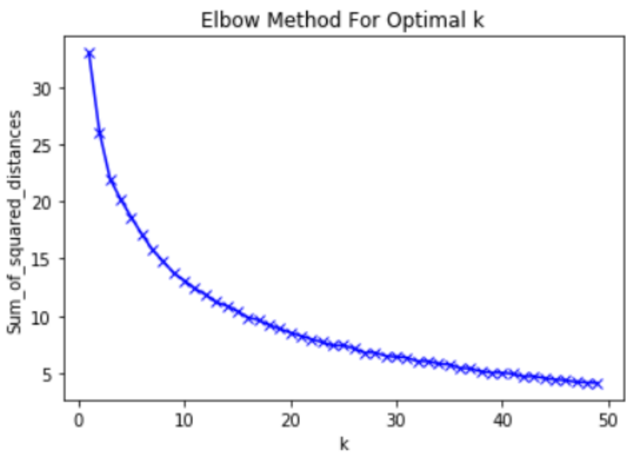
So I did some feature engineering and reduce the number of venues categories: I decrease them from 389 to 16, grouping them manually (restaurants, shops, nature etc…).

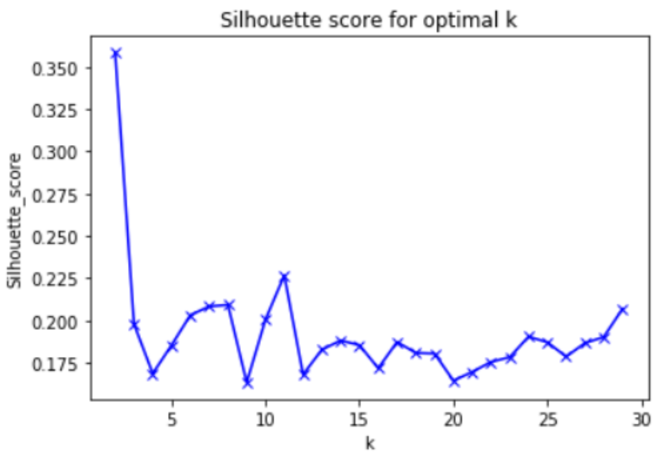
Out[55]:

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue
0	Amber Road	Restaurants	Shops	Bars	Others	Sport
1	Ang Mo Kio	Restaurants	Shops	Sport	Services	Medical
2	Braddell	Restaurants	Bars	Shops	Food Shops	Others
3	Bukit Panjang	Restaurants	Shops	Bars	Sport	Transportation
4	Bukit Timah	Sport	Nature, gardens and walks	Others	Transportation	Shops

Figure 7-Dataframe after creating meta-categories

I then ran again kmeans and dbscan, and the results where much better.





The optimum k is 9.

For DBscan, I obtained 4 clusters, and I only had 23 outliers left.

So I decided to go for kmeans (with optimized K=9 using Elbow and Silhouette method).

The result was better!

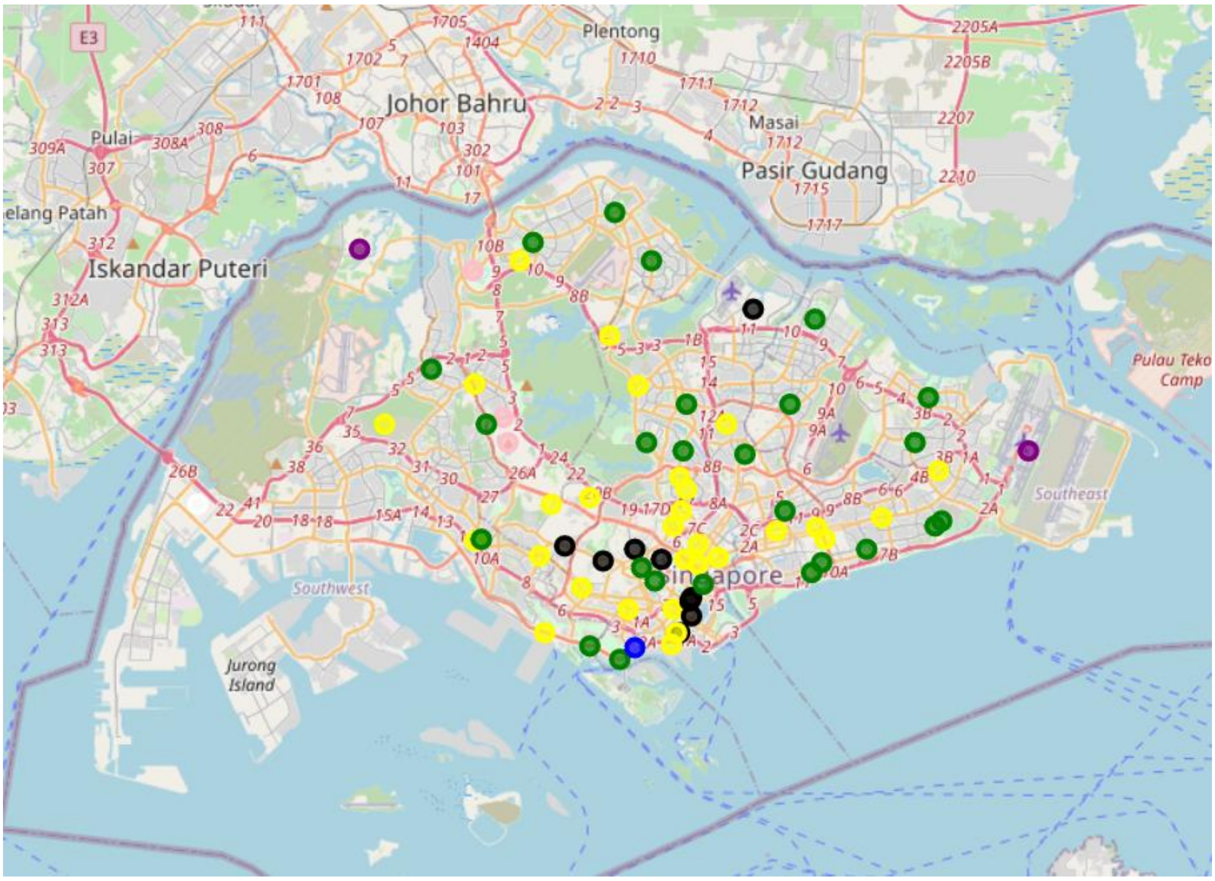


Figure 8-Map of clusters using meta categories

```
Entrée [70]: cat_merged['km Cluster Labels'].value_counts()

 Out[70]: 0    104
          3     68
          7     29
          5     25
          4     10
          2      9
          1      7
          8      6
          6      6
          Name: km Cluster Labels, dtype: int64
```

```
Entrée [71]: cat_merged['db Cluster Labels'].value_counts()

 Out[71]:  0    234
          -1     23
           1      3
           3      2
           2      2
          Name: db Cluster Labels, dtype: int64
```

Figure 9-Number of values by cluster with the two methods

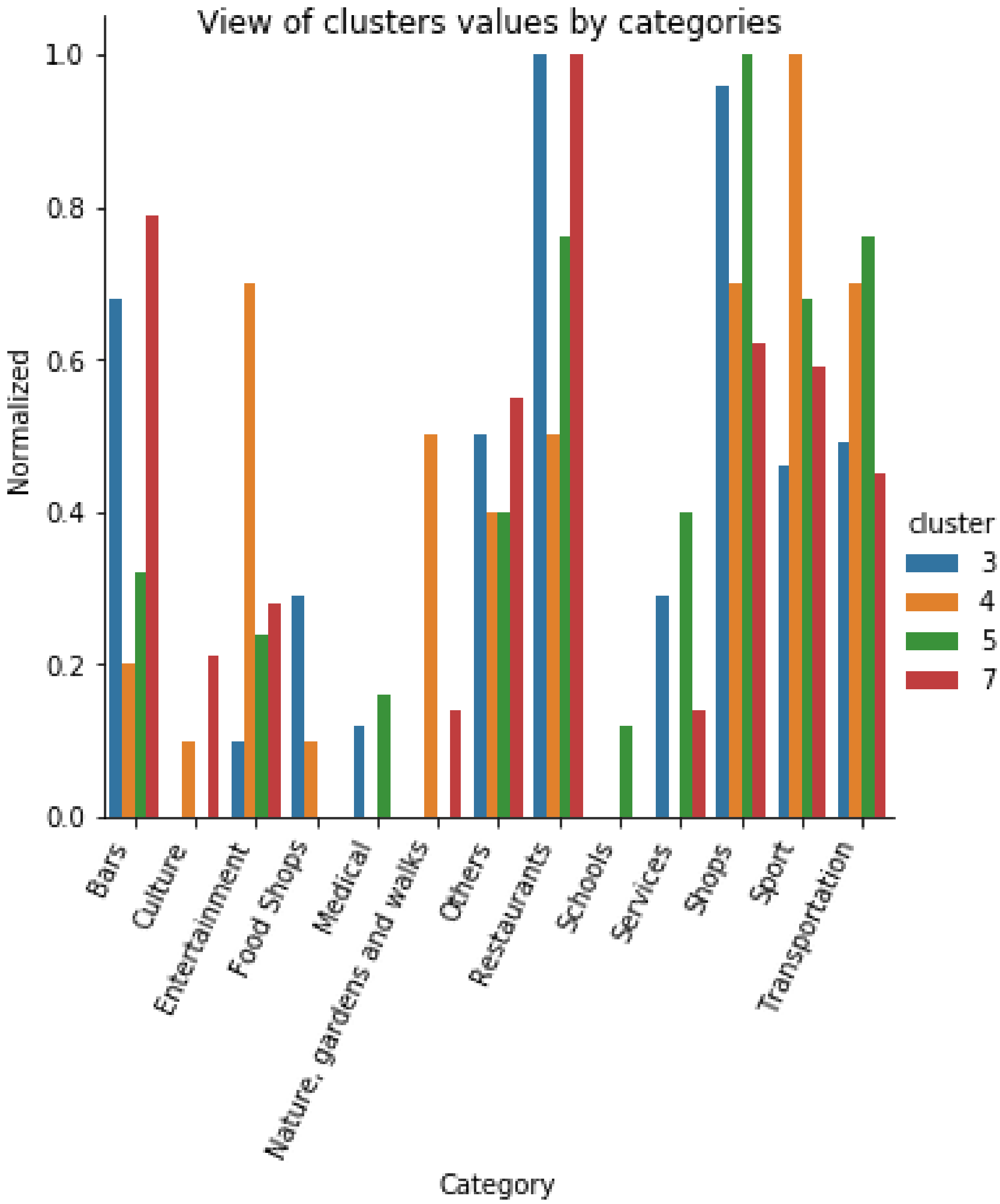


Figure 10-View of the weight of each category in the clusters with the most values with kmeans

I have decided to work with kmeans, with the meta categories. Two reasons for this choice:

- I have more clusters, meaning my recommender will be more specialized

- I don't have outliers. Outliers will be a problem with my recommender. What if the initial place is an outlier?

- Looking at the values in my cluster, dbscan has in fact one big cluster and the other ones are very small. Kmeans have more balanced clusters.

Because I lost some informations using meta categories, I have decided to offer to the user the possibility to choose one precise feature that I want to have in the chosen neighborhood.

Here are the steps:

- First, the user choose a feature he absolutly wants.

- Then he enters the neighborhood he likes

- And the targeted city

- The answer is a neighborhood in the targeted city, within the same cluster than the initial target, with the feature he absolutly wants (if exists in the cluster).

Here the result:

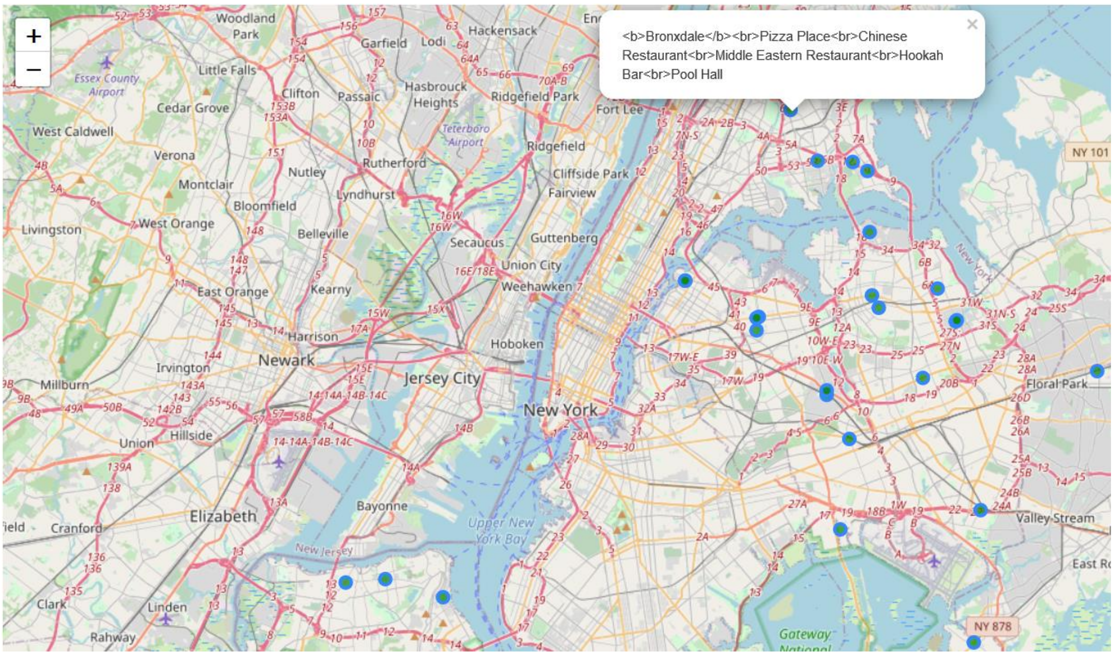
```
[148]:
#Ask informations to user
feature=input('Please choose a feature than you like.')
initial=input('Please choose a neighboorood that you like (format city, Neighboorood)')
target=input('Please choose the city where you want to move')
```

```
Please choose a feature than you like.Pizza Place
Please choose a neighboorood that you like (format city, Neighboorood)Singapore, Bukit Panjang
Please choose the city where you want to moveNew York
```

```
            Here the places similar to the neighboorood that you love, in New York and with a Pizza Place !

Out[202]:   66                    New York,Annadale
            72                     New York,Astoria
            78                     New York,Bayside
            79               New York,Bayside Hills
            81                 New York,Bedford Park
            92                   New York,Bronxdale
            93                    New York,Bruckner
            108                New York,Dongan Hills
            122                New York,Far Rockaway
            125                    New York,Flushing
            126            New York,Flushing Heights
            129                New York,Forest Hills
            139                 New York,Great Kills
            145                   New York,Hillcrest
            149                New York,Howard Beach
            153             New York,Jackson Heights
            158           New York,Kew Gardens Hills
            163                   New York,Laurelton
            183                 New York,Murray Hill
            186                    New York,New Dorp
            187               New York,New Hyde Park
            192                     New York,Norwood
            193             New York,Oakland Gardens
            196                 New York,Old Astoria
            198                   New York,Olinville
            207               New York,Port Richmond
            208                New York,Prince's Bay
            210             New York,Queensboro Hill
            215               New York,Richmond Hill
```

Out[195]:



Results

Due to data sparsity, I obtained better results with feature engineering. However, my recommender is less precise. I tried to overcome this problem adding the choice of one favorite precise feature.

Conclusion

This recommender can be improved a lot.

- For meta-categories creation: I did it manually. Use of NLP will be more efficient.
- With more cities and more data, the problem of sparsity will decrease.
- I haven't tried hierarchical clustering.