Applied Data Science Capstone Project

Neighborhood Recommendation System Using K-Means Clustering

Sathish Neelakantan

Sunday, June 29, 2019

1. Introduction to Business Problem

There are many multi-national organizations that often transfer employees from one city to another city for assignments that span longer than a year, according to its business needs. In some cases, the transfers occur between countries.

Transfer employees often reach out to their colleagues or friends, who live in the destination city, for advice regarding neighborhoods to move in to. Also, they spend significant amount of time doing online research.

To aid their search, we can build a recommendation system to help them identify similar neighborhoods to what they currently live in.

2. Background and Assumptions

For this exercise, let's assume that a multi-national organization operates in the following cities

- London, United Kingdom
- Frankfurt, Germany
- New York City, United States

This multi-national organization often transfers employees among one of the above-mentioned cities.

And transfer employees look for move-in neighborhood recommendations.

3. Methodology

To build this neighborhood recommendation system for the transfer employees, we'll do the following:

1. Get a list of neighborhoods for London, Frankfurt, and New York City.
2. Cleanse the data when necessary.
3. Using the geopy library get the geocodes (i.e., latitudes and longitudes) of each neighborhood.
4. Create a combined neighborhood dataframe from all the three cities' neighborhood data.
5. Write a custom function to use the Foursquare API to fetch the nearby venues for each of the neighborhoods in each of city.

6. Map and explore the data, to better understand the venues around each neighborhood.

7. Use k-means clustering for identifying similar neighborhoods by top nearby venues.

8. List out the neighborhoods by cluster to allow the transfer employees to pick a suitable neighborhood.

4. Data Collection and Preparation

We'll essentially use the district or neighborhood data available in the below mentioned Wikipedia pages for Frankfurt, New York City, and London.

1. https://en.wikipedia.org/wiki/Frankfurt

2. https://cocl.us/new_york_dataset

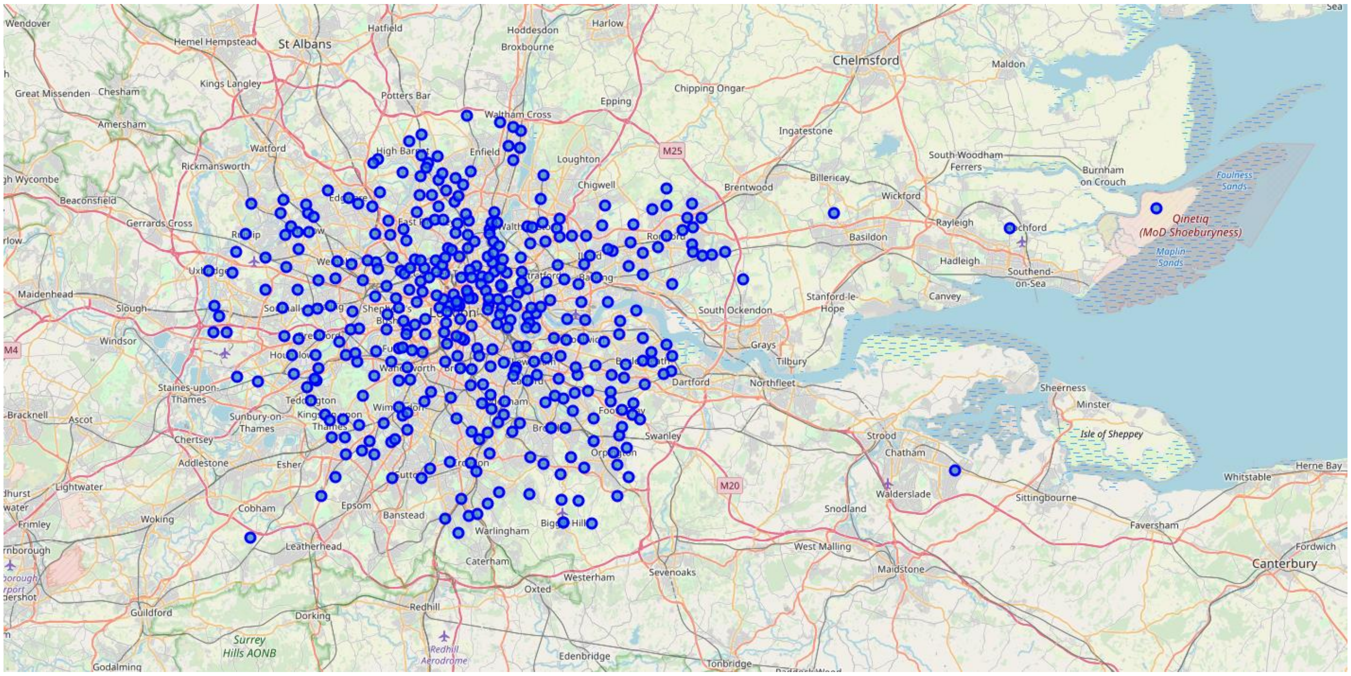
3. https://en.wikipedia.org/wiki/List_of_areas_of_London

4.1 Data Collection

Using the above-mentioned links, a list of neighborhoods for each city was obtained. Then using the Nominatim library from Open Streep Maps, the latitude and longitudes of each neighborhood was obtained. We also map these locations.

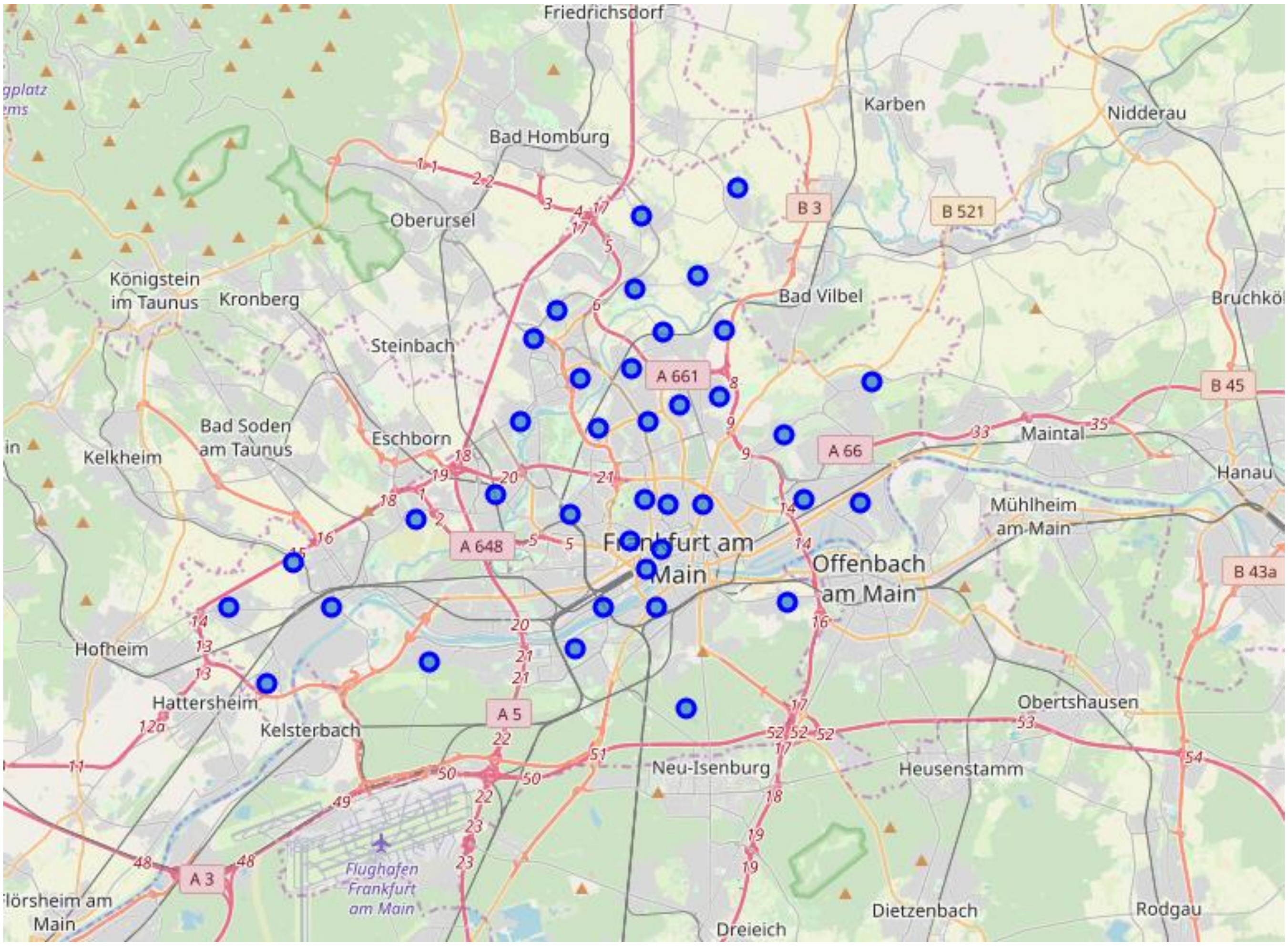
Here's a sample data for city of London, United Kingdom:

	City	Neighborhood	Latitude	Longitude
0	London	Abbey Wood	51.487621	0.114050
1	London	Acton	51.508140	-0.273261
2	London	Addington	47.725036	-66.765785
3	London	Addiscombe	51.379692	-0.074282
4	London	Albany Park	41.971937	-87.716174



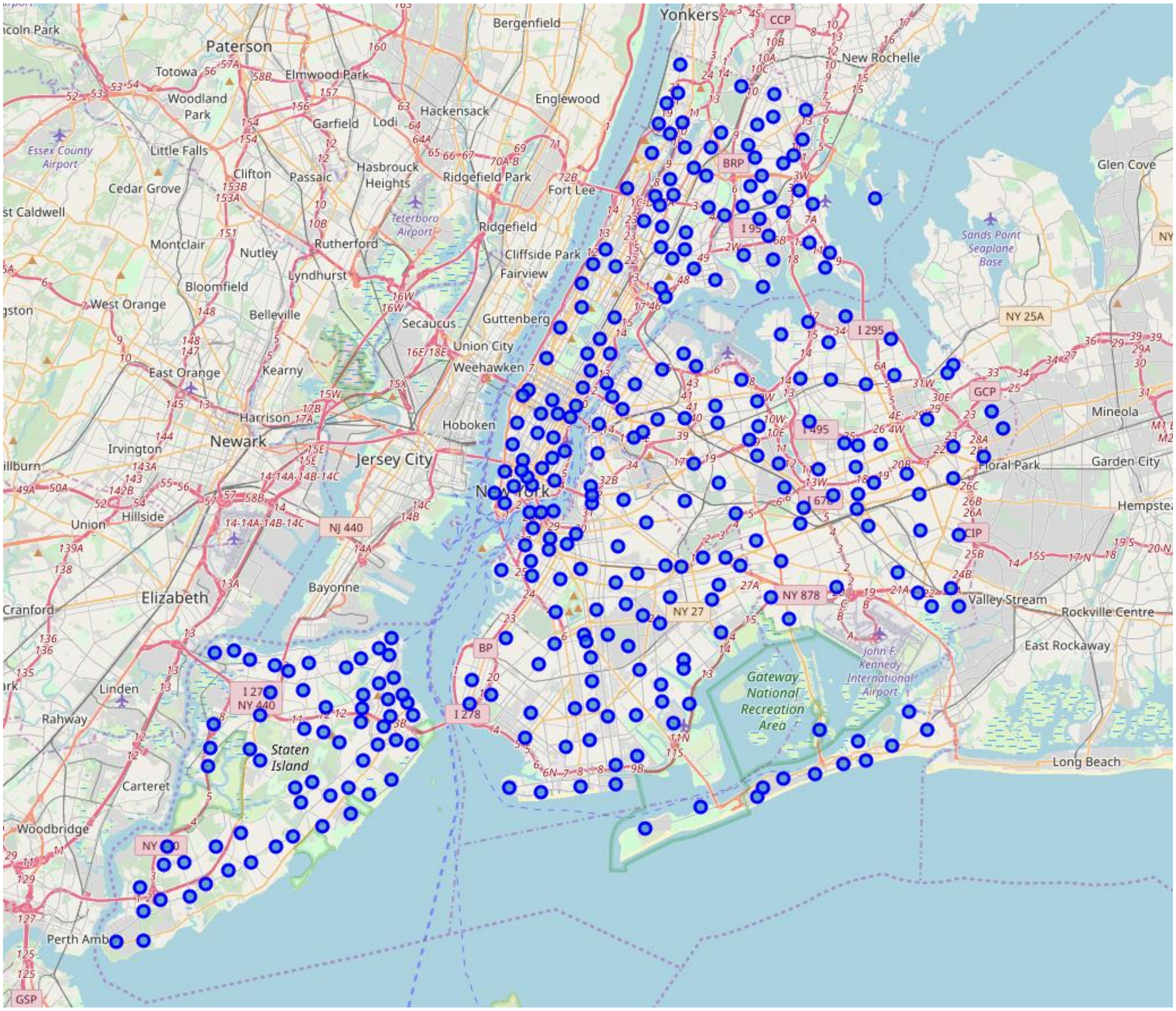
Below is some sample data for city of Frankfurt, Germany:

	City	Neighborhood	Latitude	Longitude
0	Frankfurt	Altstadt	31.504619	34.464127
1	Frankfurt	Innenstadt	50.112878	8.674922
2	Frankfurt	Bahnhofsviertel	50.107741	8.668736
3	Frankfurt	Westend-Süd	50.115245	8.662270
4	Frankfurt	Westend-Nord	50.126356	8.667921



Below is some sample data for city of New York City, United States:

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



4.2 Using Foursquare API to Get Nearby Venues

The main information about each neighborhood is obtained using the Foursquare API. Using this API, we get the top 100 nearby venues. This is accomplished by writing a custom function in Python. Below is the code:

```
def getNearbyVenues(names, latitudes, longitudes, radius=500):

    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={},{}&radius={}&limit={}'.format(
            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)
```

Explore the Venue Information

```
# Peek into the combined venues dataframe
combined_venues.sample(5)
```

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
17592	Battery Park City	40.711932	-74.016869	Vino e Grano	40.709953	-74.011574	Italian Restaurant
3837	Fitzrovia	51.518764	-0.141002	Salt Yard	51.519180	-0.136458	Tapas Restaurant
14522	Downtown	40.690844	-73.983463	Century 21 Department Store	40.690130	-73.983317	Department Store
21592	Blissville	40.737251	-73.932442	MTA - B24 Bus Stop (Van Dam)	40.735161	-73.937458	Bus Station
16740	Greenwich Village	40.726933	-73.999914	Comedy Cellar	40.730130	-74.000402	Comedy Club

For the combined list of neighborhoods from all the three cities, we get 520 unique categories.

Let's find out how many unique categories can be curated from all the returned venues

```
print('There are {} uniques categories.'.format(len(combined_venues['Venue Category'].unique())))
```

```
There are 520 uniques categories.
```

We then write a function to sort the venues in descending order. And create a 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
neighborhoods_venues_sorted = pd.DataFrame(columns=columns)
neighborhoods_venues_sorted['Neighborhood'] = combined_grouped['Neighborhood']

for ind in np.arange(combined_grouped.shape[0]):
    neighborhoods_venues_sorted.iloc[ind, 1:] = return_most_common_venues(combined_grouped.iloc[ind, :], num_top_venues)

neighborhoods_venues_sorted.head()
```

	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	Playground	Wine Shop	Grocery Store	Campground	Yoshoku Restaurant	Eye Doctor	Exhibit	Event Space	Event Service	Ethiopian Restaurant
1	Acton	Pub	Gym / Fitness Center	Hotel	Fast Food Restaurant	Grocery Store	Shopping Mall	Thai Restaurant	Chinese Restaurant	Turkish Restaurant	Athletics & Sports
2	Addiscombe	Park	Bakery	Grocery Store	Chinese Restaurant	Fast Food Restaurant	Café	Event Service	Ethiopian Restaurant	Farmers Market	Exhibit
3	Albany Park	Sandwich Place	Fried Chicken Joint	Chinese Restaurant	Diner	Café	Gas Station	Korean Restaurant	Cocktail Bar	Grocery Store	Pizza Place
4	Aldborough	Flower Shop	Construction & Landscaping	Farm	Yoshoku Restaurant	Falafel Restaurant	Factory	Eye Doctor	Exhibit	Event Space	Event Service

Clustering and Results

We then use K-means clustering to cluster the data to identify similar neighborhoods by category. This is case we randomly pick K=5. Further analysis can be done to optimize K.

10. Cluster Neighborhoods

Run *k*-means to cluster the neighborhood into 5 clusters.

```
# set number of clusters
kclusters = 5

combined_grouped_clustering = combined_grouped.drop('Neighborhood', 1)

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

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

```
array([0, 2, 4, 3, 2, 2, 2, 3, 3, 3])
```

Let's create a new dataframe that includes the cluster as well as the top 10 venues for each neighborhood.

```
# add clustering labels
neighborhoods_venues_sorted.insert(0, 'Cluster Labels', kmeans.labels_)

combined_merged = df_combined

# merge toronto_grouped with toronto_data to add latitude/longitude for each neighborhood
combined_merged = combined_merged.join(neighborhoods_venues_sorted.set_index('Neighborhood'), on='Neighborhood')

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

	City	Neighborhood	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue
0	London	Abbey Wood	51.487621	0.114050	0.0	Playground	Wine Shop	Grocery Store	Campground
1	London	Acton	51.508140	-0.273261	2.0	Pub	Gym / Fitness Center	Hotel	Fast Food Restaurant
2	London	Addington	47.725036	-66.765785	NaN	NaN	NaN	NaN	NaN
3	London	Addiscombe	51.379692	-0.074282	4.0	Park	Bakery	Grocery Store	Chinese Restaurant
4	London	Albany Park	41.971937	-87.716174	3.0	Sandwich Place	Fried Chicken Joint	Chinese Restaurant	Diner

Discussion

For example, from the data, if an employee gets transferred from Berkershiem neighborhood of Frankfurt to London and is looking for a similar neighborhood, Woodside Park neighborhood of London would be a reasonable choice, because both the neighborhoods belong to cluster 2.

11. Examine Clusters

```python
combined_merged[combined_merged['Cluster Labels']==2.0].sample(10)
```

	City	Neighborhood	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue
175	London	Fortis Green	51.590997	-0.153421	2.0	Indian Restaurant	Italian Restaurant	Mediterranean Restaurant
437	London	Strawberry Hill	51.439168	-0.339301	2.0	Convenience Store	Train Station	Thai Restaurant
668	New York	South Side	40.710861	-73.958001	2.0	Bar	Coffee Shop	American Restaurant
99	London	Clapham	51.462292	-0.138856	2.0	Pub	Café	Bar
519	London	Woodside Park	37.309857	-80.036707	2.0	Brewery	Yoshoku Restaurant	Fast Food Restaurant
555	Frankfurt	Berkersheim	50.171166	8.701191	2.0	German Restaurant	Platform	Yoshoku Restaurant
632	New York	Williamsburg	40.707144	-73.958115	2.0	Grocery Store	Bar	Bagel Shop
278	London	Leytonstone	51.571078	0.006424	2.0	Pub	Café	Coffee Shop
255	London	Kensington	51.498995	-0.199123	2.0	Café	Grocery Store	Sandwich Place
243	London	Hornsey	51.587364	-0.120967	2.0	Pub	Supermarket	Pizza Place

Conclusion

So, essentially, this dataset can be presented to the transfer employees and allow them to slice and pick the subset of data they find appropriate. I believe many employees will find this useful. This can further be use for other purposes.