The Battle of Neighborhoods

Week 1

Building a Recommendation Engine for Transfer Employees

Introduction/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.

Assumption

- For this exercise, let’s assume that a multi-national organization operates in the following cities
  - Frankfurt
  - New York
  - London
- This multi-national organization often transfers employees among one of the above mentioned cities.
- And transfer employees look for move-in neighborhood recommendations.

Input Data

- We'll essentially use the district or neighborhood data available in the below mentioned Wikipedia pages for Frankfurt, New York City, and London.
- https://en.wikipedia.org/wiki/Frankfurt
- https://cocl.us/new_york_dataset
- https://en.wikipedia.org/wiki/List_of_areas_of_London

Data Description - Frankfurt

```
# Get wikipedia dataset for Frankfurt
# https://en.wikipedia.org/wiki/Frankfurt
Frankfurt_html = wp.page("Frankfurt").html().encode("UTF-16")

# Get the table containing the 46 city districts of Frankfurt on the Wikipedia page
df_Frankfurt = pd.read_html(Frankfurt_html, header = 0)[4][['No','City district (Stadtteil)']] # Table titled: "Population of the 46 city districts on 31 December 2009"
print(df_Frankfurt.shape)
df_Frankfurt.head()
```

```
(47, 2)
```

	No	City district (Stadtteil)
0	01	Altstadt
1	02	Innenstadt
2	03	Bahnhofsviertel
3	04	Westend-Süd
4	05	Westend-Nord

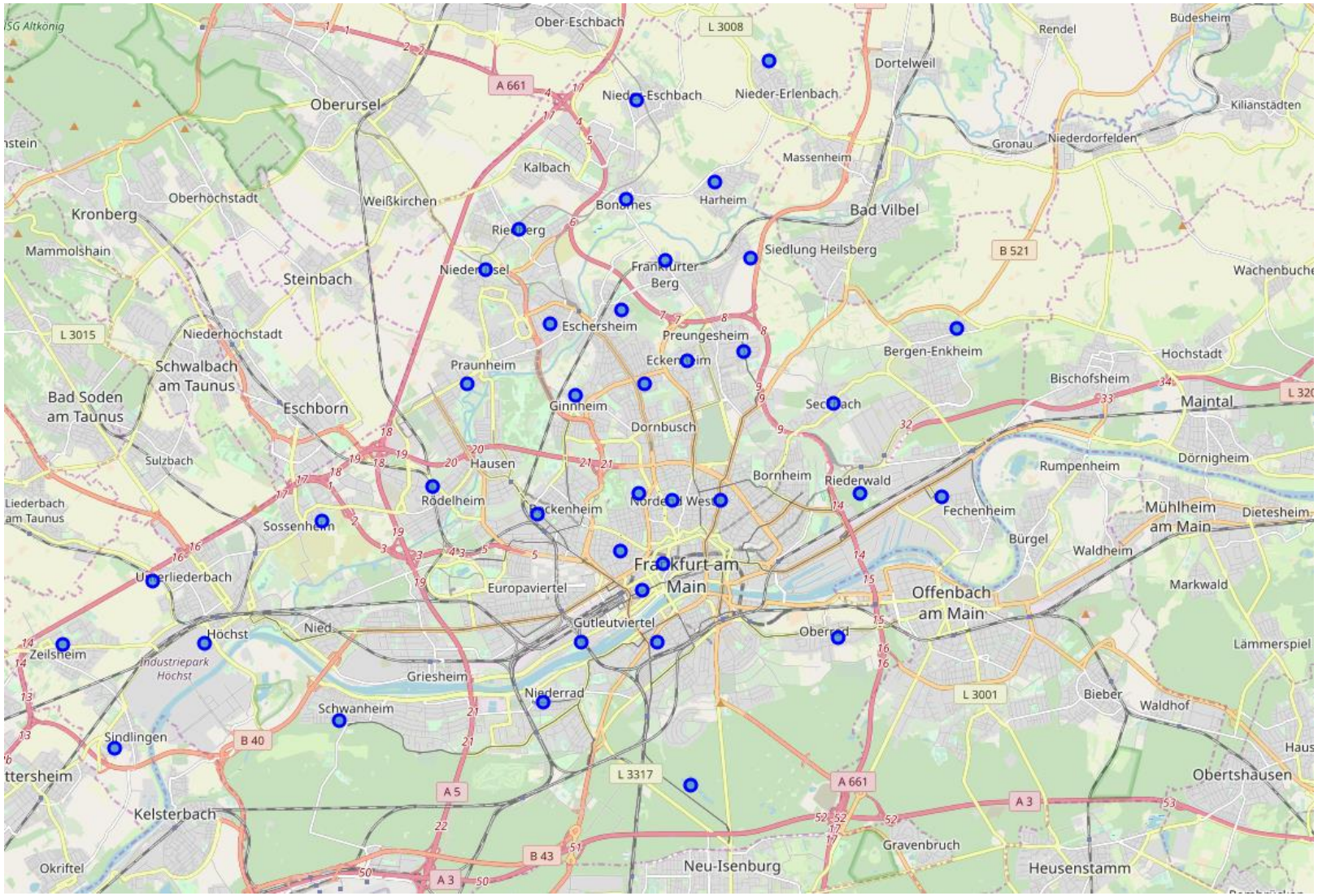
```
# Drop the last row, because it has summary information of Frankfurt.
df_Frankfurt.drop(df_Frankfurt.tail(1).index,inplace=True) # drop last n rows
```

```
geo_Frank = Nominatim(user_agent="Frankfurt_Explorer")
df_Frankfurt['Latitude'] = df_Frankfurt['City district (Stadtteil)'].apply(geo_Frank.geocode).apply(lambda x: (x.latitude))
df_Frankfurt['Longitude'] = df_Frankfurt['City district (Stadtteil)'].apply(geo_Frank.geocode).apply(lambda x: (x.longitude))
print (df_Frankfurt.shape)
df_Frankfurt.head()
```

```
(46, 4)
```

	No	City district (Stadtteil)	Latitude	Longitude
0	01	Altstadt	31.504619	34.464127
1	02	Innenstadt	50.112878	8.674922
2	03	Bahnhofsviertel	50.107741	8.668736
3	04	Westend-Süd	50.115245	8.662270
4	05	Westend-Nord	50.126356	8.667921

Data Description - Frankfurt



Function to get nearby venues using Foursquare API

2. Explore Neighborhoods in Frankfurt

Let's create a function to repeat the same process to all the neighborhoods in Manhattan

```
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)
```

```
LIMIT = 100 # limit of number of venues returned by Foursquare API
radius = 500 # define radius

frankfurt_venues = getNearbyVenues(names=df_Frankfurt['City district (Stadtteil)'],
                                   latitudes=df_Frankfurt['Latitude'],
                                   longitudes=df_Frankfurt['Longitude']
                                  )
print(frankfurt_venues.shape)
frankfurt_venues.head()
```

Foursquare API Data - Frankfurt

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Altstadt	31.504619	34.464127	حمام السمرة	31.504831	34.465252	Spa
1	Altstadt	31.504619	34.464127	Alnaffar Bowtique	31.505054	34.462776	Boutique
2	Altstadt	31.504619	34.464127	Downtown dental clinic	31.507576	34.462728	Health & Beauty Service
3	Altstadt	31.504619	34.464127	Mazaya cafe	31.507920	34.466423	Bistro
4	Altstadt	31.504619	34.464127	In The Hell!	31.501540	34.467350	Moving Target

```
1 frankfurt_venues.groupby('Neighborhood').count()
```

Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
Altstadt	5	5	5	5	5	5
Bahnhofsviertel	100	100	100	100	100	100
Bergen-Enkheim	2	2	2	2	2	2
Berkersheim	2	2	2	2	2	2
Bockenheim	22	22	22	22	22	22
Bonames	8	8	8	8	8	8
Bornheim	4	4	4	4	4	4
Dornbusch	6	6	6	6	6	6
Eckenheim	6	6	6	6	6	6
Eschersheim	4	4	4	4	4	4
Fechenheim	4	4	4	4	4	4
Flughafen	34	34	34	34	34	34
Frankfurter Berg	5	5	5	5	5	5
Ginnheim	9	9	9	9	9	9
Griesheim	2	2	2	2	2	2
Gutleutviertel	11	11	11	11	11	11
Harheim	4	4	4	4	4	4
Heddernheim	4	4	4	4	4	4
Höchst	2	2	2	2	2	2
Innenstadt	100	100	100	100	100	100
Kalbach-Riedberg	12	12	12	12	12	12
Nied	1	1	1	1	1	1

Data Description – New York City

```
for data in neighborhoods_data:
    borough = neighborhood_name = data['properties']['borough']
    neighborhood_name = data['properties']['name']

    neighborhood_latlon = data['geometry']['coordinates']
    neighborhood_lat = neighborhood_latlon[1]
    neighborhood_lon = neighborhood_latlon[0]

    neighborhoods = neighborhoods.append({'Borough': borough,
                                          'Neighborhood': neighborhood_name,
                                          'Latitude': neighborhood_lat,
                                          'Longitude': neighborhood_lon}, ignore_index=True)
```

Quickly examine the resulting dataframe.

```
neighborhoods.head()
```

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

Note

- Will do the same for London

How the data will be used to provide recommendation?

For each of cities (Frankfurt, New York City, and London), we'll take the following steps

1. Import the necessary python libraries.
2. Scrape the main neighborhoods from the links provided in the previous slide.
3. Using the geopy.geocoders python library fetch each neighborhood's latitude and longitude
4. Write a custom function to use the Foursquare API to fetch the nearby venues for each of the neighborhoods in each of city.
5. Cleanse the data when necessary.
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.