Problem Discussion & Background

With the advances in globalization and remote work, it is increasingly frequent that employees, entrepreneurs and self-employed professionals have opportunities to relocate to different cities across the world to work. However, one does not always have the ability to visit and experience the cities they will live in before they actually move. This can be stressful for families as they move into different cities they do not know and may have a hard time to get adapted to.

This project aims to provide insights on how similar certain cities across the world are, from the point of view of availability of commercial and services venues, according to the local population.

For example, a Latin American professional who wants to move to the U.S. can choose to get insights about how similar American cities are to their home city. If, for example, Latino restaurants, parks and shopping malls are very prevalent in their home city, this tool will inform them of cities in America with the most similar profiles.

Target Audience

The target audience for this project is the current and future expat community, composed by professionals who have relocated or want to relocate to different countries. For such professionals, it is always important to learn more about their prospects of personal life in the place they are about to move to.

Obtaining information on the commonly offered commercial, service and leisure options will allow the audience to build an initial perception of the top cities in the country they intend to live in, imagine how their personal life could be compared to their home city, and effectively prioritize their job search.

Data Description

The external data sources used for this project are:

- Foursquare venue review data – using Foursquare API (https://developer.foursquare.com)
- Simple Maps World Cities open database – database of the world's cities and towns (https://simplemaps.com/data/world-cities)

Below are some examples of features we can extract from each dataset:

Foursquare – Extract venue information for a selected venue; view user scores and reviews; view user comments; get nearest places to a given location

Simple Maps – City or town name, country, isocode, population, capital city status, and geo localization

Methodology

The methodology approach to solve the presented problem is the following:

1. Capture user input as to the current city they live in;
2. Capture user input as to the current country they want to move to;
3. Use Simple Maps database to get coordinates for venue review search;
4. Use Foursquare data to obtain the current city profile and for the top 20 cities in the destination country;
5. Use an Euclidian distance function to calculate distance between cities;
6. Rank the top 5 cities most similar to the input city, and present results to user.

Overview

To test the model developed and get initial results, the following use case was analyzed: the user currently lives in Paris and wants to move to the United States.

User inputs: current city and destination country (for comparison) ¶

```
In [ ]:  current_city = 'Paris'

         destination_country = 'United States'
```

The code proceeded to get localization data for the current city, and for the top 20 cities in the country of choice.

Getting location data for current_city and top 20 cities in destination_country

```
In [ ]:  current_df = df[df['city']=='Paris'].head(1)

         destination_df = df[df['country']== destination_country].head(20)

         print(current_df)
         print()
         print(destination_df)
```

```
In [283]:  print(current_df)
           print()
           print(destination_df)

              city      lat     lng country  population
           33  Paris  48.8566  2.3522  France  11020000.0

                       city      lat       lng        country  population
           12       New York  40.6943  -73.9249  United States  18713220.0
           27    Los Angeles  34.1139 -118.4068  United States  12750807.0
           49        Chicago  41.8373  -87.6862  United States   8604203.0
           93          Miami  25.7839  -80.2102  United States   6445545.0
           107        Dallas  32.7936  -96.7662  United States   5743938.0
```

The next step is then to obtain Foursquare venue data on both the current city, and the list of 20 candidate destination cities obtained in the step above. This is achieved with the function getNearbyVenues :

Function to get venues around a specific location ¶

```
In [ ]:  def getNearbyVenues(names, latitudes, longitudes, radius=1000):

             venues_list=[]
             for name, lat, lng in zip(names, latitudes, longitudes):

                 # create the API request URL
                 url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll=
                     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 = ['city',
                           'city Latitude',
                           'city Longitude',
```

All the data obtained needs to be re-grouped and pre-processed in order to be set up for further analysis:

Getting foursquare data for current & destination locations

```
In [ ]:    current_venues = getNearbyVenues(current_df['city'], current_df['lat'], current_df['lng'])

           destination_venues = getNearbyVenues(destination_df['city'], destination_df['lat'], destination_df['lng'])

In [273]:  all_venues = pd.concat([current_venues, destination_venues])

In [ ]:    # one hot encoding
           venues_onehot = pd.get_dummies(all_venues[['venue Category']], prefix="", prefix_sep="")

           # add city column back to dataframe
           venues_onehot['city'] = all_venues['city']

           # group results by city
           venues_grouped = venues_onehot.groupby('city').mean().reset_index()

           # check intermediate results
           venues_grouped
```

All cities analyzed are now part of the dataframe called `venues_grouped`. Each column has the frequency of each venue category within the respective city:

```
In [284]:  venues_grouped
Out[284]:
```

	city	Alsatian Restaurant	Alternative Healer	American Restaurant	Animal Shelter	Antique Shop	Arepa Restaurant	Argentinian Restaurant	Art Gallery	Art Museum	...	Turkish Restaurant	Used Bookstore	Vegetarian / Vegan Restaurant
0	Atlanta	0.00	0.000000	0.000000	0.000000	0.0000	0.000000	0.00	0.000000	0.00000	...	0.00	0.00	0.000000
1	Boston	0.00	0.000000	0.062500	0.000000	0.0000	0.000000	0.00	0.000000	0.00000	...	0.00	0.00	0.000000
2	Brooklyn	0.00	0.000000	0.000000	0.000000	0.0000	0.000000	0.00	0.000000	0.00000	...	0.00	0.00	0.011236
3	Chicago	0.00	0.000000	0.000000	0.000000	0.0000	0.000000	0.00	0.000000	0.00000	...	0.00	0.00	0.000000
4	Dallas	0.00	0.000000	0.000000	0.000000	0.0000	0.000000	0.00	0.025641	0.00000	...	0.00	0.00	0.000000
5	Denver	0.00	0.000000	0.043478	0.000000	0.0000	0.000000	0.00	0.000000	0.00000	...	0.00	0.00	0.000000
6	Detroit	0.00	0.000000	0.000000	0.000000	0.0000	0.000000	0.00	0.000000	0.00000	...	0.00	0.00	0.000000
7	Houston	0.00	0.000000	0.037500	0.000000	0.0125	0.000000	0.00	0.000000	0.00000	...	0.00	0.00	0.012500
8	Los Angeles	0.00	0.000000	0.000000	0.000000	0.0000	0.000000	0.00	0.000000	0.00000	...	0.00	0.00	0.000000
9	Miami	0.00	0.000000	0.021739	0.000000	0.0000	0.000000	0.00	0.000000	0.00000	...	0.00	0.00	0.000000
10	Minneapolis	0.00	0.000000	0.000000	0.000000	0.0000	0.000000	0.00	0.000000	0.03125	...	0.00	0.00	0.000000
11	New York	0.00	0.000000	0.010000	0.000000	0.0000	0.000000	0.00	0.010000	0.00000	...	0.00	0.01	0.020000
12	Paris	0.01	0.000000	0.000000	0.000000	0.0000	0.000000	0.00	0.040000	0.02000	...	0.00	0.00	0.000000
13	Philadelphia	0.00	0.000000	0.034483	0.034483	0.0000	0.000000	0.00	0.000000	0.00000	...	0.00	0.00	0.000000

The next step is to define a function that receives the dataframe above and a chosen city as inputs, and calculates the Euclidean Distance from the selected city to every other city in the dataframe.

The definition of Euclidean Distance used here, given a pair of cities, is the sum of the absolute difference between each frequency shown in matching columns. Cities with identical profiles will have a distance of zero and, the higher the distance, the more different those two cities are from each other.

The following code performs the Euclidian Distance algorithm:

Define function to calculate euclidian distance between cities

```
In [275]:  def euclidian_distance(venueslist, cityname):

               venueslist = venues_grouped
               cityname = 'Paris'
               point1 = venueslist[venueslist['city']==cityname].reset_index().drop(['index'], axis = 'columns')
               ans = pd.DataFrame(columns=['city','distance'])

               for i in venueslist['city']:
                   if i != cityname:
                       point2 = venueslist[venueslist['city']==i].reset_index().drop(['index'], axis = 'columns')
                       dist = 0
                       for j in point1.columns.drop(['city']):
                           dist = dist + abs(point1.loc[0,j] - point2.loc[0,j])
                       ans = ans.append({'city':i, 'distance':dist},ignore_index=True)

               return ans
```

Final Results

The final outputs are shown below. The top 20 cities in the United States were ranked, top to bottom, by the largest similarity (lowest distance) to Paris, France. Therefore, someone who wants to move from Paris to the US now knows that the city with the highest likelihood to find the same types of shops, restaurants and services as they would back home is San Francisco.

Final result is shown below in table format:

```
In [276]:  x = euclidian_distance(venues_grouped, 'Paris')
           x['lat'] = destination_df[['city', 'lat', 'lng']].sort_values('city').rese
           x['lng'] = destination_df[['city', 'lat', 'lng']].sort_values('city').rese
           print('List of top 20 cities by highest similarity (lowest Euclidian dista
           x[['city','distance']].sort_values('distance')

           List of top 20 cities by highest similarity (lowest Euclidian distance):

Out[276]:
```

	city	distance
16	San Francisco	1.400000
7	Houston	1.415000
19	Washington	1.580000
17	Seattle	1.580000
11	New York	1.660000
5	Denver	1.713043
9	Miami	1.733043
2	Brooklyn	1.750112
14	Queens	1.755000
10	Minneapolis	1.760000
15	San Diego	1.794545
3	Chicago	1.840000
0	Atlanta	1.860000
4	Dallas	1.877436
6	Detroit	1.880000
1	Boston	1.940000
12	Philadelphia	1.940000
18	Tampa	1.960000
13	Phoenix	1.960000
8	Los Angeles	1.980000

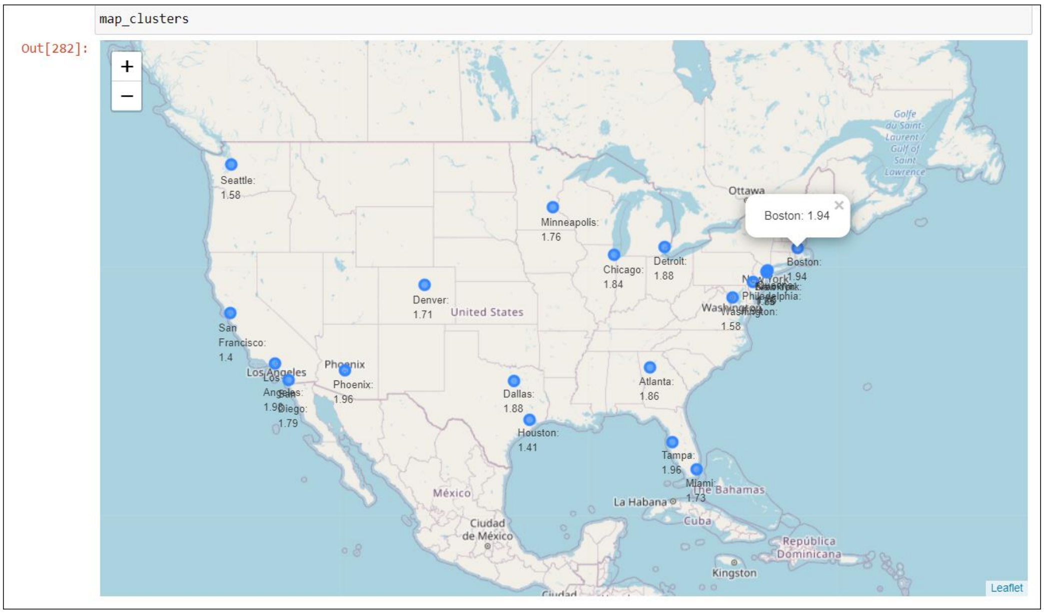
Results discussion

The results obtained are according to expectations, as among the top cities in the US, cities well known for having high immigrant influx and international influence are ranking highest.

Interestingly, cities with smaller geographical areas also tend to rank high due to the dense number of options found within a radius of their city centers. This is another feature that can be further explored in future versions of this project.

The model also provides a map view of the same results, plotting the candidate destination cities and the respective distances to the home city.

On the East Coast, the data is cluttered as many cities are very close to each other. Therefore, the map also has a tooltip menu which shows the city name and score upon clicking.



Conclusion

The model developed for this project worked well within expectations. It is a very simple model, and some ideas to improve and explore further have been brought up in the previous sections. Overall, this has been an interesting exercise and a nice way to consolidate learnings from the Data Science professional certificate course. There is an impressive amount of data available and easily accessible online, which can greatly improve the output of personal and business projects.