Battle of Neighbourhoods

Prateek Shamprasad Pawar

Email: prateekspawar@gmail.com

1)Introduction:

1.1. Background:

Tom is a Resident of Parkwoods, Toronto. His family includes his wife, 10-year-old son and himself. His firm is offering him a promotion with great future prospects but he has to relocate to New York for this opportunity. Tom is inclined to accept this offer and his only problem is finding a similar great neighbourhood in New York as Parkwoods.

1.2. Quantification:

Tom decides to find out the most important factors that make Parkwoods a great neighbourhood for his family. If he can quantify the compatibility of a neighbourhood, it will make the job of finding a similar neighbourhood easy. After giving it considerable thought, he came up with the following factors making a good neighbourhood for them:

1) Plenty of Shops and Services making daily life easy.
2) A great number of food venues available.
3) Good transportation services.
4) Less Nightlife spots in the neighbourhood.

1.3. Problem:

This makes the problem clearer. The steps involved in problem-solving are:

1) Find all the venues in the neighbourhood of Parkwoods.
2) Categorize the venues in different categories. i.e. food, shops, transport etc.
3) Assign proper weight to each category according to the factors specified above.
4) Find the total score of the neighbourhood using the weights and number of venues in each category.
5) Find the total score of each neighbourhood in New York using the same weighs.
6) Separate the neighbourhoods having total score more than Parkwoods, same as Parkwoods, less than Parkwoods.
7) Provide the separated list of neighbourhoods to Tom to help him decide which one to select for relocation.

Data:

For finding out all the venues in the neighbourhood, we will use the Foursquare API. To call in the Foursquare database, we need Latitude and Longitude values of each of the neighbourhood.

a) Neighbourhoods in New York:

1)The geospatial data of all the neighbourhoods in New York is available in .json file at https://geo.nyu.edu/catalog/nyu_2451_34572 .
2)We can extract data about Neighbourhood name, Latitude, Longitude from the file using json.load() function from json library.
3) We can append this data in a Pandas DataFrame having columns Neighborhood, Latitude and Longitude.

```
In [7]: ny.head()
```

Out[7]:

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

4) Then using the Foursquare API request of venues, we can find all the venues in each neighbourhood of New York.

b) Neighbourhoods in Toronto:

1) The table having the name of each neighbourhood in Toronto and its postal code is at https://en.wikipedia.org/wiki/List_of_postal_codes_of_Canada:_M .
2) The csv file having Latitude and Longitudes for each postal code in Toronto is at https://cocl.us/Geospatial_data/Geospatial_Coordinates.csv .
3) We can combine these two datasets to make one DataFrame having names of Neighbourhoods and their Latitudes and Longitudes.

```
In [10]: tor.head()
```

Out[10]:

	Neighborhood	Latitude	Longitude
0	Parkwoods	43.753259	-79.329656
1	Victoria Village	43.725882	-79.315572
2	Harbourfront	43.654260	-79.360636
3	Lawrence Heights,Lawrence Manor	43.718518	-79.464763
4	Queen's Park	43.662301	-79.389494

4) Then using the Foursquare API request of venues, we can find all the venues in each neighbourhood of Toronto for further analysis.

3.Methodology:

3.1 Acquiring Venues Data:

Once we have the data frame of neighbourhoods and their coordinates, we can make venues request to the Foursquare API. The request returns a .json file containing all the venues in the 1000 m radius of neighbourhood. We write a function to extract only the relevant data of each venue from the json file like 'Neighbourhood', 'Neighbourhood Latitude', 'Neighbourhood Longitude', Venue name ', 'Venue distance', 'Venue Category'. We import this data into new Data Frames named ny_venues and tor_venues respectively.

```
In [15]: ny_venues.head()
```

Out[15]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue distance	Venue Category
0	Wakefield	40.894705	-73.847201	Lollipops Gelato	127	Dessert Shop
1	Wakefield	40.894705	-73.847201	Ripe Kitchen & Bar	798	Caribbean Restaurant
2	Wakefield	40.894705	-73.847201	Ali's Roti Shop	822	Caribbean Restaurant
3	Wakefield	40.894705	-73.847201	Jackie's West Indian Bakery	686	Caribbean Restaurant
4	Wakefield	40.894705	-73.847201	Carvel Ice Cream	483	Ice Cream Shop

```
In [17]: tor_venues.head()
```

Out[17]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue distance	Venue Category
0	Parkwoods	43.753259	-79.329656	Allwyn's Bakery	833	Caribbean Restaurant
1	Parkwoods	43.753259	-79.329656	Brookbanks Park	245	Park
2	Parkwoods	43.753259	-79.329656	Tim Hortons	866	Café
3	Parkwoods	43.753259	-79.329656	A&W Canada	852	Fast Food Restaurant
4	Parkwoods	43.753259	-79.329656	Bruno's valu-mart	889	Grocery Store

3.2. Defining Categories:

We can separate the venues into different categories in any appropriate manner. Here we use the categories as in Foursquare documentations with few modifications. Then we have 11 categories of venues as given below:

1.Arts & Entertainment

2.College & university

3.Event

4.Food

5.Nightlife Spot

6.Outdoors & Recreation

7.Professional & Other Places

8.school

9.Residence

10.Shop & Service

11.Travel & Transport

We can use these categories to distinguish the venues. Add a category column to the venues data fra me and specify a category of each venue in it we get both the DataFrames modified with categories. This can be done by using a function which compares the name of each venue category with the lists containing names of all the venues in each category. The function for the ny_venues DataFrame is as follows:(here c1, c2….c11 are lists containing venues in respective categories.)

```
In [33]: for i in range(ny_venues.shape[0]):
             if(ny_venues.iloc[i,5] in c1):
                 ny_venues.iloc[i,6]=1
             elif (ny_venues.iloc[i,5] in c2):
                 ny_venues.iloc[i,6]=2
             elif (ny_venues.iloc[i,5] in c3):
                 ny_venues.iloc[i,6]=3
             elif (ny_venues.iloc[i,5] in c4):
                 ny_venues.iloc[i,6]=4
             elif (ny_venues.iloc[i,5] in c5):
                 ny_venues.iloc[i,6]=5
             elif (ny_venues.iloc[i,5] in c6):
                 ny_venues.iloc[i,6]=6
             elif (ny_venues.iloc[i,5] in c7):
                 ny_venues.iloc[i,6]=7
             elif (ny_venues.iloc[i,5] in c8):
                 ny_venues.iloc[i,6]=8
             elif (ny_venues.iloc[i,5] in c9):
                 ny_venues.iloc[i,6]=9
             elif (ny_venues.iloc[i,5] in c10):
                 ny_venues.iloc[i,6]=10
             elif (ny_venues.iloc[i,5] in c11):
                 ny_venues.iloc[i,6]=11
```

3.3. One-Hot Encoding:

For making the data processing easy, there should be one column for each category indicating whether the venue is of that category or not in binary. This process is called ‘one_hot encoding’. Using get_dummies() function on ‘Category’ column we can add these columns to original dataframe.

After adding the dummy columns and removing category column with some processing the dataframes become:

```
In [70]: tor_cat.head()
```

Out[70]:

	Neighborhood	Venue Category	Venue distance	cat__1	cat__2	cat__3	cat__4	cat__5	cat__6	cat__7	cat__8	cat__9	cat__10	cat__11
0	Parkwoods	Caribbean Restaurant	833	0	0	0	1	0	0	0	0	0	0	0
1	Parkwoods	Park	245	0	0	0	0	0	1	0	0	0	0	0
2	Parkwoods	Café	866	0	0	0	1	0	0	0	0	0	0	0
3	Parkwoods	Fast Food Restaurant	852	0	0	0	1	0	0	0	0	0	0	0
4	Parkwoods	Grocery Store	889	0	0	0	0	0	0	0	0	0	1	0

```
In [71]: ny_cat.head()
```

Out[71]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue Category	Venue distance	cat__1	cat__2	cat__3	cat__4	cat__5	cat__6	cat__7	cat__8	cat__9	cat__10	cat__11
0	Wakefield	40.894705	-73.847201	Dessert Shop	127	0	0	0	1	0	0	0	0	0	0	0
1	Wakefield	40.894705	-73.847201	Caribbean Restaurant	798	0	0	0	1	0	0	0	0	0	0	0
2	Wakefield	40.894705	-73.847201	Caribbean Restaurant	822	0	0	0	1	0	0	0	0	0	0	0
3	Wakefield	40.894705	-73.847201	Caribbean Restaurant	686	0	0	0	1	0	0	0	0	0	0	0
4	Wakefield	40.894705	-73.847201	Ice Cream Shop	483	0	0	0	1	0	0	0	0	0	0	0

3.4. Data conditioning (Venue Distance):

Here all the venue data is binary i.e. 0 or 1. To incorporate the importance of distance of venue from the neighbourhood, we can condition the influence of venue as:

1) If the distance is less than 500 m the influence remains 1.
2) If the distance is more than 500m the influence becomes 0.5.

This conditioning is done on all the venues and all the neighbourhoods of both datasets and the ‘Venue Distance’ Column is removed.

3.5. Grouping:

Now we group the venues according to their Neighbourhoods using groupby() function. During grouping, we sum values in category columns of each venue. Hence, we get a DataFrame where the total number of venues corresponding to each category in each neighbourhood is shown. The grouped DataFrames look as follows:

```
In [75]: ny_cat.head()
```

Out[75]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	cat__1	cat__2	cat__3	cat__4	cat__5	cat__6	cat__7	cat__8	cat__9	cat__10	cat__11
0	Allerton	40.865788	-73.859319	0.5	0.0	0.0	14.5	1.0	1.5	0.0	0.0	0.0	2.5	0.0
1	Annadale	40.538114	-74.178549	0.0	0.0	0.0	9.5	2.0	2.0	0.0	0.5	0.0	1.0	1.0
2	Arden Heights	40.549286	-74.185887	0.0	0.0	0.0	6.0	0.0	2.0	0.0	0.0	0.0	5.5	1.5
3	Arlington	40.635325	-74.165104	0.0	0.0	0.0	2.0	0.5	0.0	0.0	0.0	0.0	4.5	2.0
4	Arrochar	40.596313	-74.067124	1.0	0.0	0.0	10.0	0.0	4.0	0.0	0.0	0.0	1.0	2.0

```
In [76]: tor_cat.head()
```

Out[76]:

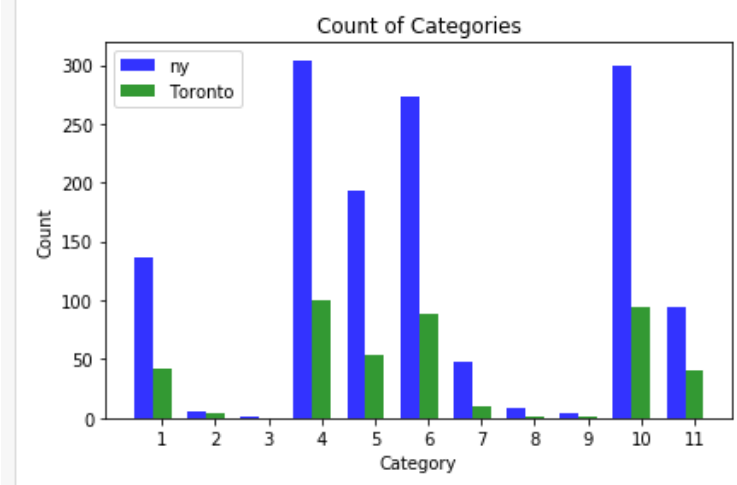
	Neighborhood	cat__1	cat__2	cat__3	cat__4	cat__5	cat__6	cat__7	cat__8	cat__9	cat__10	cat__11
0	Adelaide,King,Richmond	2.0	0.0	0.0	19.0	2.0	3.0	1.0	0.0	0.0	1.0	2.0
1	Agincourt	0.5	0.0	0.0	12.0	1.0	0.5	0.0	0.0	0.0	3.0	0.0
2	Agincourt North,L'Amoreaux East,Milliken,Steel...	0.0	0.0	0.0	11.0	0.0	2.0	0.5	0.0	0.0	2.0	0.0
3	Albion Gardens,Beaumond Heights,Humbergate,Jam...	0.0	0.0	0.0	6.5	0.0	0.5	0.0	0.0	0.0	5.0	0.5
4	Alderwood,Long Branch	0.0	0.0	0.0	5.0	1.0	4.0	0.0	0.0	0.0	7.0	0.5

3.6. Category Selection:

It is observed in the grouped data that there are very a smaller number of venues in some of the categories in all the neighbourhoods. These categories should be neglected for further processing.

To find out which categories to neglect, consider the nonzero number of values in each category. This indicates the number of neighbourhoods having at least one venue of that category.

This data is expressed as a bar graph as follows:



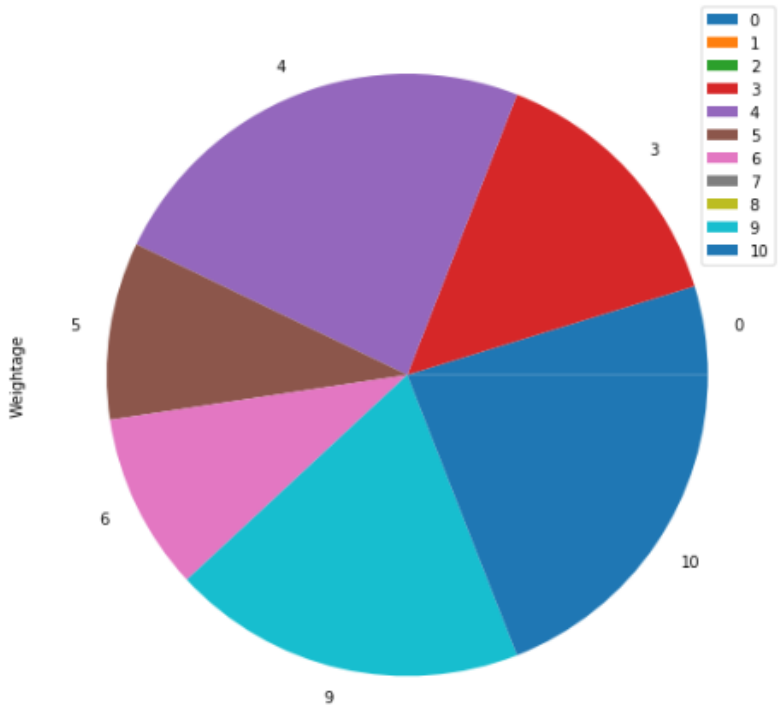
From the bar graph, we can conclude that categories 2,3,8,9 can be neglected because of lower occurrence in both cities.

3.7. Category Weightage:

After selecting which categories to consider, we have to assign some weight to the categories. This weight will depend upon importance given by Tom to each category. Depending on the factors specified in the introduction we can assign the following weight to categories:

No	Category	Weight
1	Arts & Entertainment	0.05
2	College & university	0
3	Event	0
4	Food	0.15
5	Nightlife Spot	-0.20
6	Outdoors & Recreation	0.10
7	Professional & Other Places	0.10
8	school	0.0
9	Residence	0.0
10	Shop & Service	0.20
11	Travel & Transport	0.20

Here the yellow background defines neglected categories with 0 weight. The category ‘Nightlife Spot’ is given negative weight because one of the factors requires that this category venues should be less in the neighbourhood. The weights can be represented as a Pie Diagram:



3.8. Finding Total Score:

Once the weight of each category is defined, we can find the total score of each neighbourhood by taking a weighted sum of all the categories for that neighbourhood. We add a new column named ‘Total’ to the DataFrames and append total score of each neighbourhood in that column. Then dropping the columns of each category, we get new dataframes with only the neighbourhoods and respective total scores. We then sort the neighbourhoods in the descending order of their total scores.

```
In [133]: tor_tot.head()
```

Out[133]:

	Neighborhood	Total
0	Adelaide,King,Richmond	3.550
1	Agincourt	2.275
2	Agincourt North,L'Amoreaux East,Milliken,Steel...	2.300
3	Albion Gardens,Beaumond Heights,Humbergate,Jam...	2.125
4	Alderwood,Long Branch	2.450

```
In [137]: ny_tot.head()
```

Out[137]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Total
0	Park Slope	40.672321	-73.977050	5.00
1	Bay Terrace	40.782843	-73.776802	4.85
2	Soho	40.722184	-74.000657	4.55
3	Flatiron	40.739673	-73.990947	4.50
4	Midtown South	40.748510	-73.988713	4.45

We can then find the total score of the ‘Parkwoods’ neighbourhood from tor_tot dataframe.

This is done by first finding the index if ‘Parkwoods’ and then finding the total score corresponding to that index.

3.9. Conditional Separating:

Once we have the total score for ‘Parkwoods’ neighbourhood, we can separate the neighbourhoods in New York in three sections as follows:

1) Neighbourhoods having more Total score than ‘Parkwoods’.
2) Neighbourhoods having same Total score as that of ‘Parkwoods’.
3) Neighbourhoods having less Total score than ‘Parkwoods’.

We separate these neighbourhoods in three dataframes namely: ny_up, ny_same, ny_down.

4.Results:

4.1. Total Score of ‘Parkwoods’:

From the procedure given in 3.8, we found that the index of ‘Parkwoods’ is 37 with a total score of 2.625.

```
In [138]: for i in range(tor_tot.shape[0]):
              if(tor_tot.iloc[i,0]=='Parkwoods'):
                  break
          print(i)
```

```
37
```

```
In [152]: tor_tot.iloc[37,1]
```

```
Out[152]: 2.625
```

4.2. Separated neighbourhoods:

From the procedure as in 3.9, we separated three dataframes of New York neighbourhoods.

```
In [153]: ny_up.head()
```

Out[153]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Total	Tollerance
0	Park Slope	40.672321	-73.977050	5.00	2.375
1	Bay Terrace	40.782843	-73.776802	4.85	2.225
2	Soho	40.722184	-74.000657	4.55	1.925
3	Flatiron	40.739673	-73.990947	4.50	1.875
4	Midtown South	40.748510	-73.988713	4.45	1.825

```
In [145]: ny_up.shape
```

```
Out[145]: (179, 5)
```

In [154]:
```
ny_same.head()
```

Out[154]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Total	Tollerance
179	Cobble Hill	40.687920	-73.998561	2.625	0.0
180	Gramercy	40.737210	-73.981376	2.625	0.0
181	Chelsea	40.594726	-74.189560	2.625	0.0
182	St. George	40.644982	-74.079353	2.625	0.0
183	Elm Park	40.630147	-74.141817	2.625	0.0

In [146]:
```
ny_same.shape
```

Out[146]: (6, 5)

In [155]:
```
ny_down.head()
```

Out[155]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Total	Tollerance
185	Sheepshead Bay	40.586890	-73.943186	2.6	-0.025
186	Spuyten Duyvil	40.881395	-73.917190	2.6	-0.025
187	East Tremont	40.842696	-73.887356	2.6	-0.025
188	Longwood	40.815099	-73.895788	2.6	-0.025
189	Canarsie	40.635564	-73.902093	2.6	-0.025

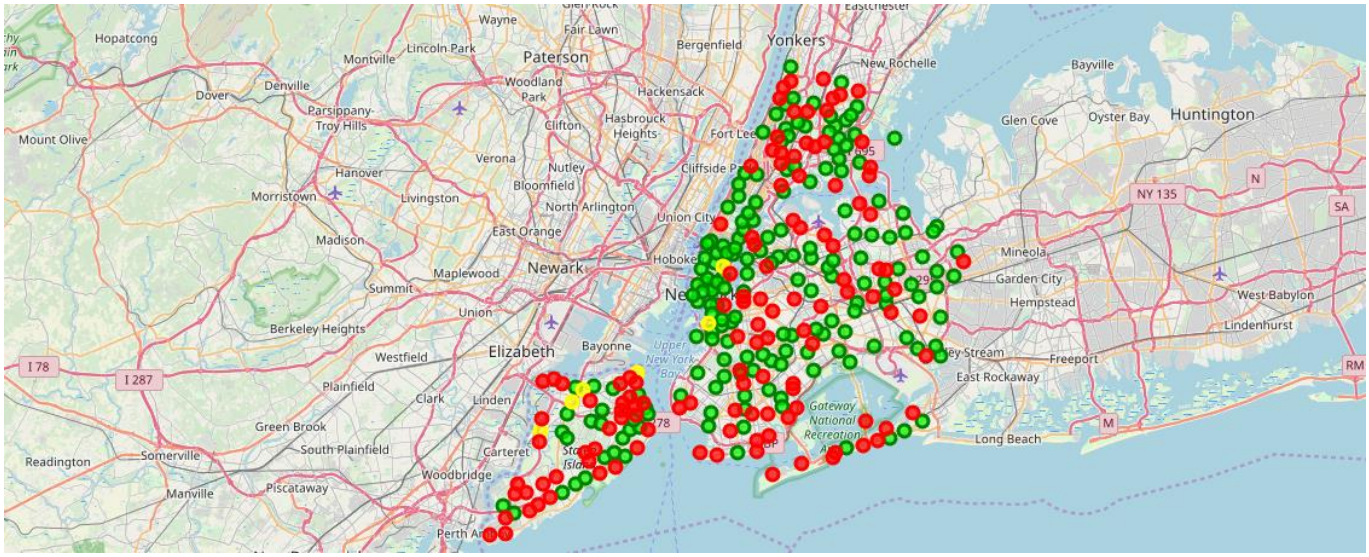
In [147]:
```
ny_down.shape
```

Out[147]: (121, 5)

4.3. Plotting Result on Map:

We can plot the results on a New York city map using folium library. The colour convention used for plotting the neighbourhoods is as follows:

1) ny_up: "Green"
2) ny_same: "Yellow"
3) ny_down: "Red"



5. Conclusion:

From the results, we can conclude that neighbourhood in green are better than 'Parkwoods', in yellow are same as it and the red ones are less compatible than 'Parkwoods' for Tom's family as per factors specified by him.