Introduction/Business Problem Introduction/Business Problem

For an American citizen who lives in New York city, they decided to go on a vacation to a south eastern country, and specifically they has to decide the venue to be in one of the following cities:

1. Kuala Lumpur
2. Bangkok
3. Tokyo

The citizen preference is Historic places, but they also want to compare which city is more similar to New York City in different categories such as restaurants (different types such as Chinese, Spanish…etc.), hotels, cafes and historic places.

The citizen also needs to know what the frequency of each place in each of those cities is! In addition, the probability that they would find different venues in those cities.

Data

I used data of all the above-mentioned cities, and used Foursquare to explore venues at each place using my free account, which makes the data limited to a specific number of observations.

I then collected the category of each venue in all those cities, latitudes and longitudes to plot a geo-map of those venues, started comparing the categories of each city, counting the occurrence of venues in each city and collected them all in one dataset so that you can see frequency of each place. Then we calculate the probability of each venue and make clustering to find the similar cities.

Methodology

I first starting by identifying the link of each city to explore using Foursquare using my client id and client secret. I determined the latitude and longitude first of each city using geolocator and then copied the link and showed the number of observations available for each city and the columns used for comparison. I later imported all the datasets into a list so that I can easier obtain each one of them.

```
For Tokyo, The latitude is: 35.6828387 and Longitude is: 139.7594549
The url for Tokyo: is https://api.foursquare.com/v2/venues/explore?client_id=H2NZQN05FD0V0DFDN1FO5VR2
ZKCDKCKUJPSYEALGQJCLIUQJ&client_secret=CNTMAUZVWIPTZXHE3ZGEQ1AAFHHRP5YY1GIWRPOS2EEIF1YS&ll=35.682838
7,139.7594549&v=20180604&radius=500&limit=30
There are 30 observations and 22 columns for each item around Tokyo
--------------------
For Kuala Lumpur, The latitude is: 3.1516964 and Longitude is: 101.6942371
The url for Kuala Lumpur: is https://api.foursquare.com/v2/venues/explore?client_id=H2NZQN05FD0V0DFDN
1FO5VR2ZKCDKCKUJPSYEALGQJCLIUQJ&client_secret=CNTMAUZVWIPTZXHE3ZGEQ1AAFHHRP5YY1GIWRPOS2EEIF1YS&ll=3.1
516964,101.6942371&v=20180604&radius=500&limit=30
There are 30 observations and 22 columns for each item around Kuala Lumpur
--------------------
For Bangkok, The latitude is: 13.7544238 and Longitude is: 100.4930399
The url for Bangkok: is https://api.foursquare.com/v2/venues/explore?client_id=H2NZQN05FD0V0DFDN1FO5V
R2ZKCDKCKUJPSYEALGQJCLIUQJ&client_secret=CNTMAUZVWIPTZXHE3ZGEQ1AAFHHRP5YY1GIWRPOS2EEIF1YS&ll=13.75442
38,100.4930399&v=20180604&radius=500&limit=30
There are 29 observations and 21 columns for each item around Bangkok
--------------------
For New York City, The latitude is: 14.09212055 and Longitude is: -87.19113829894533
The url for New York City: is https://api.foursquare.com/v2/venues/explore?client_id=H2NZQN05FD0V0DFD
N1FO5VR2ZKCDKCKUJPSYEALGQJCLIUQJ&client_secret=CNTMAUZVWIPTZXHE3ZGEQ1AAFHHRP5YY1GIWRPOS2EEIF1YS&ll=1
4.09212055,-87.19113829894533&v=20180604&radius=500&limit=30
There are 30 observations and 20 columns for each item around New York City
--------------------
All datasets are into city_data list!
```


I then created two definitions that I might use later for different dataframes, which would save much time of repeating the lines of code. A definition to extract each dataset from the list and another on to extract categories of each venue.

```
In [5]: def get_cities(df):
            df0 = df[0]
            df1 = df[1]
            df2 = df[2]
            df3 = df[3]
            return [df0, df1, df2, df3]
```

```
In [6]: def get_category_type(row):
            categories_list = row['categories']
            if len(categories_list) == 0:
                return None
            else:
                return categories_list[0]['name']
```

I explore the dataframe in raw data first before applying any functions to it.

```
In [8]: df_New_York.head()
```

Out[8]:

	referralId	reasons.count	reasons.items	venue.id	venue.name	venue.location.address
0	e-0-5467e80f498ecfc74854fe59-0	0	[{'summary': 'This spot is popular', 'type': '...	5467e80f498ecfc74854fe59	Lúbara	Colonia Tepeyac, Calle Ocotepeque, Avenida Gra...
1	e-0-4e7a391aae60757c759ef263-1	0	[{'summary': 'This spot is popular', 'type': '...	4e7a391aae60757c759ef263	Mandarin Oriental	Col. Tepeyac
2	e-0-4ef1591d93adbace602edea9-2	0	[{'summary': 'This spot is popular', 'type': '...	4ef1591d93adbace602edea9	RadioHouse	Colonia Tepeyac
3	e-0-4d0127cd1ebe6dcb47ae8b91-3	0	[{'summary': 'This spot is popular', 'type': '...	4d0127cd1ebe6dcb47ae8b91	Hacienda Real	Calle Corea del Sur
4	e-0-4b9c54f4f964a520396036e3-4	0	[{'summary': 'This spot is popular', 'type': '...	4b9c54f4f964a520396036e3	Coco Baleadas	NaN

Since there are many different columns for each dataset, I needed to find the common columns between them so that I can use it to compare between the different datasets in a good way, so that was the first step to do before exploring the datasets, so that there are only 19 columns that exist in all dataframes.

```
Out[7]: ['venue.location.country',
 'venue.location.distance',
 'venue.location.labeledLatLngs',
 'venue.location.state',
 'venue.categories',
 'reasons.count',
 'venue.location.address',
 'reasons.items',
 'referralId',
 'venue.location.crossStreet',
 'venue.location.lng',
 'venue.location.lat',
 'venue.id',
 'venue.photos.count',
 'venue.location.cc',
 'venue.photos.groups',
 'venue.location.city',
 'venue.location.formattedAddress',
 'venue.name']
```

And then I started to determine which columns of them specifically that could be used for comparison and I selected('venue.name', 'venue.categories', 'venue.location.lng','venue.location.lat') but we find out that further tuning for our dataset is needed so that I start by removing unnecessary words from the columns names. Moreover, since venue categories has many dictionaries, so that we need to determine which one to show, so that the dataset would include the name of the venue only either short-name or the name itself.

I would also then show the dataset tuned and would show the occurrence of each venue category in each city so that the decision could be made easier.

Out[13]:

	name	categories	lng	lat
0	Adya Hotel Kuala Lumpur	Hotel	101.695623	3.151703
1	Restoran Jai Hind	Indian Restaurant	101.696074	3.151061
2	Cafeteria DBKL	Asian Restaurant	101.694922	3.152154
3	Syawarma Raihani Kebab	Kebab Restaurant	101.696364	3.153069
4	TEH Songket	Bridal Shop	101.695964	3.152254

```
The most common categories to visit in Kuala_Lumpur are:
```

Out[14]:

```
Indian Restaurant                6
Café                             2
Hotel                            2
Coffee Shop                      2
Asian Restaurant                 1
South Indian Restaurant          1
Food Truck                       1
Monument / Landmark              1
Night Market                     1
Men's Store                      1
Bridal Shop                      1
Chettinad Restaurant             1
Kebab Restaurant                 1
Theater                          1
Sporting Goods Shop              1
Art Gallery                      1
Boutique                         1
Vegetarian / Vegan Restaurant    1
Restaurant                       1
Athletics & Sports               1
Flea Market                      1
Gym                              1
Name: categories, dtype: int64
```

I then concated all the datasets into one, so that we can make analysis for it.

Out[19]:

	name	categories	lng	lat	city
0	Adya Hotel Kuala Lumpur	Hotel	101.695623	3.151703	Kuala_Lumpur
1	Restoran Jai Hind	Indian Restaurant	101.696074	3.151061	Kuala_Lumpur
2	Cafeteria DBKL	Asian Restaurant	101.694922	3.152154	Kuala_Lumpur
3	Syawarma Raihani Kebab	Kebab Restaurant	101.696364	3.153069	Kuala_Lumpur
4	TEH Songket	Bridal Shop	101.695964	3.152254	Kuala_Lumpur

I merged all the cities based on the categories they share, and then set the NaN values as zeros, so that now we can do clustering using K-means and find out which cities are similar based on the distribution of the venue categories. I used the probability of each venue existence based on the sample of observations we have as a fraction.

Then I made scaling for our dataset as preparation for segmentation and I chose number of clusters as 2 since we need to find a similar city to NewYork city at least. I plotted the similarity using heatmap as an easier way for visualization for the results as similarity factor is the cluster number they belong to.

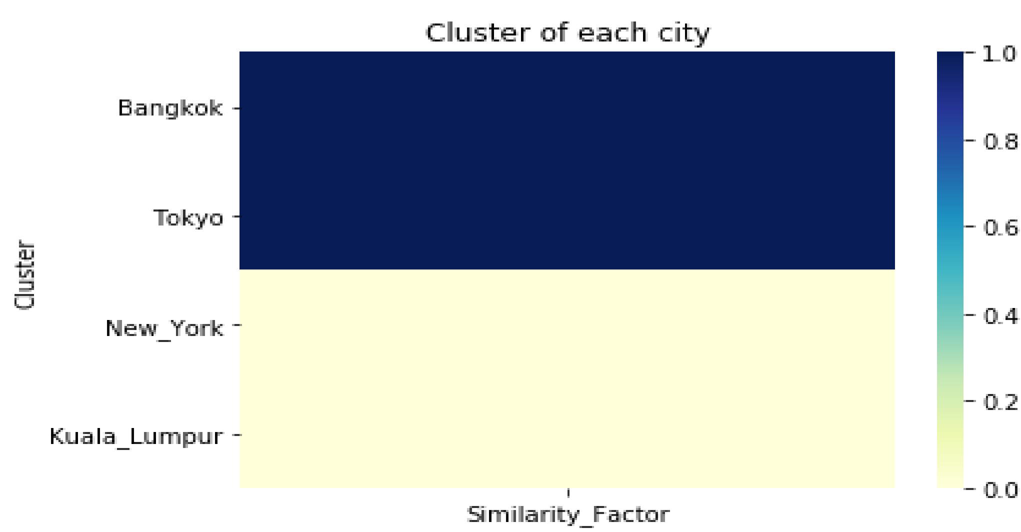
Out[21]:

	American Restaurant	Art Gallery	Art Museum	Asian Restaurant	Athletics & Sports	Bakery	Beer Garden	Bistro	Boutique	Brazilian Restaurant	...	Sports Bar
New_York	0.03	0.00	0.00	0.00	0.00	0.03	0.03	0.03	0.00	0.00	...	0.03
Bangkok	0.00	0.00	0.03	0.00	0.00	0.03	0.00	0.00	0.00	0.00	...	0.00
Tokyo	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03	...	0.00
Kuala_Lumpur	0.00	0.03	0.00	0.03	0.03	0.00	0.00	0.00	0.03	0.00	...	0.00

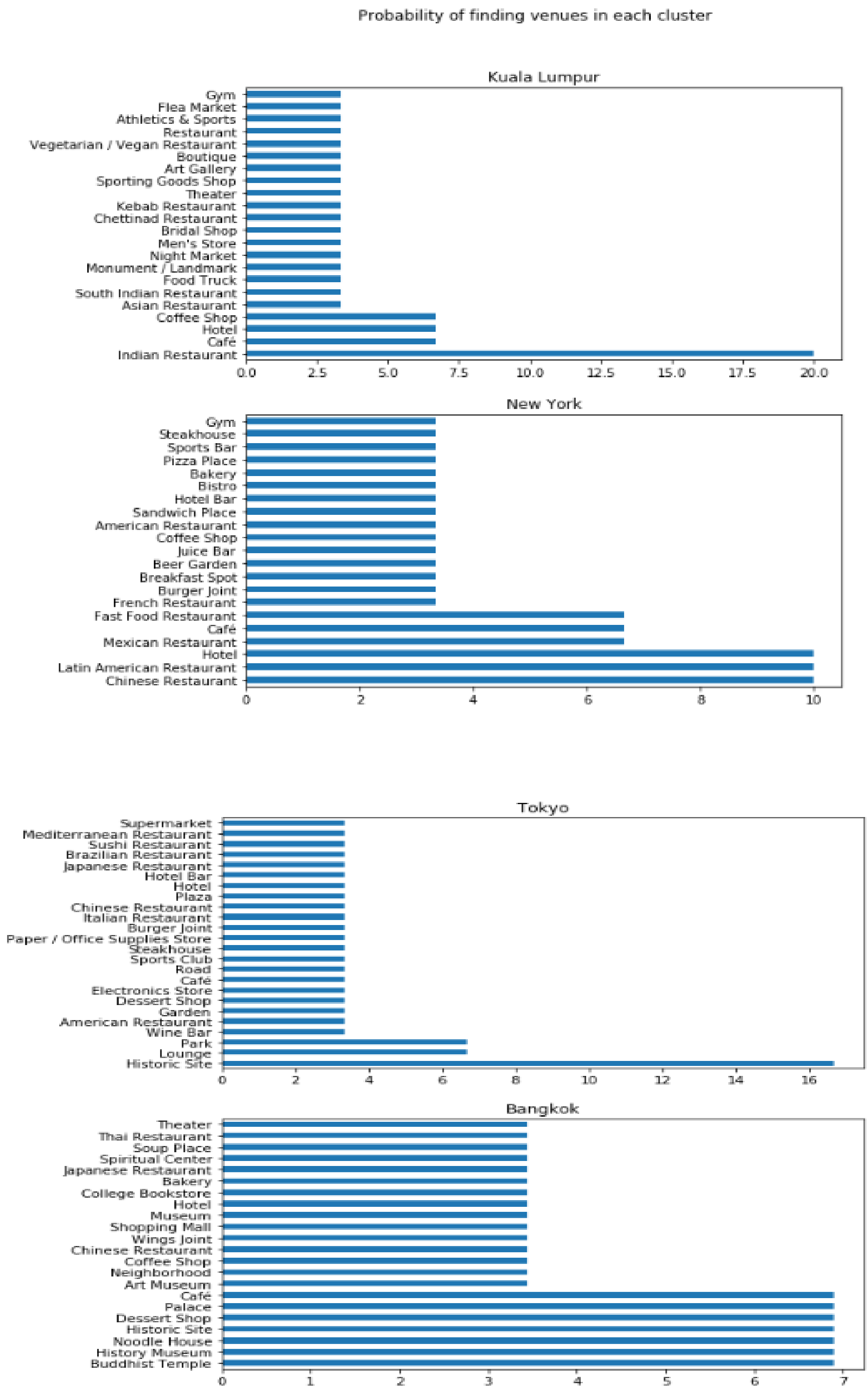
4 rows × 69 columns

Out[33]:

	Similarity_Factor
New_York	0
Bangkok	1
Tokyo	1
Kuala_Lumpur	0



Based on the segmentation, we can compare similar cities and show the most common venues and the probability that a venue would be available nearby when the citizen go to this city.



Results and Observations

We can see that Kuala Lumpur is the most similar city to NewYork city and this is easily figure from the plots that they both have the highest probability of finding hotel, cafes or restaurants nearby your location.

Conclusion

Based on the results, Kuala Lumpur is the most similar city to NewYork so that it would be highly recommended to the tourist yet we can see that Tokyo or Bangkok would only preferred only if the tourist would like to visit many historic places since they have plenty of them