Battle of neighbourhoods: A recommendation engine for the ones on-the-go

Introduction:

“Journeys end in lovers meeting. Every wise man’s son doth know.” A quote from William Shakespeare’s Twelfth Night, that rings true even to this day. For some, their lovers are someone else, towards whom, they are striving to, and to some, their great love is the journey itself. This project was made keeping in mind the latter kind of people. For these people know where they’ve been, but never where they will be. To help with this little problem, a recommendation engine would be ideal. Something that will give these people a suggestion on where their next journey might end, customized to their likings, by having a machine learn their preferences of spots around town and suggesting them other places just like these where they can find like-minded people, interact, or just be.

Data

As the foursquare API is not working properly for me, I’ll be using the Dataset NYC and Tokyo Check-in Dataset by Dingqi Yang [1]

This dataset contains check-ins in NYC and Tokyo collected for about 10 month (from 12 April 2012 to 16 February 2013). It contains 227,428 check-ins in New York city and 573,703 check-ins in Tokyo. Each check-in is associated with its time stamp, its GPS coordinates and its semantic meaning (represented by fine-grained venue-categories). This dataset is originally used for studying the spatial-temporal regularity of user activity in LBSNs.

The dataset can be found here (about 775MB zipped)

This dataset includes long-term (about 10 months) check-in data in New York city and Tokyo collected from Foursquare from 12 April 2012 to 16 February 2013.

It contains two files in tsv format. Each file contains 8 columns, which are:

1. User ID (anonymized)

2. Venue ID (Foursquare)

3. Venue category ID (Foursquare)

4. Venue category name (Fousquare)

5. Latitude

6. Longitude

7. Timezone offset in minutes (The offset in minutes between when this check-in occurred and the same time in UTC)

8. UTC time

The file **dataset_TSMC2014_NYC.txt** contains 227428 check-ins in New York City.

The file **dataset_TSMC2014_TKY.txt** contains 537703 check-ins in Tokyo.

Methodology:

This project can be fragmented into the following parts:

1. Data loading
2. Data preparation
3. Clustering of users on prepared data
4. Re-structuring data for recommendation
5. Finding all recommendations
6. Displaying recommendations on the Map

The main question this project tries to answer is as follows:

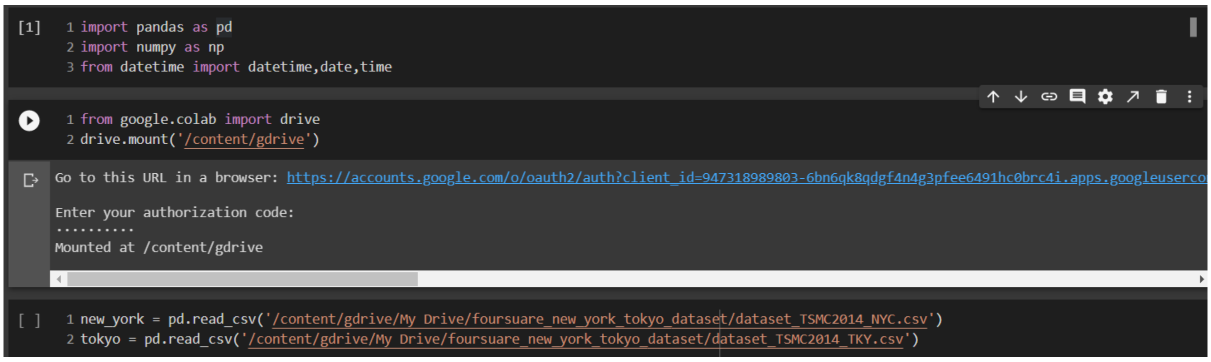
Given sets of location along with the type of location travelled to previously, what are the noteworthy locations that can be travelled to except past locations?

We shall be explaining each part in detail below and show how this project answered this question.

Data loading

This is probably the simplest part of the entire project.

For ease of access, the given files are loaded onto a google drive, and are loaded onto local runtime using Google Colab(Primary Platform for development of the code for this project).



```
[1]  import pandas as pd
     import numpy as np
     from datetime import datetime,date,time
```

```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleuserco

Enter your authorization code:
··········
Mounted at /content/gdrive
```

```
[ ]  new_york = pd.read_csv('/content/gdrive/My Drive/foursuare_new_york_tokyo_dataset/dataset_TSMC2014_NYC.csv')
     tokyo = pd.read_csv('/content/gdrive/My Drive/foursuare_new_york_tokyo_dataset/dataset_TSMC2014_TKY.csv')
```

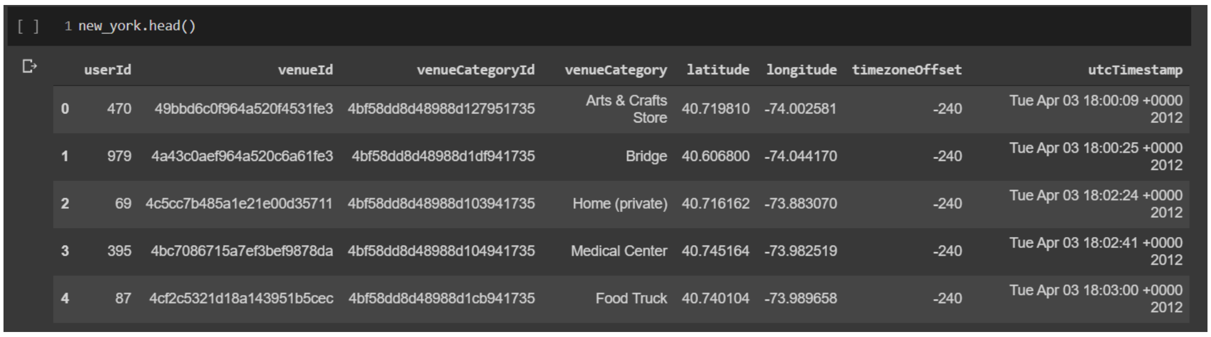
Once the files are uploaded to the local runtime, they are read into pandas dataframes for analysis.

Data preparation:

As any well exposed data scientist would tell you, the models used will not give the desired results as long as the data is not properly structured to the given problem.

To be able to answer our primary question, we need to restructure the data(which is mainly in the form of words and strings, to a numeric format so that it can be used by the machine learning model.

Let’s first have a look at what the data looks like in its raw form:



```
new_york.head()
```

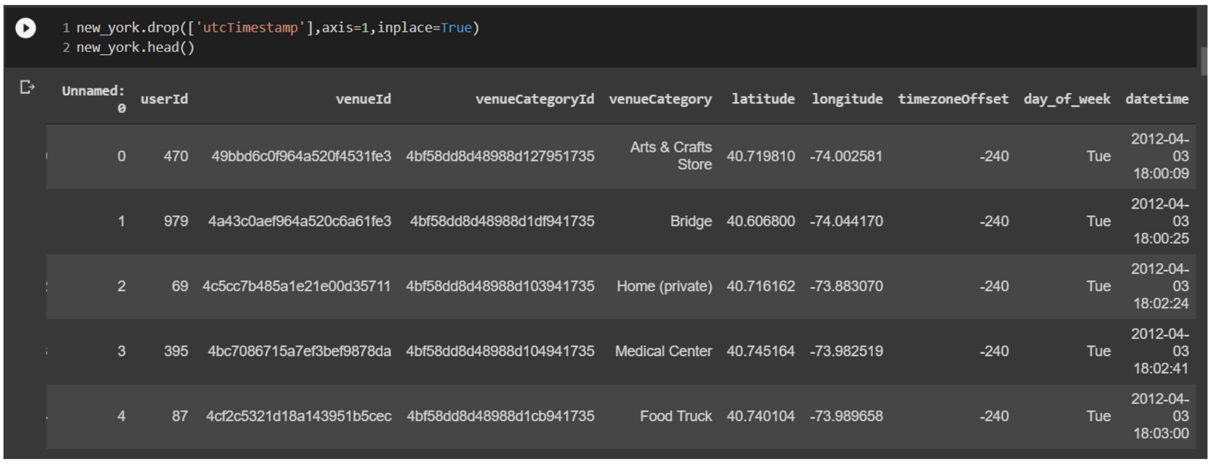
	userId	venueId	venueCategoryId	venueCategory	latitude	longitude	timezoneOffset	utcTimestamp
0	470	49bbd6c0f964a520f4531fe3	4bf58dd8d48988d127951735	Arts & Crafts Store	40.719810	-74.002581	-240	Tue Apr 03 18:00:09 +0000 2012
1	979	4a43c0aef964a520c6a61fe3	4bf58dd8d48988d1df941735	Bridge	40.606800	-74.044170	-240	Tue Apr 03 18:00:25 +0000 2012
2	69	4c5cc7b485a1e21e00d35711	4bf58dd8d48988d103941735	Home (private)	40.716162	-73.883070	-240	Tue Apr 03 18:02:24 +0000 2012
3	395	4bc7086715a7ef3bef9878da	4bf58dd8d48988d104941735	Medical Center	40.745164	-73.982519	-240	Tue Apr 03 18:02:41 +0000 2012
4	87	4cf2c5321d18a143951b5cec	4bf58dd8d48988d1cb941735	Food Truck	40.740104	-73.989658	-240	Tue Apr 03 18:03:00 +0000 2012

So we have the user ID, a bunch of other indecipherable IDs, the kind of venue at the given latitude and longitude, and the time of the check-in.

We are mainly interested in the userID,venueCategory, utcTimestamp, latitude and longitude columns.

But to work with date-times, we first need to convert the timestamp to a date-time object.

We convert the given column and it gives the following dataframe:



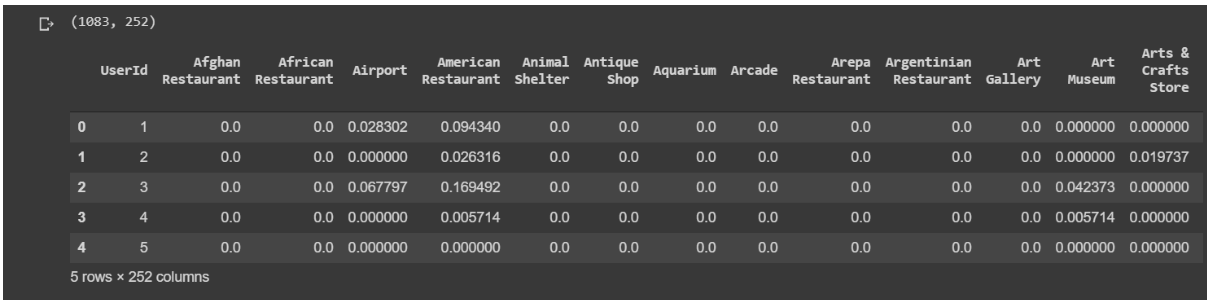
```
new_york.drop(['utcTimestamp'],axis=1,inplace=True)
new_york.head()
```

Unnamed: 0	userId	venueId	venueCategoryId	venueCategory	latitude	longitude	timezoneOffset	day_of_week	datetime
0	470	49bbd6c0f964a520f4531fe3	4bf58dd8d48988d127951735	Arts & Crafts Store	40.719810	-74.002581	-240	Tue	2012-04-03 18:00:09
1	979	4a43c0aef964a520c6a61fe3	4bf58dd8d48988d1df941735	Bridge	40.606800	-74.044170	-240	Tue	2012-04-03 18:00:25
2	69	4c5cc7b485a1e21e00d35711	4bf58dd8d48988d103941735	Home (private)	40.716162	-73.883070	-240	Tue	2012-04-03 18:02:24
3	395	4bc7086715a7ef3bef9878da	4bf58dd8d48988d104941735	Medical Center	40.745164	-73.982519	-240	Tue	2012-04-03 18:02:41
4	87	4cf2c5321d18a143951b5cec	4bf58dd8d48988d1cb941735	Food Truck	40.740104	-73.989658	-240	Tue	2012-04-03 18:03:00

Now, to convert this categorical data into a more machine-friendly format.

We have 251 unique categories of venues for the new york dataset, so we create one-hot encodings for each, group them by the 1083 unique users and average the encodings.

After the above mentioned steps, we get a dataframe that looks like this:



```
(1083, 252)
```

	UserId	Afghan Restaurant	African Restaurant	Airport	American Restaurant	Animal Shelter	Antique Shop	Aquarium	Arcade	Arepa Restaurant	Argentinian Restaurant	Art Gallery	Art Museum	Arts & Crafts Store
0	1	0.0	0.0	0.028302	0.094340	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000
1	2	0.0	0.0	0.000000	0.026316	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.019737
2	3	0.0	0.0	0.067797	0.169492	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.042373	0.000000
3	4	0.0	0.0	0.000000	0.005714	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.005714	0.000000
4	5	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000

5 rows × 252 columns

Now our data is perfectly formatted for the next step, i.e. Clustering of users.

Clustering of users on prepared data:

Using the now structured data to cluster the users based on their frequency of visits to particular locations into 10 unique clusters.

For this purpose we use the classic algorithm k-means.

k-means clustering is a method of vector quantization, originally from signal processing, that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean (cluster centers or cluster centroid), serving as a prototype of the cluster.

After clustering, we get a set of labels between 0 and 9(inclusive) for each user which we then assign to the respective users.

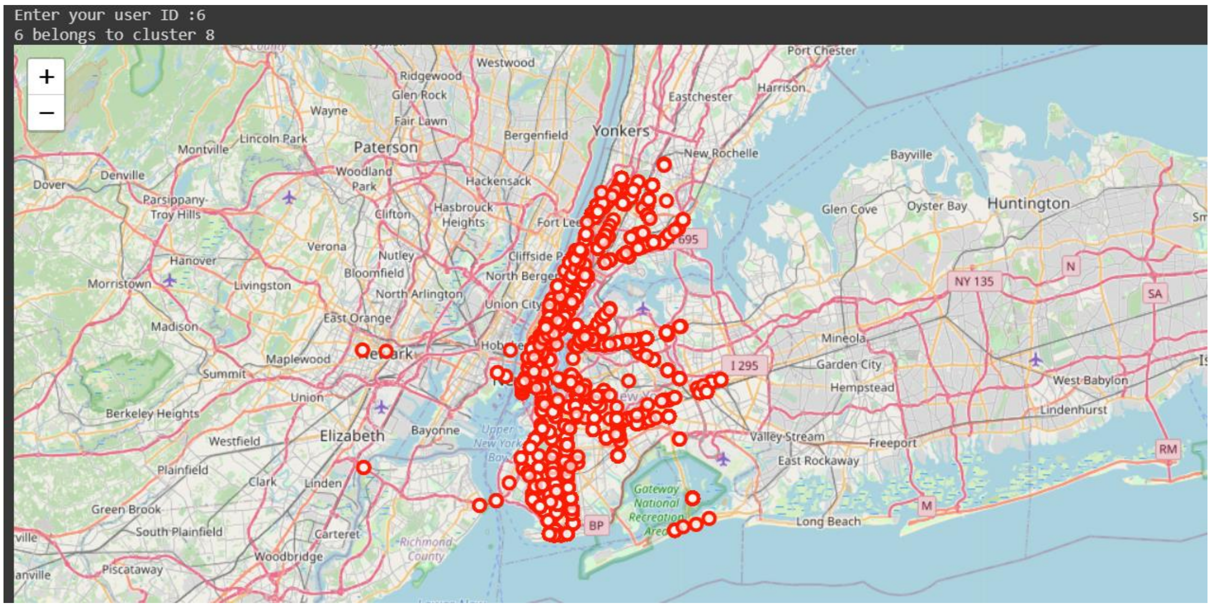
For simplicity, we also filter out the top ten visited places by the users so as to reduce non-essential features.

Now, armed with the most frequented places for each user and the clustered users, we find a pattern in each cluster and try to recommend to the users, similar places that have been frequented by others in their clusters.

The successive parts of the methodology are the results obtained, and therefore will be explained in the results section.

Results

After all the clustering and finding the most common places visited in a cluster, we filter out the places that the user has visited and map the top 10 visited places of each category by the same kind of users from the clusters, resulting in a map like this:



```
Enter your user ID :6
6 belongs to cluster 8
```

Figure 1: Suggestions for userID 6

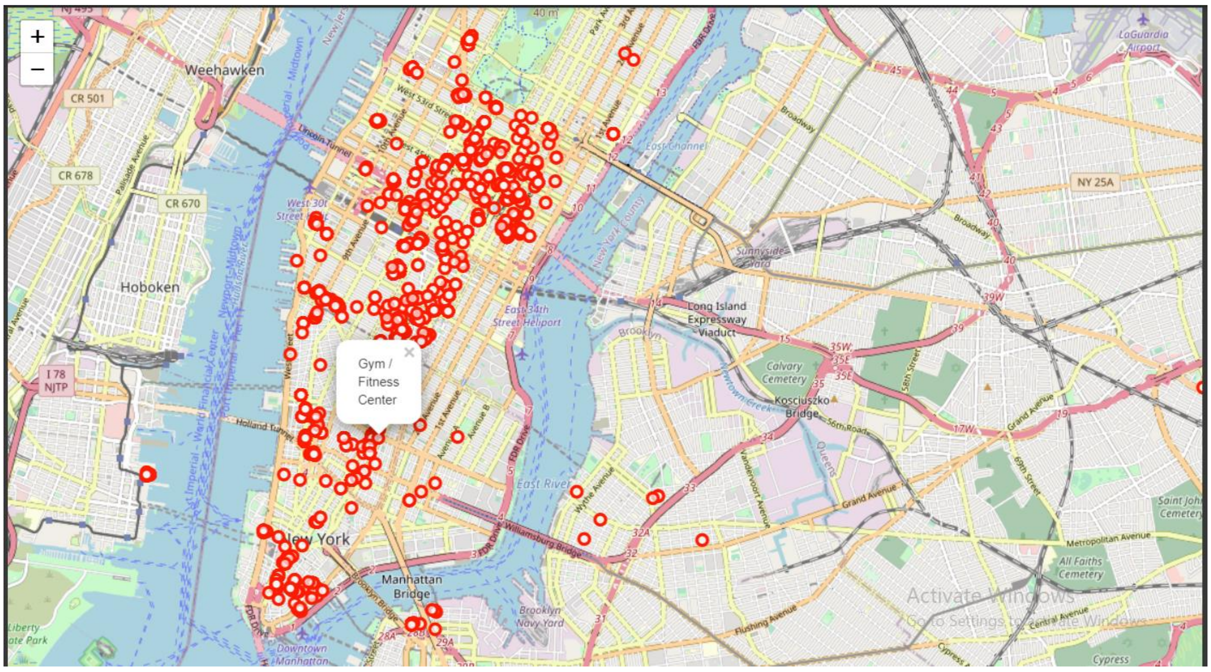


Figure 2: Recommendations for userID 395

Discussion:

As observed, the result as it is now, the recommendations seem more or less relevant to the user, as the clusters have common.

There can be a few optimizations that can be done:

- Increase the number of cluster for more customized suggestions
- Include datetime as a feature while clustering

- Have weighted recommendations by using ratings given by the users for the locations

Conclusion:

Now we have met our lover at the end of our journey. Using this concept, one can further the work done here and make more advancements in the adventure!