Applied Data Science Capstone
Project Report

The Battle of Neighborhoods
Prospects of Restaurant close to railway station in Tokyo, Japan

Dinh-Dung Le

2020.03.09

Contents

1. Introduction/Business Problem

1.1. Introduction

I used to live in Japan 3 years and really impressed by the convenience of the railway system. Tokyo especially is home to the world's busiest train stations, with the capital's rail operators handling a combined 13 billion passenger trips annually. From the station, all the needs of human life can be reach easily. Many restaurants, shopping malls are usually located near the stations.

1.2. Problem to be resolved:

In this project, I will try to use Data Science techniques to investigate about the prospects of restaurant at station in Tokyo, Japan.

1.3. Interested Audience

- I believe the methodology, tools and strategy used in this project is relevant for a person or entity considering moving to Tokyo and want to find their country's food.
- Likewise, it can be helpful approach to explore the opening of a new business. The use of FourSquare data and mapping techniques combined with data analysis will help resolve the key questions arisen.
- Lastly, this project is a good practical case for a person developing Data Science skills.

2. Data Section

2.1. Data Requirements:

The data to be used in this project come from:

- Geodata for railway stations in central Tokyo with venues established using Foursquare.
- List of metro train stations is at: https://en.wikipedia.org/wiki/List_of_Tokyo_Metro_stations

2.2. Data Sources, Data Processing and Tools used

- Tokyo data and map is to be created with use of Nominatim , Foursquare and Folium mapping.
- List of railway stations and Wards in Tokyo is collected using BeautifulSoup.
- Seaborn is also used for visualization in investigated results.

3. Methodology

3.1 Data crawling

The first step of this project is to crawl the Wikipedia website to collect information about the railway station in central Tokyo. We use the **BeautifulSoup** to collect the table on source: https://en.wikipedia.org/wiki/List_of_Tokyo_Metro_station

```
url = 'https://en.wikipedia.org/wiki/List_of_Tokyo_Metro_stations'
page = requests.get(url).text
soup = BeautifulSoup(page, 'html.parser')
tables = soup.find_all('table',{'class':'wikitable sortable'})
```

Fig 1. Crawling data using BeautifulSoup

3.2. Data preprocessing

During the data preprocessing stage, we prepare the data to be used in later steps. Since the Wikipedia page consists of two separate tables. I collected and concatenated into 1 table for later steps.

```
# Table 1
row = []
for tr in tables[0].find_all('tr'):
    if tr.find_all('th') == []:
        row.append([td.get_text(strip=True) for td in tr.find_all('td')])
column_names = ['Station', 'Line', 'Ward', 'Opening Day','Design', 'Daily ridership']
df0 = pd.DataFrame(row, columns=column_names)
# Table 2
row = []
for tr in tables[1].find_all('tr'):
    if tr.find_all('th') == []:
        row.append([td.get_text(strip=True) for td in tr.find_all('td')])
column_names = ['Station', 'Line', 'Ward', 'Opening Day','Design', 'Daily ridership']
df1 = pd.DataFrame(row, columns=column_names)

# Combine two table into one
df = pd.concat([df0, df1])
```

Fig 2. Collect the table crawled from Wikipedia.

Next, Remove the stations which appear in both table (duplicated) and remove unnecessary columns: (Line, Open day).

```
# Option: sort name of station
df.sort_values("Station", inplace = True)
# Remove dupicated stations
df.drop_duplicates(subset ="Station",
                   keep = False, inplace = True)
# Remove unused columns
df.drop(['Line','Opening Day' ], axis=1, inplace=True)
# Reset index of df from 0
df.index = np.arange(0, len(df))
df.head(10)
```

Fig 3. Remove duplicate and rows and unused columns.

The collected data is as follow:

	Station	Ward	Design	Daily ridership
0	Akabane Iwabuchi	Kita	Underground	92,093
1	Akasaka	Minato	Underground	95,556
2	Akasaka Mitsuke	Minato	Underground	127,252
3	Akihabara	Chiyoda	Underground	125,928
4	Aoyama-Itchōme	Minato	Underground	117,633
5	Asakusa	Taitō	Underground	107,628
6	Awajichō	Chiyoda	Underground	59,445
7	Ayase	Adachi	Elevated	451,413
8	Azabu-Jūban	Minato	Underground	49,467
9	Baraki-Nakayama	Funabashi(Chiba)	Elevated	27,342

Fig 4. The collected data.

There are two rows have incorrect information in table include

- Row [9] Baraki-Nakayama Funabashi(Chiba). this station seems belong to Chiba Perfecture.
- Row [51] Kokkai Gijidō-maeTameike-Sannō. In fact, this is the name of two stations: Kokkai Gijidō-mae and Tameike-Sannō.

In this project, I decided removing two rows from the table.

```
# Remove wrong station in the table index 9 and 51
df.drop(df.index[9], inplace=True)
# [51] become [50] after remove [9]
df.drop(df.index[50], inplace=True)
# Reset index from 0
df.index = np.arange(len(df))
#df
```

Fig 5. Remove wrong information rows.

The next step in data preprocessing is use Nominatium to search for the latitude and longitude of the stations. The results can be used to add latitude and longitude to the table as can be seen in the Fig 6.

	Ward	Station	Latitude	Longitude
0	Kita	Akabane Iwabuchi	35.783448	139.722100
1	Minato	Akasaka	35.671679	139.735622
2	Minato	Akasaka Mitsuke	35.676298	139.737440
3	Chiyoda	Akihabara	35.699736	139.771250
4	Minato	Aoyama-Itchōme	35.672749	139.724061
5	Taitō	Asakusa	35.709929	139.731142
6	Chiyoda	Awajichō	35.695129	139.767457
7	Adachi	Ayase	35.762188	139.825389
8	Minato	Azabu-Jūban	35.655132	139.737083
9	Nerima	Chikatetsu Akatsuka	35.769913	139.644122

Fig 6. Examples of 10 rows in the table.

To finish the data preprocessing step, we have some basic statistic results from the table .

```
Ward
Adachi                2
Arakawa               3
Bunkyō               12
Chiyoda              18
ChiyodaShinjuku       1
Chūō                 14
Edogawa               2
Funabashi(Chiba)      1
Ichikawa(Chiba)       3
Itabashi              1
Kita                  5
Kōtō                 10
Meguro                1
Minato               15
Nakano                5
Nerima                4
Shibuya               5
Shinagawa             1
Shinjuku             13
Suginami              5
Sumida                1
Taitō                 8
Toshima               6
Urayasu(Chiba)        1
Wakō(Saitama)         1
Name: Station, dtype: int64
```

Fig 7. The number of stations in each Ward of Tokyo.

Furthermore, the statistic can be seen in the map by the help of **Nominatim and Folium**

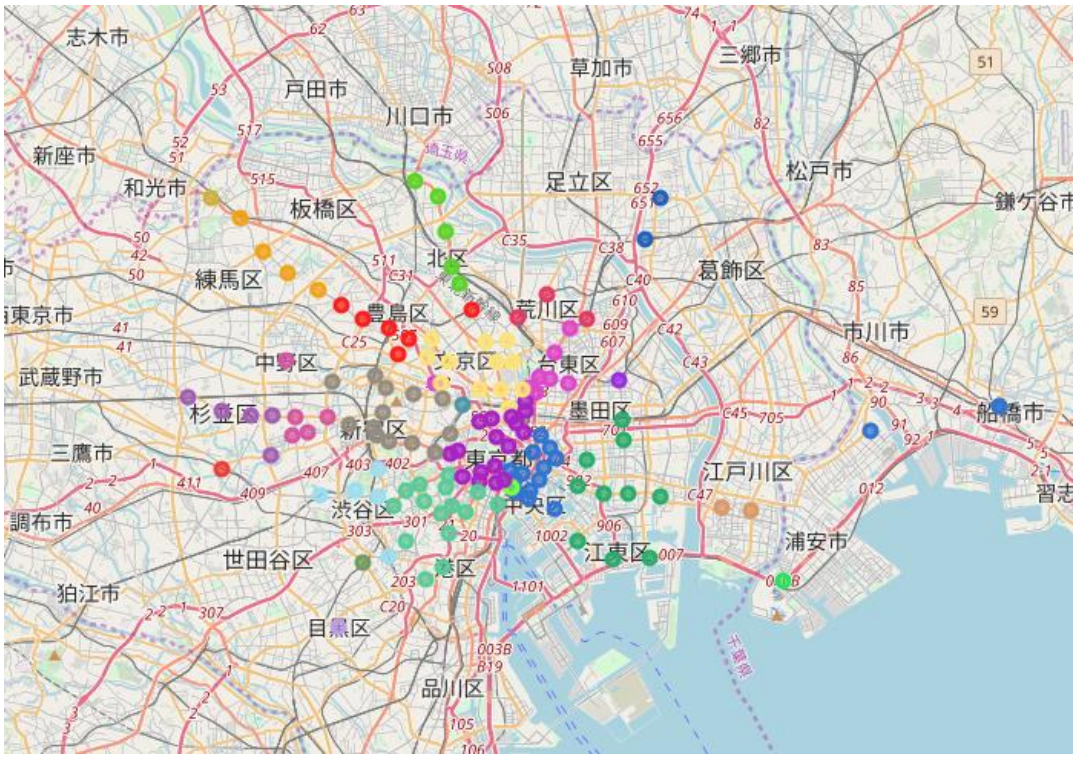

Fig 8. Visualization of stations in each ward of Tokyo.

3.3 Using Foursquare API to explore the station

Provide the personal information to access the Foursquare API.

```
CLIENT_ID = 'ID' #'your-client-ID' # your Foursquare ID
CLIENT_SECRET = 'SECRET' #'your-client-secret' # your Foursquare Secret
VERSION = '20200309'
print('My credentails:')
print('My CLIENT_ID: ' + CLIENT_ID)
print('My CLIENT_SECRET:' + CLIENT_SECRET)
```

Fig 9. Access the Foursquare API information

Because the stations are not too far from each others. The radius of 500 is used in this project. Each station is also limited at 100 venues.

```
Tokyo_Venues = getNearbyVenues(names = df['Station'], cities = df['Ward'],
                               latitudes = df['Latitude'],
                               longitudes = df['Longitude']
                              )
```

Fig 10. Get information of 100 venues nearby the stations in Tokyo.

As a results, we got 9580 venues nearby 137 stations in central Tokyo as shown in F ig.11.

```
Total number of venues in Tokyo:  9580
```

	Station	Ward	Sta_Lat	Sta_Log	Venue	Venue Latitude	Venue Longitude	Venue Category
9570	Ōtemachi	Chiyoda	35.686757	139.763616	SCEnT house den Marunouchi	35.684047	139.762416	Asian Restaurant
9571	Ōtemachi	Chiyoda	35.686757	139.763616	Crown (クラウン)	35.684525	139.761243	French Restaurant
9572	Ōtemachi	Chiyoda	35.686757	139.763616	Freshness Burger (フレッシュネスバーガー)	35.687588	139.766096	Burger Joint
9573	Ōtemachi	Chiyoda	35.686757	139.763616	Craft Beer Market	35.688275	139.765194	Beer Bar
9574	Ōtemachi	Chiyoda	35.686757	139.763616	稲荷湯	35.690098	139.766236	Bath House
9575	Ōtemachi	Chiyoda	35.686757	139.763616	Shodoten (小洞天)	35.685703	139.763902	Chinese Restaurant
9576	Ōtemachi	Chiyoda	35.686757	139.763616	Mouyan Curry (もうやんカレー)	35.686196	139.761700	Japanese Curry Restaurant
9577	Ōtemachi	Chiyoda	35.686757	139.763616	Orchestra Vino (オルケストラヴィーノ)	35.688340	139.765523	Italian Restaurant
9578	Ōtemachi	Chiyoda	35.686757	139.763616	時津洋 神田店	35.690806	139.764572	Nabe Restaurant
9579	Ōtemachi	Chiyoda	35.686757	139.763616	江戸城跡	35.682758	139.761370	Historic Site

Fig 11. The last 10 venues in total 9580 explored by using Foursquare.

We can collect some further information about the number venues vs wards and stat ions.

Ward	Station	Venue Category
Adachi	Ayase	71
	Kita-Ayase	51
Arakawa	Machiya	43
	Minami-Senju	42
	Nishi-Nippori	48
Bunkyō	Edogawabashi	58
	Gokokuji	29
	Hon-Komagome	39
	Hongō-Sanchōme	40
	Kōrakuen	100

Fig 12. Total number of venues in each station of all wards of Tokyo.

I also use **Seaborn** to visualize the above information in a easy to follow way as can be seen as follow:

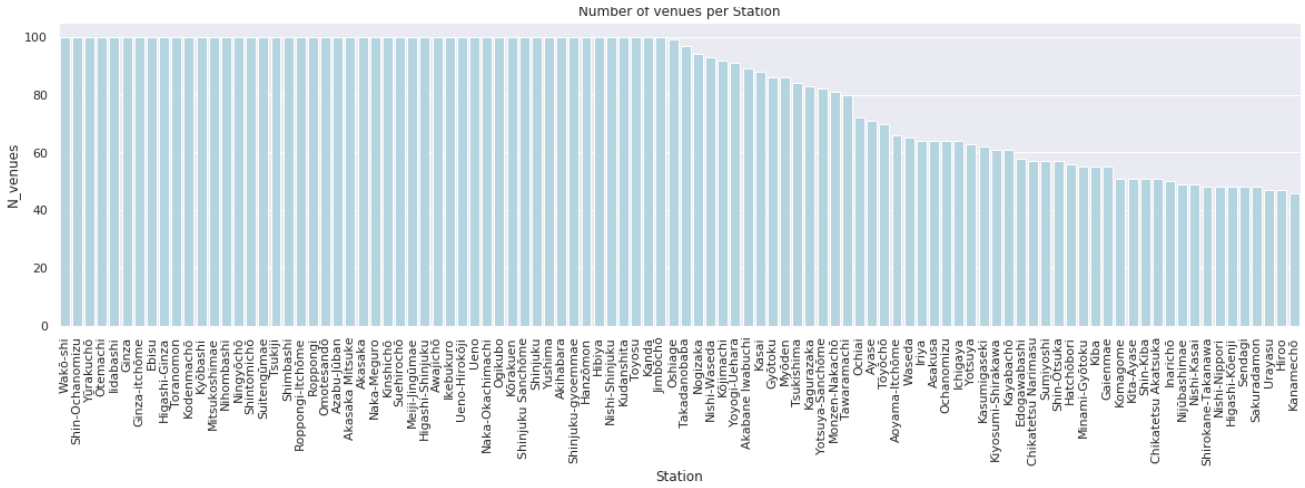

Fig 13. The visualization of number of venues per station.

3.4. Clustering the stations based on Restaurant information

In this section, I will present the implement of clustering the stations focusing on the restaurant information only.

To do this, I remove the others venues catogery and create a new dataframe.

```
# Create a Data-Frame out of it to Concentrate Only on Restaurants
Tokyo_only_restaurant = Tokyo_Venues[Tokyo_Venues['Venue Category']\
                                     .str.contains('Restaurant')].reset_index(drop=True)
Tokyo_only_restaurant.index = np.arange(1, len(Tokyo_only_restaurant)+1)
print ("Shape of the Data-Frame with Venue Category only Restaurant: ", Tokyo_only_restaurant.shape)
Tokyo_only_restaurant.tail(3)
```

Fig 14. New dataframe with only restaurants

The results of this work is as follow:

```
Shape of the Data-Frame with Venue Category only Restaurant:  (3786, 8)
```

	Station	Ward	Sta_Lat	Sta_Log	Venue	Venue Latitude	Venue Longitude	Venue Category
3784	Ōtemachi	Chiyoda	35.686757	139.763616	Mouyan Curry (もうやんカレー)	35.686196	139.761700	Japanese Curry Restaurant
3785	Ōtemachi	Chiyoda	35.686757	139.763616	Orchestra Vino (オルケストラヴィーノ)	35.688340	139.765523	Italian Restaurant
3786	Ōtemachi	Chiyoda	35.686757	139.763616	時津洋 神田店	35.690806	139.764572	Nabe Restaurant

Fig 15. Number of Restaurant explored by Foursquare API.

Some similar results are also collected.

	Ward	Station	N_venues
0	Adachi	Ayase	22
1	Adachi	Kita-Ayase	10
2	Arakawa	Machiya	13
3	Arakawa	Minami-Senju	12
4	Arakawa	Nishi-Nippori	21
5	Bunkyō	Edogawabashi	27
6	Bunkyō	Gokokuji	11
7	Bunkyō	Hon-Komagome	15
8	Bunkyō	Hongō-Sanchōme	16
9	Bunkyō	Kōrakuen	23

Fig 16. Total number of restaurants per stations

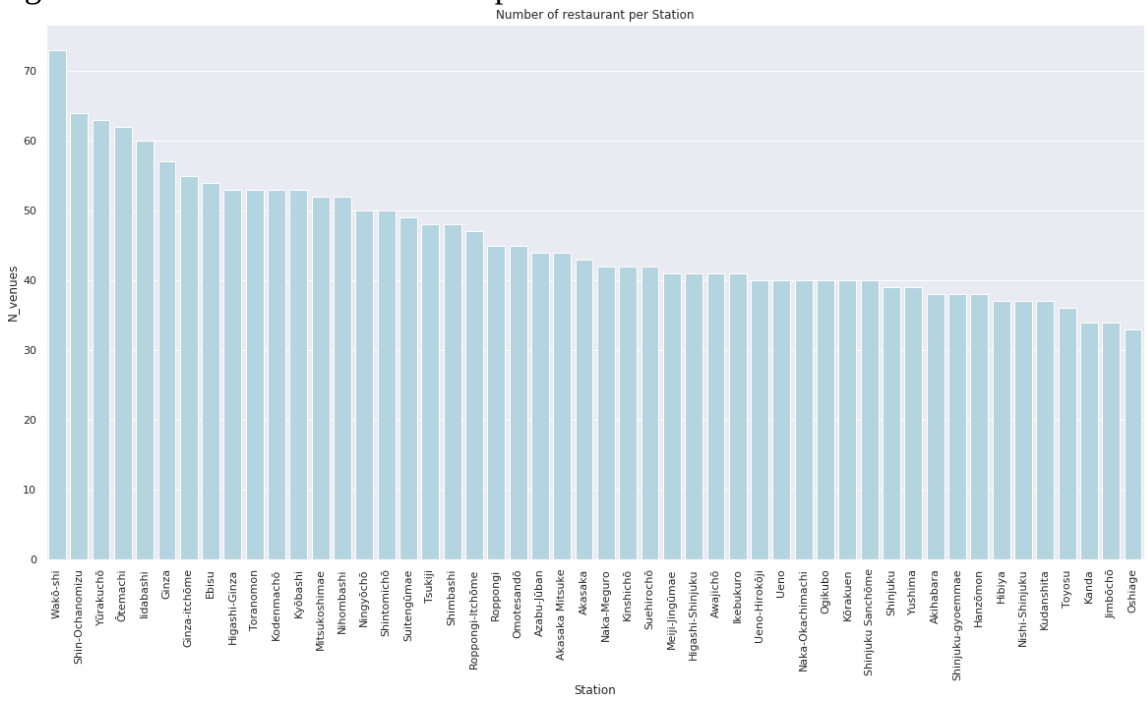

Fig 17. The visualization of restaurants per stations

- To clustering the stations, I first do the one-hot encoding on the new datafram e.

```
#One hot encoding
venues_onehot = pd.get_dummies(Tokyo_only_restaurant[['Venue Category']], prefix="", prefix_sep="")
```

- Add back to the dataframe

```
# Add the stations column back to the dataframe
venues_onehot['Station'] = Tokyo_only_restaurant['Station']
venues_onehot.head(3)
```

- Average the values

```
# Average per station
venues_grouped = venues_onehot.groupby(['Station']).mean().reset_index()
venues_grouped.sample(15)
```

The result is as follow:

	Station	American Restaurant	Argentinian Restaurant	Asian Restaurant	Australian Restaurant	Bangladeshi Restaurant	Beijing Restaurant	Belarusian Restaurant	Belgian Restaurant	Brazilian Restaurant	Burmese Restaurant	Cantonese Restaurant	Chinese Restaurant	Comfort Food Restaurant	Czech Restaurant	Dim Sum Restaurant	Donburi Restaurant	Dongbei Restaurant	Dumpling Restaurant	Eastern European Restaurant	Falafel Restaurant	Fast Food Restaurant
75	Nezu	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0	0.000000	0.000000	0.0	0.0	0.000000	0.0	0.0	0.0	0.000000	0.0	0.000000	0.0	0.0	0.000000
100	Shin-Nakano	0.000000	0.0	0.052632	0.0	0.0	0.0	0.0	0.000000	0.000000	0.0	0.0	0.105263	0.0	0.0	0.0	0.052632	0.0	0.000000	0.0	0.0	0.000000
131	Yushima	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0	0.000000	0.000000	0.0	0.0	0.050000	0.0	0.0	0.0	0.050000	0.0	0.000000	0.0	0.0	0.000000
81	Nishi-Nippori	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0	0.000000	0.000000	0.0	0.0	0.000000	0.0	0.0	0.0	0.095238	0.0	0.047619	0.0	0.0	0.047619
59	Minami-Asagaya	0.000000	0.0	0.100000	0.0	0.0	0.0	0.0	0.000000	0.000000	0.0	0.0	0.100000	0.0	0.0	0.0	0.000000	0.0	0.100000	0.0	0.0	0.000000
115	Toranomon	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0	0.015873	0.000000	0.0	0.0	0.079365	0.0	0.0	0.0	0.000000	0.0	0.000000	0.0	0.0	0.000000
113	Takebashi	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0	0.000000	0.000000	0.0	0.0	0.181818	0.0	0.0	0.0	0.000000	0.0	0.000000	0.0	0.0	0.000000
73	Nakano-Fujimichō	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0	0.000000	0.000000	0.0	0.0	0.111111	0.0	0.0	0.0	0.111111	0.0	0.000000	0.0	0.0	0.111111
104	Shinjuku Sanchōme	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0	0.000000	0.035714	0.0	0.0	0.035714	0.0	0.0	0.0	0.035714	0.0	0.000000	0.0	0.0	0.000000
37	Kagurazaka	0.000000	0.0	0.030303	0.0	0.0	0.0	0.0	0.000000	0.000000	0.0	0.0	0.181818	0.0	0.0	0.0	0.000000	0.0	0.030303	0.0	0.0	0.000000
129	Yoyogi-Kōen	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0	0.000000	0.000000	0.0	0.0	0.000000	0.0	0.0	0.0	0.000000	0.0	0.000000	0.0	0.0	0.000000
122	Ueno	0.027778	0.0	0.000000	0.0	0.0	0.0	0.0	0.000000	0.000000	0.0	0.0	0.111111	0.0	0.0	0.0	0.027778	0.0	0.000000	0.0	0.0	0.000000
108	Shirokanedai	0.076923	0.0	0.000000	0.0	0.0	0.0	0.0	0.000000	0.000000	0.0	0.0	0.153846	0.0	0.0	0.0	0.000000	0.0	0.000000	0.0	0.0	0.000000
111	Sumiyoshi	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0	0.000000	0.000000	0.0	0.0	0.000000	0.0	0.0	0.0	0.076923	0.0	0.000000	0.0	0.0	0.000000
83	Nishi-Waseda	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0	0.000000	0.000000	0.0	0.0	0.095238	0.0	0.0	0.0	0.023810	0.0	0.000000	0.0	0.0	0.023810

Fig 18. The dataframe with average values.

From the received dataframe, I calculate the top common type of restaurant.

	Station	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue	9th Most Common Venue	10th Most Common Venue
0	Akabane Iwabuchi	Ramen Restaurant	Italian Restaurant	Japanese Restaurant	Japanese Curry Restaurant	Kushikatsu Restaurant	Donburi Restaurant	Chinese Restaurant	Indian Restaurant	Persian Restaurant	Restaurant
1	Akasaka	Japanese Restaurant	Chinese Restaurant	Kaiseki Restaurant	Ramen Restaurant	Italian Restaurant	Szechuan Restaurant	Nabe Restaurant	Korean Restaurant	Sushi Restaurant	Tonkatsu Restaurant
2	Akasaka Mitsuke	Japanese Restaurant	Szechuan Restaurant	Ramen Restaurant	Chinese Restaurant	Italian Restaurant	Soba Restaurant	Sushi Restaurant	Seafood Restaurant	American Restaurant	Unagi Restaurant
3	Akihabara	Ramen Restaurant	Donburi Restaurant	Tonkatsu Restaurant	Sushi Restaurant	Chinese Restaurant	Szechuan Restaurant	Soba Restaurant	Brazilian Restaurant	Kebab Restaurant	Yoshoku Restaurant
4	Aoyama-Itchōme	Italian Restaurant	Ramen Restaurant	Japanese Restaurant	Chinese Restaurant	Soba Restaurant	Sushi Restaurant	Japanese Curry Restaurant	Tonkatsu Restaurant	Thai Restaurant	Udon Restaurant
5	Asakusa	Ramen Restaurant	Chinese Restaurant	Soba Restaurant	Japanese Curry Restaurant	Sushi Restaurant	Udon Restaurant	Kushikatsu Restaurant	Tonkatsu Restaurant	Yoshoku Restaurant	Indian Restaurant
6	Awajichō	Ramen Restaurant	Japanese Curry Restaurant	Soba Restaurant	Tonkatsu Restaurant	Japanese Restaurant	Chinese Restaurant	Indian Restaurant	Yoshoku Restaurant	Unagi Restaurant	Nabe Restaurant
7	Ayase	Ramen Restaurant	Donburi Restaurant	Chinese Restaurant	Japanese Restaurant	Tonkatsu Restaurant	Restaurant	Dumpling Restaurant	Indian Restaurant	Sushi Restaurant	Italian Restaurant

Fig 19. Top 10 common type of restaurant among stations in Tokyo

Finally, I use the combined data frame to feed into the k-means clustering algorithm with the number of k = 3.

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=3, n_init=15, n_jobs=None, precompute_distances='auto',
    random_state=None, tol=0.0001, verbose=0)
```

Fig 20. K-means clustering (k = 3)

4. Results summary

By using K-means clustering algorthim, 137 stations in central Tokyo is separated into 3 cluster with the quantity of each cluster is as follow:

```
Cluster
0    71
1    11
2    55
Name: Station, dtype: int64
```

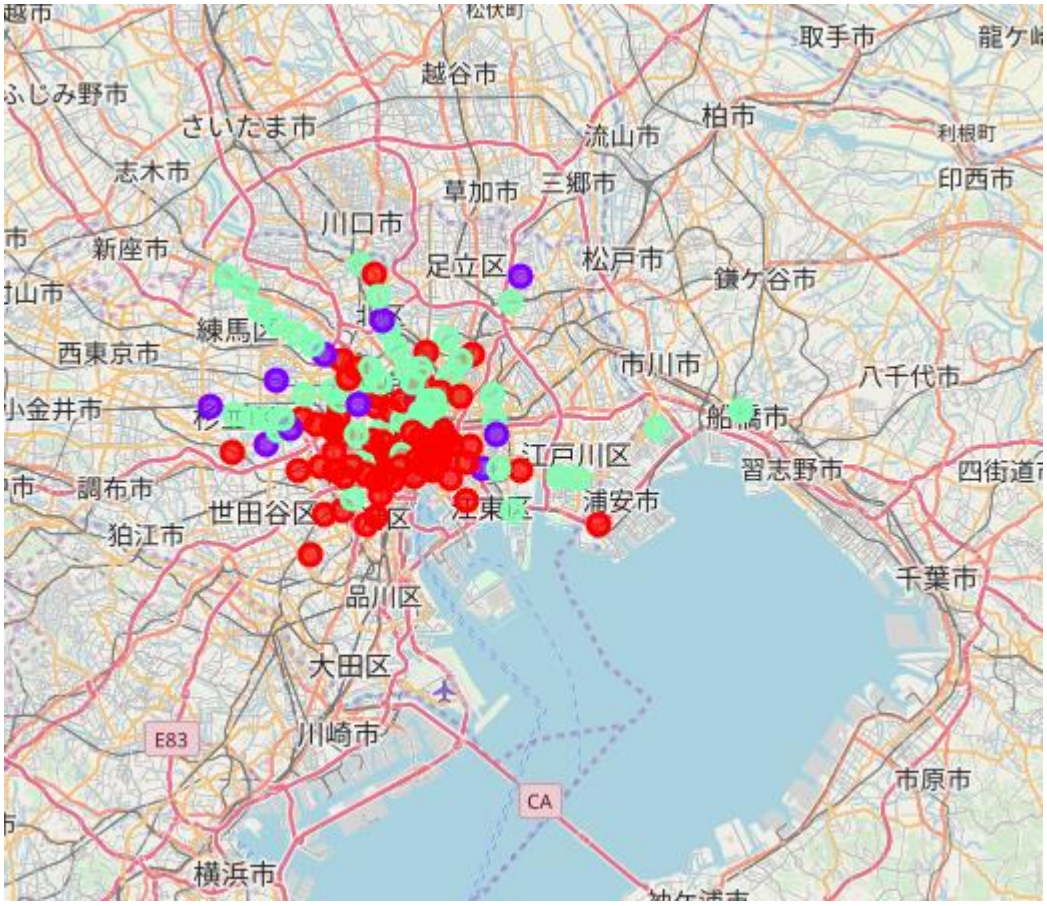

Fig 21. Clusters visualization on Tokyo map

The results of the exploratory data analysis and clustering are summarized follow:

- It is clear that Japanese restaurant is popular in all stations.
- We can found some foreign restaurant such as Chinese restaurants, Italian restaurants, French restaurant in the stations of Cluster 1. Cluster 1 consists of main stations in the central of Tokyo.
- The Cluster 2 consists of medium stations in Tokyo. Some foreign restaurant located here.
- The Cluster 3 consists of small stations and outside Tokyo.

5. Discussion and Conclusion

5.1 Discussion

According to this analysis, cluster 1 station will provide most competition for an upcoming restaurant.

Some drawbacks of this analysis are the clustering is completely based on the most common venues obtained from Foursquare data. Since land price, number of potential customers, benefits and drawbacks of station with type underground or on ground, could all play a major role and thus, this analysis is definitely far from being conclusory.

5.2. Conclusion

Finally, to conclude this project, we have got a small glimpse of how real life data-science projects look like. I have made use of some frequently used python libraries to scrap web-data, use Foursquare API to explore the stations of Tokyo and saw the results of segmentation of stations using Folium leaflet map. Potential for this kind of analysis in a real life business problem is discussed in great detail. Also, some of the drawbacks and chance for improvements to represent even more realistic pictures are mentioned.