July 22, 2021

NEIGHBORHOOD EXPLORATION (KOBE CITY)

1. INTRODUTION

In this assignment I would like to use this opportunity to introduce Kobe city in Japan and its surrounding by adopting knowledge that I gain through this data science certification course. I am a foreign resident here who has been living here in Japan for more than 15years. Kobe City, while it is not the most famous city in Japan, but it has many attractions. I have many relative and friend who are making plan to visit Japan but not quite where to visit if they want to explore Kobe City. I hope this exploration of mine hopefully will help them to gain interest in visiting Kobe City in the future.

First, geocoder will be used to convert the city name into their equivalent latitude and longitude values. Then, using the Foursquare API, I will bring you to explore neighborhoods in Kobe City. I will use the **explore** function to get the most common venue categories in each neighborhood, and then use this feature to group the neighborhoods into clusters. I will use the _k_-means clustering algorithm to complete this task. Finally, I will use the Folium library to visualize the neighborhoods in Kobe City and their emerging clusters.

2. TABLE OF CONTENTS

3. DOWNLOAD AND EXPLORE DATASET

Before we get the data and start exploring it, all libraries required for processing the data is downloaded. These libraries need to be used for the exploration. Libraries used are as follow:

```
: import numpy as np # library to handle data in a vectorized manner

  import pandas as pd # library for data analsysis
  pd.set_option('display.max_columns', None)
  pd.set_option('display.max_rows', None)

  import json # library to handle JSON files

  !conda install -c conda-forge geopy --yes # uncomment this line if you haven't complet
  from geopy.geocoders import Nominatim # convert an address into latitude and longitude

  import requests # library to handle requests
  from pandas.io.json import json_normalize # tranform JSON file into a pandas datafram

  # Matplotlib and associated plotting modules
  import matplotlib.cm as cm
  import matplotlib.colors as colors

  # import k-means from clustering stage
  from sklearn.cluster import KMeans

  !conda install -c conda-forge folium=0.5.0 --yes # uncomment this line if you haven't
  import folium # map rendering library

  print('Libraries imported.')
```

Next, we will start to download the data set. While I was struggling to find a list of location data in Japan, I came across a list that is available to be downloaded from Japan Post homepage. Data of location with zip code list is provided in csv format. The data came with zip code (post code), Prefecture or normally called state in other country, city, ward and town. It is a list for all location but I am only interested to explore Kobe City which located in Hyogo Prefecture.

Since I am using notebook which been provided by IBM studio, the data is uploaded as one of an asset to the project created specifically for this assignment.

Top 5 data are as shown below:

	PostCode	Prefecture	City	Town
0	600000	HOKKAIDO	SAPPORO SHI CHUO KU	IKANIKEISAIGANAIBAAI
1	640941	HOKKAIDO	SAPPORO SHI CHUO KU	ASAHIGAOKA
2	600041	HOKKAIDO	SAPPORO SHI CHUO KU	ODORIHIGASHI
3	600042	HOKKAIDO	SAPPORO SHI CHUO KU	ODORINISHI(1-19-CHOME)
4	640820	HOKKAIDO	SAPPORO SHI CHUO KU	ODORINISHI(20-28-CHOME)

The data frame come with postcode, prefecture name or state name which typically in other country, city and town. This data is actually a data of whole Japan but for this

assignment, exploration is only made for Kobe city, so I will slice the data to include only data with prefecture Hyogo. Kobe city is actually under Hyogo prefecture.

As shown above, there is no latitude and longitude data provided which are a crucial information of this exploration. Here, I will use geopy library to retrieve the required latitude and longitude data by using the prefecture, city and town information. First prefecture, city and town will be combined and referred as address. Then using geopy library, latitude and longitude information will be retrieved via the address information.

Code:

```
from geopy.exc import GeocoderTimedOut
from geopy.geocoders import Nominatim

#declare an empty list to store latitude and longitude of values of city column
latitude = []
longitude = []

#Function to find the coordinate of a given city
def findGeocode(city):
    #try and catch to overcome the exception thrown by geolocator using geocodertimedout
    try:
        #specify the user-agent as your app name it should not be none
        geolocator = Nominatim(user_agent="jp-explorer")
        return geolocator.geocode(city)
    except GeocoderTimedOut:
        return findGeocode(city)

for i in df_kobe["Address"]:
    if findGeocode(i) != None:
        loc = findGeocode(i)
        #coordinate return from function is stored into two separate list
        latitude.append(loc.latitude)
        longitude.append(loc.longitude)

    #if coordinate for a city not found, insert "NaN" indicating missing value
    else:
        latitude.append(np.nan)
        longitude.append(np.nan)

#Adding latitude and longitude values to dataframe
df_kobe["Latitude"] = latitude
df_kobe["Longitude"] = longitude

df_kobe.head()
```

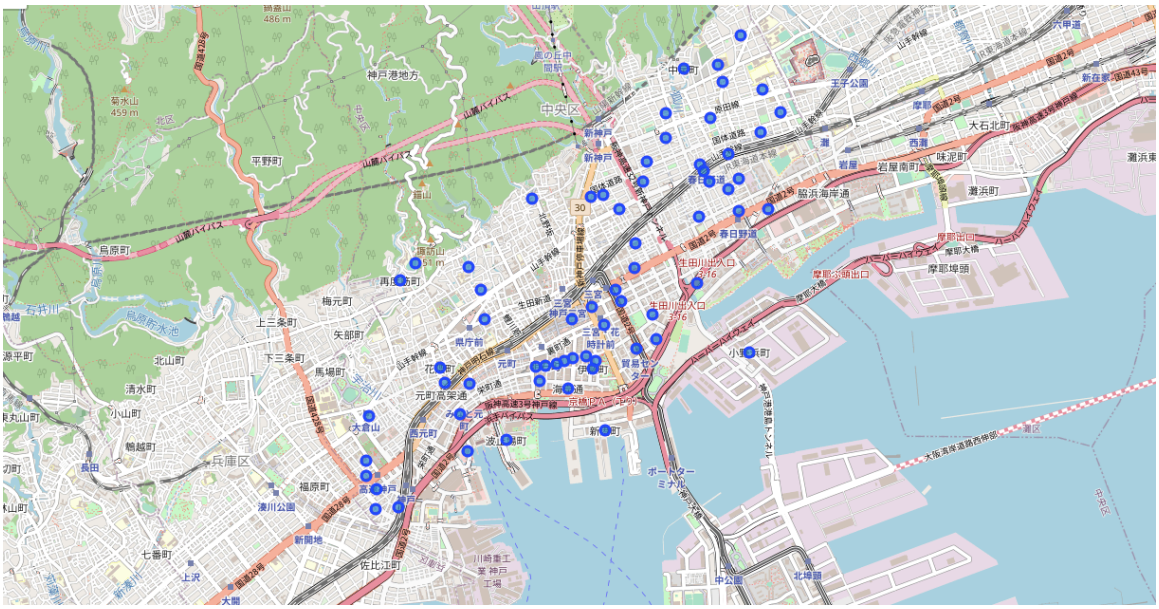
Result:

	PostCode	Prefecture	City	Town	Address	Latitude	Longitude
89093	6500000	HYOGO KEN	KOBE SHI CHUO KU	IKANIKEISAIGANAIBAAI	HYOGO KEN, KOBE SHI CHUO KU, IKANIKEISAIGANAIBAAI	NaN	NaN
89094	6500025	HYOGO KEN	KOBE SHI CHUO KU	AIOICHO	HYOGO KEN, KOBE SHI CHUO KU, AIOICHO	34.677720	135.177548
89095	6500037	HYOGO KEN	KOBE SHI CHUO KU	AKASHIMACHI	HYOGO KEN, KOBE SHI CHUO KU, AKASHIMACHI	34.688200	135.190912
89096	6510095	HYOGO KEN	KOBE SHI CHUO KU	ASAHIDORI	HYOGO KEN, KOBE SHI CHUO KU, ASAHIDORI	34.697306	135.199023
89097	6510076	HYOGO KEN	KOBE SHI CHUO KU	AZUMADORI	HYOGO KEN, KOBE SHI CHUO KU, AZUMADORI	34.699679	135.208416

Column which are shown as NaN then removed as they will cause on later mapping process.

	PostCode	Prefecture	City	Town	Address	Latitude	Longitude
89094	6500025	HYOGO KEN	KOBE SHI CHUO KU	AIOICHO	HYOGO KEN, KOBE SHI CHUO KU, AIOICHO	34.677720	135.177548
89095	6500037	HYOGO KEN	KOBE SHI CHUO KU	AKASHIMACHI	HYOGO KEN, KOBE SHI CHUO KU, AKASHIMACHI	34.688200	135.190912
89096	6510095	HYOGO KEN	KOBE SHI CHUO KU	ASAHIDORI	HYOGO KEN, KOBE SHI CHUO KU, ASAHIDORI	34.697306	135.199023
89097	6510076	HYOGO KEN	KOBE SHI CHUO KU	AZUMADORI	HYOGO KEN, KOBE SHI CHUO KU, AZUMADORI	34.699679	135.208416
89098	6510092	HYOGO KEN	KOBE SHI CHUO KU	IKUTACHO	HYOGO KEN, KOBE SHI CHUO KU, IKUTACHO	34.700937	135.196116

Then, using these data map of Kobe city with neighborhoods superimposed on top is created:



4. EXPLORING NEIGHBORHOODS OF KOBE CITY

To explore the neighborhood, function to get nearby venue is created:

```
def getNearbyVenues(names, latitudes, longitudes, radius=500):

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

        # create the API request URL
        url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={},{}&radius={}&limit={}'.format(
            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 = ['Neighborhood',
                  'Neighborhood Latitude',
                  'Neighborhood Longitude',
                  'Venue',
                  'Venue Latitude',
                  'Venue Longitude',
                  'Venue Category']
```

Function is then run and result is as per below:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	AIOICHO	34.67772	135.177548	神戸ハーバーランド温泉 万葉倶楽部	34.679449	135.180736	Hot Spring
1	AIOICHO	34.67772	135.177548	Kōbe Station (神戸駅)	34.679137	135.178284	Train Station
2	AIOICHO	34.67772	135.177548	OS Cinemas (OSシネマズ 神戸ハーバーランド)	34.679604	135.182504	Multiplex
3	AIOICHO	34.67772	135.177548	Starbucks	34.680117	135.178634	Coffee Shop
4	AIOICHO	34.67772	135.177548	中畑商店	34.674219	135.179923	BBQ Joint

5. ANALYZE EACH NEIGHBORHOODS

Each neighborhood is then analyzed further as per below:

```
# one hot encoding
kobe_onehot = pd.get_dummies(kobe_venues[['Venue Category']], prefix="", prefix_sep="")

# add neighborhood column back to dataframe
kobe_onehot['Neighborhood'] = kobe_venues['Neighborhood']

# move neighborhood column to the first column
fixed_columns = [kobe_onehot.columns[-1]] + list(kobe_onehot.columns[:-1])
kobe_onehot = kobe_onehot[fixed_columns]

kobe_onehot.head()
```

]:

	Neighborhood	Airport	Airport Lounge	Airport Service	Airport Terminal	American Restaurant	Arcade	Art Gallery	Art Museum	Arts & Crafts Store	Auto Dealership	Automotive Shop	BBQ Joint	Bagel Shop	Bakery	Bar	Bath House	Bavarian Restaurant	Bay	Bed & Breakfast	Beer Garden
0	AIOICHO	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	AIOICHO	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	AIOICHO	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	AIOICHO	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	AIOICHO	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

And let's examine the new dataframe size.

```
kobe_onehot.shape
```

]: (3484, 185)

Next, let's group rows by town and by taking the mean of the frequency of occurrence of each category

```
kobe_grouped = kobe_onehot.groupby('Neighborhood').mean().reset_index()
kobe_grouped
```

]:

	Neighborhood	Airport	Airport Lounge	Airport Service	Airport Terminal	American Restaurant	Arcade	Art Gallery	Art Museum	Arts & Crafts Store	De
0	AIOICHO	0.000000	0.000000	0.0	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	
1	AKASHIMACHI	0.000000	0.000000	0.0	0.0	0.000000	0.000000	0.000000	0.000000	0.010000	
2	ASAHIDORI	0.000000	0.000000	0.0	0.0	0.000000	0.011111	0.000000	0.000000	0.000000	
3	AZUMADORI	0.000000	0.000000	0.0	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	
4	BENTENCHO	0.000000	0.000000	0.0	0.0	0.012987	0.000000	0.000000	0.000000	0.000000	
5	DAINICHIDORI	0.000000	0.000000	0.0	0.0	0.000000	0.000000	0.000000	0.066667	0.000000	
6	EDOMACHI	0.000000	0.000000	0.0	0.0	0.000000	0.000000	0.000000	0.000000	0.010638	
7	FUKIAICHO	0.000000	0.000000	0.0	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	
8	FUTATABISUJICHO	0.000000	0.000000	0.0	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	

Let's print each neighborhood along with the top 5 most common venues

:
```
num_top_venues = 5

for hood in kobe_grouped['Neighborhood']:
    print("----"+hood+"----")
    temp = kobe_grouped[kobe_grouped['Neighborhood'] == hood].T.reset_index()
    temp.columns = ['venue','freq']
    temp = temp.iloc[1:]
    temp['freq'] = temp['freq'].astype(float)
    temp = temp.round({'freq': 2})
    print(temp.sort_values('freq', ascending=False).reset_index(drop=True).head(num_top_venues))
    print('\n')
```

```
----AIOICHO----
                venue  freq
0  Convenience Store  0.22
1              Café  0.07
2      Shopping Mall  0.07
3        Coffee Shop  0.04
4   Ramen Restaurant  0.04


----AKASHIMACHI----
                 venue  freq
0                Café  0.16
1   Chinese Restaurant  0.06
2  Dumpling Restaurant  0.05
3   Donburi Restaurant  0.04
4              Bakery  0.04
```

The data is then stored in a dataframe:

	Neighborhood	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 I Com V
0	AIOICHO	Convenience Store	Shopping Mall	Café	Ramen Restaurant	Coffee Shop	Clothing Store	Seafood Restaurant	Bakery	Superm
1	AKASHIMACHI	Café	Chinese Restaurant	Dumpling Restaurant	Yoshoku Restaurant	Donburi Restaurant	Clothing Store	Bakery	Burger Joint	Steakh
2	ASAHIDORI	Convenience Store	Coffee Shop	Rock Club	Hotel	Donburi Restaurant	Chinese Restaurant	Café	Japanese Restaurant	Japa (Resta
3	AZUMADORI	Convenience Store	Shopping Mall	Sushi Restaurant	Train Station	Chinese Restaurant	Grocery Store	History Museum	Donburi Restaurant	Mult
4	BENTENCHO	Hotel	Dessert Shop	Italian Restaurant	Japanese Restaurant	Café	Shopping Mall	Steakhouse	Coffee Shop	Monum Land

6. CLUSTER NEIGHBORHOODS

Here unsupervised machine learning K-mean method is used to automatically categorized each of the nearby venue retrieve in the neighborhoods. Then cluster map is created to visualize the cluster location.

```
# set number of clusters
kclusters = 5

kobe_grouped_clustering = kobe_grouped.drop('Neighborhood', 1)

# run k-means clustering
kmeans = KMeans(n_clusters=kclusters, random_state=0).fit(kobe_grouped_clustering)

# check cluster labels generated for each row in the dataframe
kmeans.labels_[0:10]
```

```
array([0, 1, 3, 0, 1, 2, 1, 1, 3, 1], dtype=int32)
```

Let's create a new dataframe that includes the cluster as well as the top 10 venues for each neighborhood.

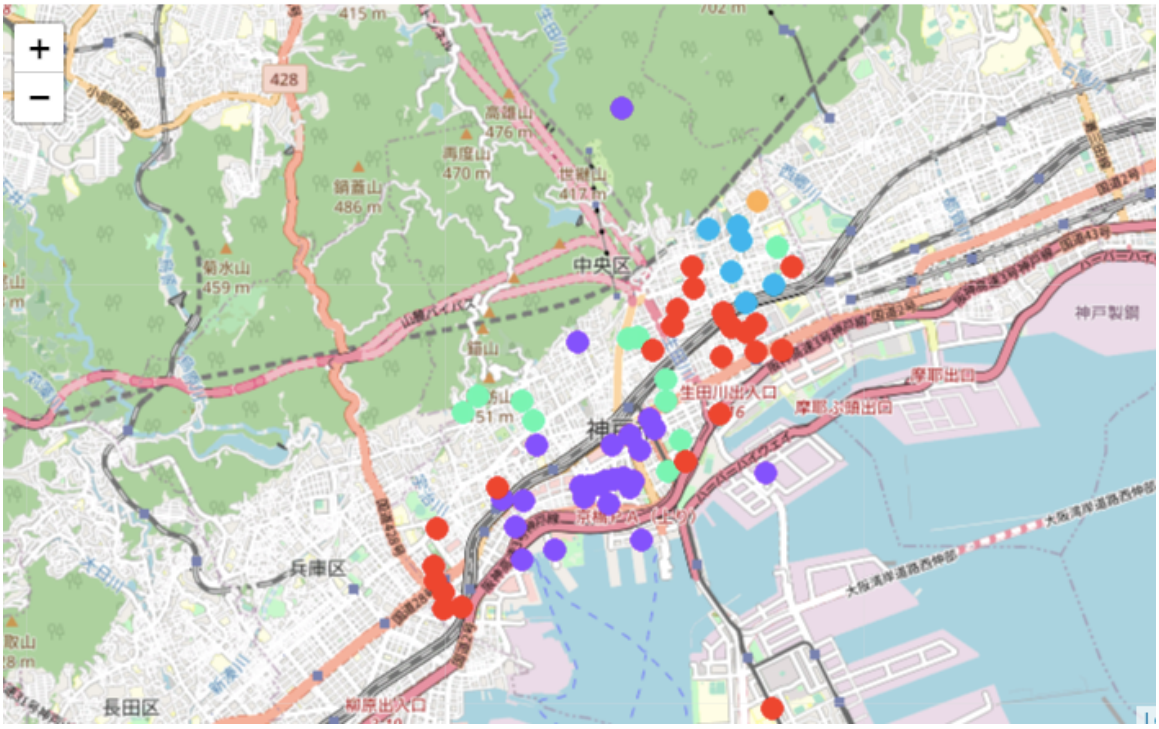
```
# add clustering labels
#towns_venues_sorted.insert(0, 'Cluster Labels', kmeans.labels_)

kobe_merged = kobe_venues

# merge kobe_grouped with kobe_data to add latitude/longitude for each neighborhood
kobe_merged = kobe_merged.join(towns_venues_sorted.set_index('Neighborhood'), on='Neighborhood')

kobe_merged.head() # check the last columns!
```

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category	Cluster Labels	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 M Comn Ve
0	AIOICHO	34.67772	135.177548	神戸ハーバーランド温泉 万葉倶楽部	34.679449	135.180736	Hot Spring	0	Convenience Store	Shopping Mall	Café	Ramen Restaurant	Coffee Shop	Clothing Store	Seafood Restaurant	Bakery	Superma
1	AIOICHO	34.67772	135.177548	Kōbe Station	34.679137	135.178284	Train Station	0	Convenience Store	Shopping Mall	Café	Ramen Restaurant	Coffee Shop	Clothing Store	Seafood Restaurant	Bakery	Superma



7. EXAMINE CLUSTER

Cluster 1

```
kobe_merged.loc[kobe_merged['Cluster Labels'] == 0, kobe_merged.columns[[1] + list(ran
```

49	34.677720	135.176873	Convenience Store	0	Convenience Store	Shopping Mall	Café	Ramen Restaurant	
50	34.677720	135.182502	Food Court	0	Convenience Store	Shopping Mall	Café	Ramen Restaurant	
51	34.677720	135.175740	Udon Restaurant	0	Convenience Store	Shopping Mall	Café	Ramen Restaurant	
52	34.677720	135.178603	Convenience Store	0	Convenience Store	Shopping Mall	Café	Ramen Restaurant	
53	34.677720	135.182796	Convenience Store	0	Convenience Store	Shopping Mall	Café	Ramen Restaurant	
54	34.677720	135.178256	Grocery Store	0	Convenience Store	Shopping Mall	Café	Ramen Restaurant	
55	34.677720	135.179871	Convenience Store	0	Convenience Store	Shopping Mall	Café	Ramen Restaurant	
56	34.677720	135.175503	Convenience Store	0	Convenience Store	Shopping Mall	Café	Ramen Restaurant	
57	34.677720	135.178389	Donut Shop	0	Convenience	Shopping	Café	Ramen	

Cluster 2

```
kobe_merged.loc[kobe_merged['Cluster Labels'] == 1, kobe_merged.columns[[1] + list(range
```

]:

	Neighborhood Latitude	Venue Longitude	Venue Category	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th M Comr Ve
92	34.688200	135.192963	Hotel	1	Café	Chinese Restaurant	Dumpling Restaurant	Yoshoku Restaurant	Don Restau
93	34.688200	135.192991	Italian Restaurant	1	Café	Chinese Restaurant	Dumpling Restaurant	Yoshoku Restaurant	Don Restau
94	34.688200	135.189119	Chinese Restaurant	1	Café	Chinese Restaurant	Dumpling Restaurant	Yoshoku Restaurant	Don Restau
95	34.688200	135.191020	Café	1	Café	Chinese Restaurant	Dumpling Restaurant	Yoshoku Restaurant	Don Restau
96	34.688200	135.192220	Movie Theater	1	Café	Chinese Restaurant	Dumpling Restaurant	Yoshoku Restaurant	Don Restau
97	34.688200	135.190484	Bakery	1	Café	Chinese Restaurant	Dumpling Restaurant	Yoshoku Restaurant	Don Restau
98	34.688200	135.190166	Creperie	1	Café	Chinese	Dumpling	Yoshoku	Don

Cluster 3

```
kobe_merged.loc[kobe_merged['Cluster Labels'] == 2, kobe_merged.columns[[1] + list(range
```

]:

	Neighborhood Latitude	Venue Longitude	Venue Category	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	
857	34.710543	135.206369	Convenience Store	2	Convenience Store	Bakery	Bus Stop	Chinese Restaurant	Grocery Store	R
858	34.710543	135.202391	Convenience Store	2	Convenience Store	Bakery	Bus Stop	Chinese Restaurant	Grocery Store	R
859	34.710543	135.208535	Soba Restaurant	2	Convenience Store	Bakery	Bus Stop	Chinese Restaurant	Grocery Store	R
860	34.710543	135.206052	Bakery	2	Convenience Store	Bakery	Bus Stop	Chinese Restaurant	Grocery Store	R
861	34.710543	135.208513	Bus Stop	2	Convenience Store	Bakery	Bus Stop	Chinese Restaurant	Grocery Store	R

Cluster 4

```
kobe_merged.loc[kobe_merged['Cluster Labels'] == 3, kobe_merged.columns[[1] + list(range
```

2]:

	Neighborhood Latitude	Venue Longitude	Venue Category	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th I Com V
192	34.697306	135.197714	Rock Club	3	Convenience Store	Coffee Shop	Rock Club	Hotel	Do Resta
193	34.697306	135.197296	Seafood Restaurant	3	Convenience Store	Coffee Shop	Rock Club	Hotel	Do Resta
194	34.697306	135.197496	Donburi Restaurant	3	Convenience Store	Coffee Shop	Rock Club	Hotel	Do Resta
195	34.697306	135.195769	Donburi Restaurant	3	Convenience Store	Coffee Shop	Rock Club	Hotel	Do Resta
196	34.697306	135.198786	Udon Restaurant	3	Convenience Store	Coffee Shop	Rock Club	Hotel	Do Resta

Cluster 5

```
kobe_merged.loc[kobe_merged['Cluster Labels'] == 4, kobe_merged.columns[[1] + list(range
```

]:

	Neighborhood Latitude	Venue Longitude	Venue Category	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue
2085	34.71275	135.212060	Gym	4	Trail	Mountain	Bus Stop	Gym	Electronics Store	Fountain
2086	34.71275	135.208713	Bus Stop	4	Trail	Mountain	Bus Stop	Gym	Electronics Store	Fountain
2087	34.71275	135.208124	Mountain	4	Trail	Mountain	Bus Stop	Gym	Electronics Store	Fountain
2088	34.71275	135.209148	Trail	4	Trail	Mountain	Bus Stop	Gym	Electronics Store	Fountain

As a conclusion from this clustering, we can see here location as been classified as cluster is mainly a location for shopping activity. If we are interested to shop for clothes or etc. then this is the location we should be looking forward to.

Next, for cluster 2, this is a location of food and beverage. If we are hunting for foods then this should be the place we look to.

For cluster 3, this is also a place full of foods and beverage but at the same time, there are many groceries store nearby and it is also close to public transportation. This is probably a place to go to shop for groceries.

For cluster 4, this is the location of entertainment and leisure. There are many clubs and hotel nearby.

Lastly, for cluster 5, this is the place for someone who likes adventurous activity.