6/24/2021

Coursera Capstone

“The Battle of Neighbourhoods (Kyoto Edition)”

ShengJun

Capstone Project - The Battle of Neighbourhoods (Week 1)

Contents

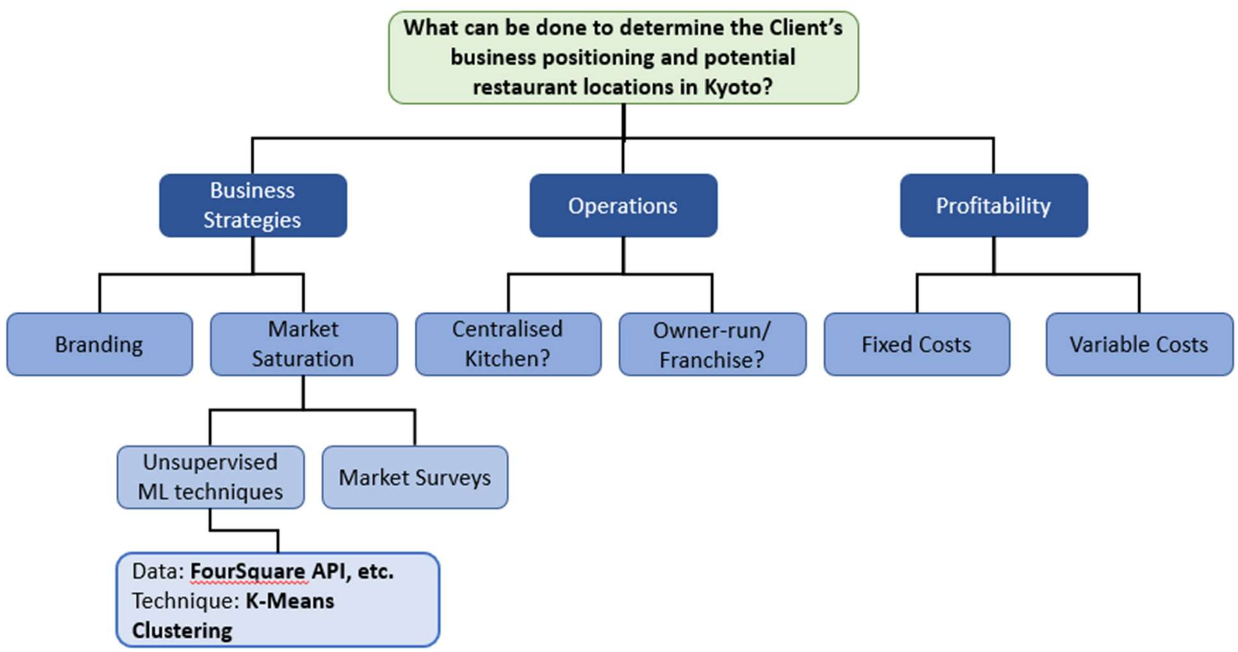
Introduction/ Business Problem

A client has approached the Consultation firm to advise on the business strategies and execution roadmap on setting up restaurants in Kyoto. The initial business problem question is **"Should the Client setup a restaurant chain in Kyoto, and where?"**

Imagined I have been assigned to this project. Working with the Client, we systemically reviewed the Client's business problem and outlined the following:

a) The Client is targeting to set up restaurant presence in Kyoto
b) They are not certain of the market saturation nor potential locations in Kyoto to act on.

The reframed problem statement is thus: **"What can be done to determine the Client's business positioning and potential restaurant locations in Kyoto?"**
With the reframed problem statement, we next worked closely with the client to establish the following top-level business drivers viz. Business Strategies, Operations and Profitability.



Along the line of Business Strategies, it was decided that Unsupervised Machine Learning technique could be applied to analysis and uncover insights valuable to influencing the formulation of Client's business strategies.
Specifically**, K-Means clustering** will be applied onto **the relevant restaurants' geo spatial data** to **cluster these entities and uncover insights such as viable restaurant themes and suitable restaurant locations**.

Data

Two data sources were identified for use. These are:

1) **List of Kyoto wards and their respective geo coordinates**. The wards list can be retrieved from the following webpage (https://en.wikipedia.org/wiki/Wards_of_Kyoto), whereas the coordinates can be retrieved using the geopy library.

2) **Restaurants in each neighborhood of Kyoto**. The data can be retrieved using the Foursquare API, and specifying the particular category of interest.

Data Collection & Preparation

The list of Kyoto wards was scraped from the afore-mentioned Wikipedia page. To facilitate the scraping process, the pandas **read_html()** method is used.

The relevant data resides in one of the table on the stated wikipedia url. Use pd.read_html() to retrieve the data.

In [55]:
```
# wards of kyoto url
ward_url = 'https://en.wikipedia.org/wiki/Wards_of_Kyoto'
```

In [56]:
```
# More concise method than requests and Beautiful soup
dataframe_list = pd.read_html(ward_url, flavor='bs4')
df_raw = dataframe_list[1]
df_raw
```

Out[56]:

	Name	Population	Density (/km²)	Area (km²)	% of total	Forest Area (km²)	Forest Cover	Inhabitable Area	
	Name	Population	Density (/km²)	Area (km²)	% of total	Forest Area (km²)	Forest Cover	(km²)	Density (/km²)
0	Fushimi-ku	280655	4552	61.66	7.4%	14.58	23.6%	47.12	5956
1	Higashiyama-ku	39044	5220	7.48	0.9%	1.58	21.1%	5.90	6618
2	Kamigyō-ku	85113	12107	7.03	0.8%	0.00	0.0%	7.03	12107
3	Kita-ku	119474	1259	94.88	11.5%	78.85	83.1%	16.02	7458
4	Minami-ku	99927	6320	15.81	1.9%	0.00	0.0%	15.81	6320

It is observed that the table returned has repeated column headers (i.e., multi-indexed). To collapse the column headers, the pandas **get_levels()** method is used.

In [57]:
```
# Data processing - refine and get relevant data
df_raw.columns = df_raw.columns.get_level_values(1)
```

In [58]:
```
df_raw
```

Out[58]:

	Name	Population	Density (/km²)	Area (km²)	% of total	Forest Area (km²)	Forest Cover	(km²)	Density (/km²)
0	Fushimi-ku	280655	4552	61.66	7.4%	14.58	23.6%	47.12	5956
1	Higashiyama-ku	39044	5220	7.48	0.9%	1.58	21.1%	5.90	6618
2	Kamigyō-ku	85113	12107	7.03	0.8%	0.00	0.0%	7.03	12107
3	Kita-ku	119474	1259	94.88	11.5%	78.85	83.1%	16.02	7458
4	Minami-ku	99927	6320	15.81	1.9%	0.00	0.0%	15.81	6320

Sanity checks is carried out regarding the data types and null values. A tally of expected number of Kyoto wards is also conducted.

```
In [60]:  1  # Check data types
          2  df_ward.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 11 entries, 0 to 10
Data columns (total 4 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Name            11 non-null     object 
 1   Population      11 non-null     int64  
 2   Density (/km²)  11 non-null     int64  
 3   Area (km²)      11 non-null     float64
dtypes: float64(1), int64(2), object(1)
memory usage: 440.0+ bytes

In [61]:  1  print(f"df_ward has {df_ward.shape[0]} rows, {df_ward.shape[1]} columns")

df_ward has 11 rows, 4 columns
```

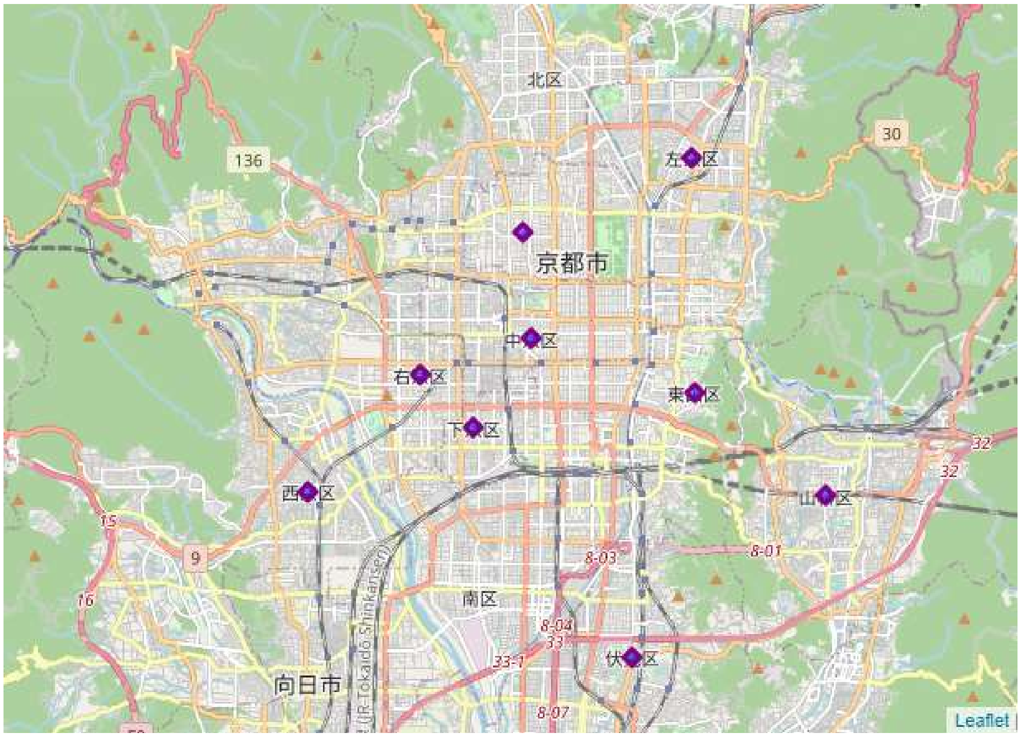
The data is then furnished with the geographical coordinates using the **geopy** library. Of specific note, a user-agent is specified as illustrated. This is because the Nominatum service runs on donated servers, which have limited capacity. Specifying a user-agent allows Open Street Map to track users. Not specifying user-agent would be a violation of their terms of service, and may result in blocking of one's IP address from accessing the service.

```
In [62]:  1  # Define and name user_Agent as "Kyoto_wards".
          2  geolocator = Nominatim(user_agent="Kyoto_wards")

In [63]:  1  # Get the coordinates per ward name
          2  df_ward['Name'].apply(geolocator.geocode).apply(lambda x: (x.latitude, x.longitude))

Out[63]: 0     (34.9535921, 135.7680447)
         1     (34.9985608, 135.7808951)
         2     (35.0257221, 135.7453647)
         3       (35.755838, 139.736687)
         4     (35.0887632, 136.9266427)
         5     (35.0076892, 135.7470707)
         6      (34.981741, 135.7008828)
         7      (35.0385567, 135.780494)
         8     (34.9925269, 135.7350963)
         9      (35.0015936, 135.724014)
         10    (34.9809453, 135.8081872)
         Name: Name, dtype: object
```

These coordinates are then combined into the data frame.

Data Visualization

Visualization of the Kyoto wards is done using the **folium** library.



A review of the number of wards plotted on the map indicates two missing wards. A possible hypothesis is that some of the coordinates of the wards returned could be those in other cities of Japan. A detailed review reveals that 'Kita-Ku' and 'Minami-ku' of Kyoto are not plotted on the Kyoto Map.

The next course of actions would be to first check the returned addresses for these two wards, then find the correct coordinates. Finally replace the coordinates in the dataframe.

```python
# Summary of the coordinates and wards
for lat, lng, label in zip(df_ward['Latitude'], df_ward['Longitude'], df_ward['Name']):
    print(lat,lng, label)
```

```
34.9535921 135.7680447 Fushimi-ku
34.9985608 135.7808951 Higashiyama-ku
35.0257221 135.7453647 Kamigyō-ku
35.755838 139.736687 Kita-ku
35.0887632 136.9266427 Minami-ku
35.0076892 135.7470707 Nakagyō-ku
34.981741 135.7008828 Nishikyō-ku
35.0385567 135.780494 Sakyō-ku
34.9925269 135.7350963 Shimogyō-ku
35.0015936 135.724014 Ukyō-ku
34.9809453 135.8081872 Yamashina-ku
```

While not immediately apparent, preliminary investigation indicate the coordinates for both Kita-ku and Minami-ku are off, compared to the other wards' coordinates.

The returned addresses confirm the hypothesis.

```
# Get the addresses for Kita-ku & Minami-ku
loc_kita = geolocator.geocode("Kita-ku")
loc_minami = geolocator.geocode("Minami-ku")
print(f"returned address of Kita-ku: {loc_kita.address}")
print(f"returned address of Minami-ku: {loc_minami.address}")
```

```
returned address of Kita-ku: 北区, 東京都, 日本
returned address of Minami-ku: 南区, 名古屋市, 愛知県, 日本
```

```
# Get the actual address of the kyoto wards
loc_kita_kyo = geolocator.geocode("Kita-ku, Kyoto")
loc_minami_kyo = geolocator.geocode("Minami-ku, Kyoto")
print(f"returned address of Kita-ku: {loc_kita_kyo.address}")
print(f"returned address of Minami-ku: {loc_minami_kyo.address}")
```

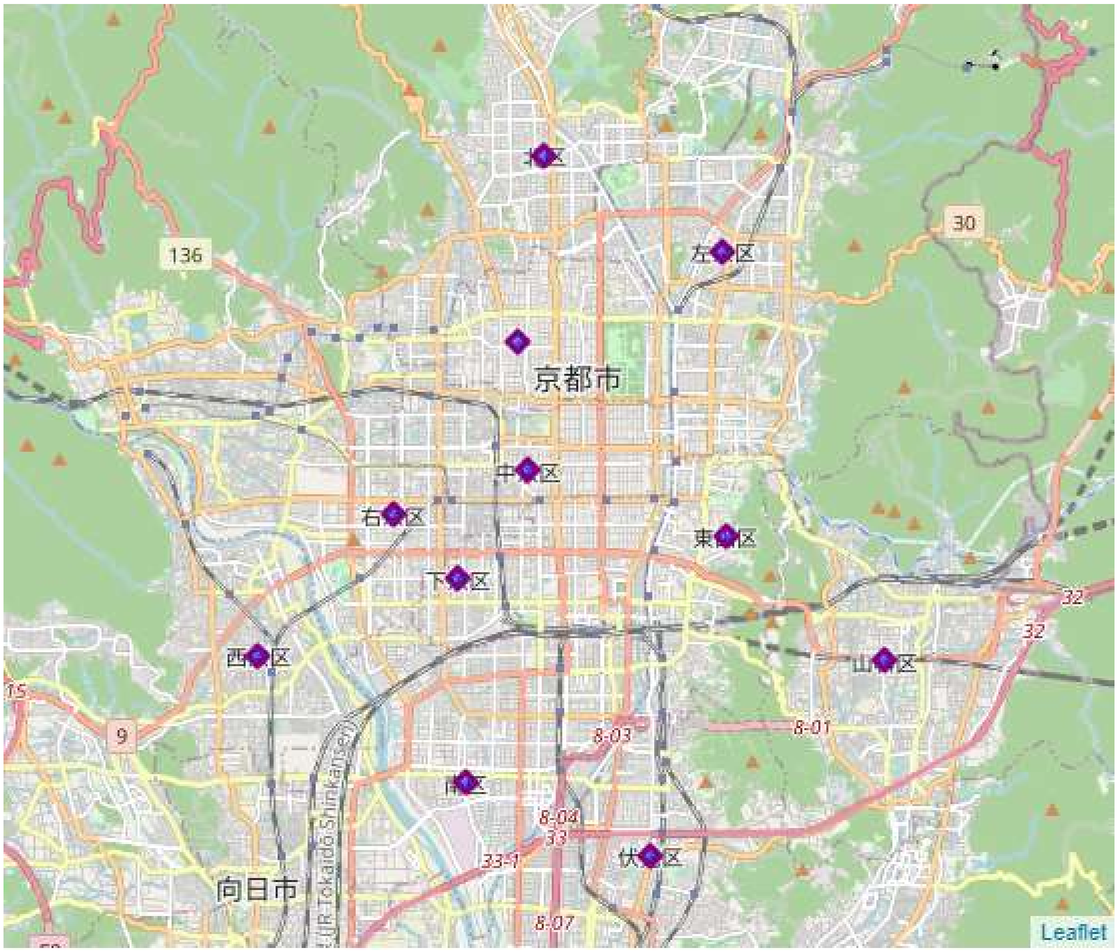
```
returned address of Kita-ku: 北区, 京都市, 京都府, 日本
returned address of Minami-ku: 南区, 京都市, 京都府, 日本
```

It is not uncommon for shared ward names across different areas of Japan. Specifying the desired city names facilitates the retrieval of correct coordinates.

```
kita_kyo_lat = loc_kita_kyo.latitude
kita_kyo_lon = loc_kita_kyo.longitude
minami_kyo_lat = loc_minami_kyo.latitude
minami_kyo_lon = loc_minami_kyo.longitude
print(f"corrected coordinates of Kita-ku, Kyoto: {kita_kyo_lat}, {kita_kyo_lon}")
print(f"corrected coordinates of Minami-ku, Kyoto: {minami_kyo_lat}, {minami_kyo_lon}")
```

```
corrected coordinates of Kita-ku, Kyoto: 35.0519284, 135.7499268
corrected coordinates of Minami-ku, Kyoto: 34.9637985, 135.736383
```

The corrected coordinates are then updated into the data frame. The updated map confirms the corrected coordinates with all eleven wards plotted on the map.



Exploratory Data Analysis

The restaurant data is retrieved using Foursquare API. Before scaling up to all wards in Kyoto, the first Kyoto ward is used for experimentation; Fushimi-ku.

Explore the first neighborhood in our dataframe.

```
# Name of first Kyoto ward in dataframe
nb_name = df_ward.loc[0,'Name']
print(f"First ward in Kyoto wards dataframe: {nb_name}")
```

```
First ward in Kyoto wards dataframe: Fushimi-ku
```

```
# Get this neighbourhood's lat and lon
nb_lat = df_ward.loc[0, 'Latitude']
nb_lon = df_ward.loc[0, 'Longitude']
nb_name = df_ward.loc[0,'Name']

print(f"{nb_name}'s latitude and longitude: {nb_lat} & {nb_lon}.")
```

```
Fushimi-ku's latitude and longitude: 34.9535921 & 135.7680447.
```

Specifying the criteria to return 100 restaurants within radius of 500 meters, the following data is returned.

```
# Number of venues returned by Foursquare
print(f"number of venues returned via Foursquare API: {nearby_venues.shape[0]}")
```

```
number of venues returned via Foursquare API: 25
```

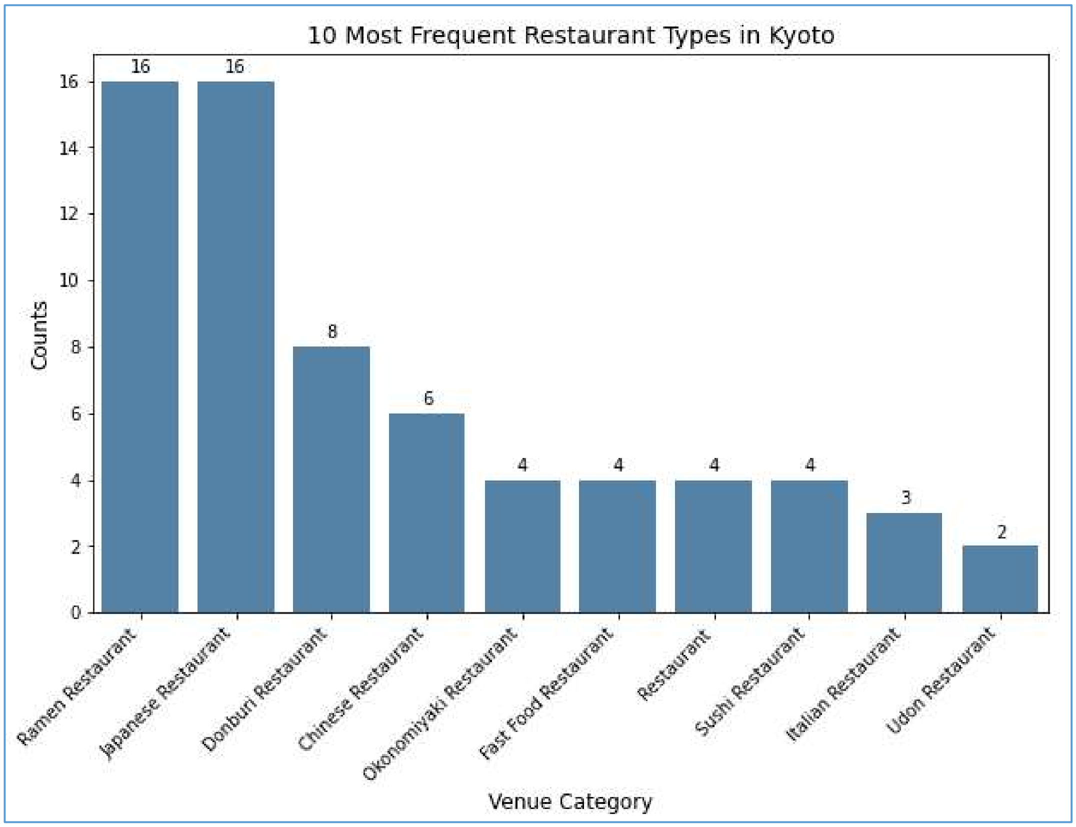
```
# Number of unique categories in the neighbourhood
print(f"number of unique venue categories: {len(nearby_venues['categories'].unique())}")
```

```
number of unique venue categories: 15
```

```
# venues by category count
nearby_venues['categories'].value_counts()
```

```
Convenience Store    5
Bakery               3
Park                 2
Café                 2
Intersection         2
Bus Stop             2
Science Museum       1
Train Station        1
Discount Store       1
Shopping Mall        1
Ramen Restaurant     1
Bath House           1
Supermarket          1
History Museum       1
Food & Drink Shop    1
```

Having verified the data retrieval process, it is scaled to all eleven wards. Closer inspection of the common types of restaurants by count across Kyoto wards reveals Ramen and Japanese restaurants as the most common.




```
Neighborhood
Fushimi-ku          1
Kita-ku             2
Minami-ku           2
Yamashina-ku        6
Kamigyō-ku          8
Nishikyō-ku         9
Shimogyō-ku         9
Nakagyō-ku         10
Sakyō-ku           10
Ukyō-ku            10
Higashiyama-ku     14
Name: Venue Category, dtype: int64
```

Notable observation#1: Drilling down into the restaurants by ward, it is discovered that **Higashiyama-ku, Ukyō-ku and Nishikyō-ku & Sakyō-ku** have higher density of restuarants (more than 10 in its area).

Further analysis is then conducted by grouping and studying the top five most common restaurant types by proportion for each ward.

Modeling

Noting the characteristics (i.e. restaurant types by proportion for each ward), **K-Means clustering** is then applied to **cluster these entities and uncover potential insights such as viable restaurant themes and suitable restaurant locations.** These insights could enhance the formulation of business strategies.

Several methods exists for establishing the optimum number of clusters k, such as the **Elbow method** and the **Silhouette method.**

Elbow Method

In this approach, the approach is to calculate the Within-Cluster-Sum of Squared Errors (WSS) for different values of k, and choose the k for which WSS becomes first starts to diminish. If the dataset is not well clustered (i.e. overlapping clusters), the elbow from the elbow method may potentially not be distinct.

Silhouette Method

The Silhouette method measures the similarity of a point to its own cluster, compared to other clusters. The range of the Silhouette coefficients is between positive one and negative one.

A positive coefficient tending close to positive one indicates the particular point is assigned in the ideal cluster. It also implies point is as practically distanced from the neighboring clusters as possible. A coefficient of zero indicates that the particular point is on or very close to the decision boundary between two neighboring clusters. A negative coefficient indicate that the point has been assigned to the wrong cluster.

In lieu of possibility of overlapping clusters, the Silhouette method is selected.

Use Silhouette method to discover optimum k

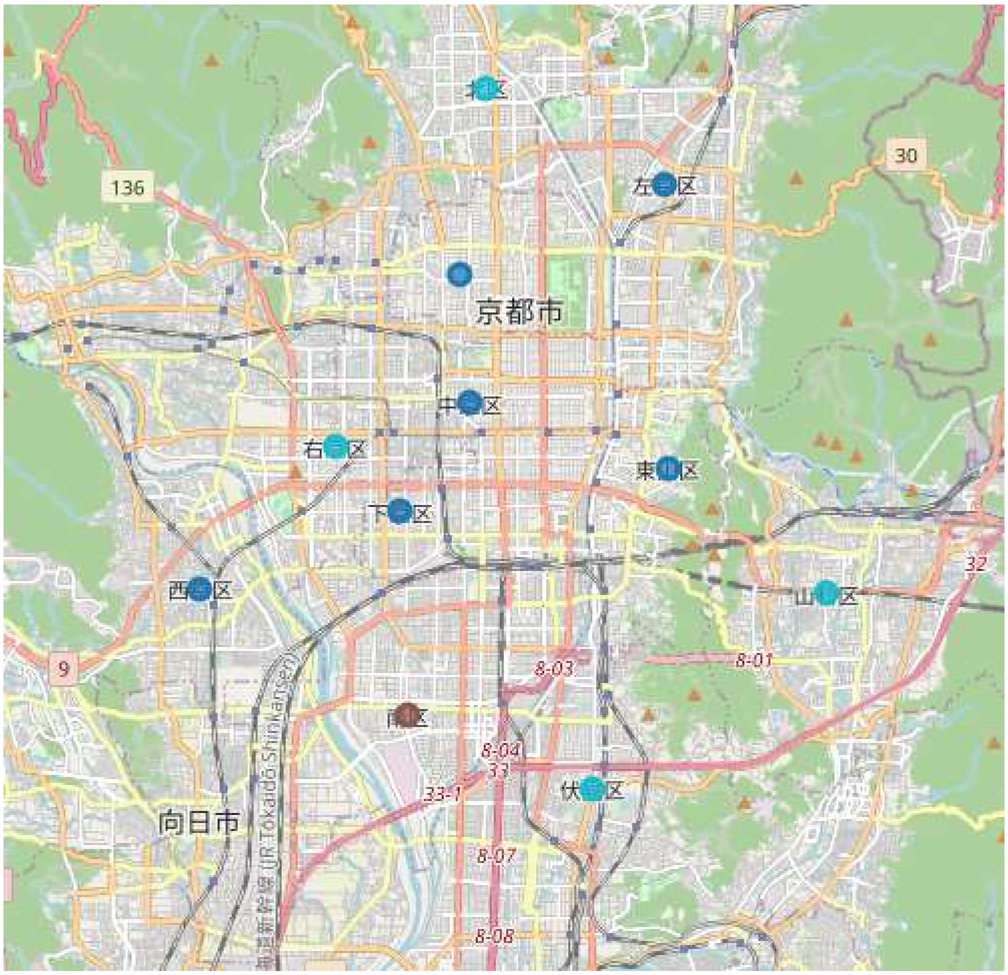
```python
# drop first column
kyoto_grouped_clustering = kyoto_grouped.drop('Neighborhood', axis=1)
```

```python
kmax = 10

# dissimilarity would not be defined for a single cluster, thus, minimum number of clusters should be 2
for k in range(2, kmax+1):
    kmeans = KMeans(init='k-means++', n_clusters=k, random_state=42).fit(kyoto_grouped_clustering)
    cluster_labels = kmeans.labels_

    silhouette_avg = silhouette_score(kyoto_grouped_clustering, cluster_labels)
    print(f"For n_clusters: {k}, average silhouette score: {silhouette_avg:.3f}")
```

```
For n_clusters: 2, average silhouette score: 0.192
For n_clusters: 3, average silhouette score: 0.194
For n_clusters: 4, average silhouette score: 0.108
For n_clusters: 5, average silhouette score: 0.122
For n_clusters: 6, average silhouette score: 0.091
For n_clusters: 7, average silhouette score: 0.099
For n_clusters: 8, average silhouette score: 0.086
For n_clusters: 9, average silhouette score: 0.056
For n_clusters: 10, average silhouette score: 0.030
```

Based on the coefficients, the optimum number of clusters is three.

The K-Means clustering is then implemented with init='k-means++', and random_state=42 for reproducibility of results. The resulting clusters is then plotted.



Evaluation & Discussion

Notable observations#2: Examining each cluster based on venue categories, the following observations are derived.

Ramen restaurants is predominantly prevalent in cluster 1. This is closely followed by restaurants offering asian-styled cuisine such as Chinese, Yoshoku or Sushi dishes.

```
kyoto_merged.loc[kyoto_merged['Cluster Labels'] == 0,
                 kyoto_merged.columns[[0] + list(range(4, kyoto_merged.shape[1]))]]
```

	Neighborhood	Latitude	Longitude	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 Most Common Venue
0	Fushimi-ku	34.953592	135.768045	0	Ramen Restaurant	Yoshoku Restaurant	Japanese Family Restaurant	Chinese Restaurant	Donburi Restaurant	Dumpling Restaurant	Fast Food Restaurant	French Restaurant	Indian Restaurant
3	Kita-ku	35.051928	135.749927	0	Chinese Restaurant	Ramen Restaurant	Yoshoku Restaurant	Japanese Family Restaurant	Donburi Restaurant	Dumpling Restaurant	Fast Food Restaurant	French Restaurant	Indian Restaurant
9	Ukyō-ku	35.001594	135.724014	0	Ramen Restaurant	Fast Food Restaurant	Japanese Restaurant	Udon Restaurant	Restaurant	Japanese Curry Restaurant	Chinese Restaurant	Donburi Restaurant	Dumpling Restaurant
10	Yamashina-ku	34.980945	135.808187	0	Sushi Restaurant	Ramen Restaurant	Japanese Family Restaurant	Donburi Restaurant	Yoshoku Restaurant	Chinese Restaurant	Dumpling Restaurant	Fast Food Restaurant	French Restaurant

Japanese restaurants is predominantly prevalent in cluster 2. This is closely followed by a mix of either Chinese or Ramen or Donburi restaurants.

```
kyoto_merged.loc[kyoto_merged['Cluster Labels'] == 1,
                 kyoto_merged.columns[[0] + list(range(4, kyoto_merged.shape[1]))]]
```

	Neighborhood	Latitude	Longitude	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 Mos Commo Venue
1	Higashiyama-ku	34.998561	135.780895	1	Japanese Restaurant	Donburi Restaurant	Soba Restaurant	Seafood Restaurant	Ramen Restaurant	Italian Restaurant	Japanese Curry Restaurant	Chinese Restaurant	Dumpli Restaura
2	Kamigyō-ku	35.025722	135.745365	1	Chinese Restaurant	Japanese Restaurant	Tonkatsu Restaurant	Sushi Restaurant	Donburi Restaurant	Fast Food Restaurant	Okonomiyaki Restaurant	Japanese Curry Restaurant	Dumpli Restaura
5	Nakagyō-ku	35.007689	135.747071	1	Chinese Restaurant	Donburi Restaurant	Restaurant	French Restaurant	Ramen Restaurant	Okonomiyaki Restaurant	Yakitori Restaurant	Yoshoku Restaurant	Japane Cur Restaura
6	Nishikyō-ku	34.981741	135.700883	1	Japanese Restaurant	Japanese Curry Restaurant	Chinese Restaurant	Donburi Restaurant	Sushi Restaurant	Restaurant	Ramen Restaurant	Yakitori Restaurant	Itali Restaura
7	Sakyō-ku	35.038557	135.780494	1	Japanese Restaurant	Ramen Restaurant	Asian Restaurant	Donburi Restaurant	Fast Food Restaurant	Mexican Restaurant	Italian Restaurant	Tonkatsu Restaurant	Japane Cur Restaura
8	Shimogyō-ku	34.992527	135.735096	1	Japanese Restaurant	Ramen Restaurant	Okonomiyaki Restaurant	Dumpling Restaurant	Indian Restaurant	Yoshoku Restaurant	Seafood Restaurant	Restaurant	Sol Restaura

Udon restaurants is predominantly prevalent in cluster 3, followed by Donburi restaurants.

```
kyoto_merged.loc[kyoto_merged['Cluster Labels'] == 2,
                 kyoto_merged.columns[[1] + list(range(5, kyoto_merged.shape[1]))]]
```

	Population	Longitude	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 Most Common Venue	10th Most Common Venue
4	99927	135.736383	2	Udon Restaurant	Donburi Restaurant	Yoshoku Restaurant	Japanese Family Restaurant	Chinese Restaurant	Dumpling Restaurant	Fast Food Restaurant	French Restaurant	Indian Restaurant	Italian Restaurant

Recommendations

Exploring the Neighbourhoods in Kyoto, Higashiyama-ku, Ukyō-ku and Nishikyō-ku & Sakyō-ku have higher density of restuarants (more than 10 in its area). The higher restaurant density could imply these areas as being more popular with visitors with more tourist attractions in their vicinities.

For example, Higashiyama-ku, features many historical sights such as the entertainment district of Gion in front of Yasaka Shrine, Ninenzaka, Sannenzaka and Kiyomizu Temple (designated as World Heritage site). Ukyō-ku is also home to many famous sites such as Tenryū-ji, and Arashiyama, a hill famed for its maple leaves.

The Preliminary recommended locations are Higashiyama-ku and Ukyō-ku for market entry.

Higashiyama-ku is assigned to cluster 2; a restaurant offering Japanese cuisine could have a higher chance of success with the visitors.

Ukyō-ku is assigned to cluster 1; a restaurant offering Ramen could have a higher chance of success with the visitors.

Regardless of above recommendations, the other fundamentals of F&B service such as quality food & services and strict hygiene practices are not to be overlooked.