The right spot for a bakery shop

Finding the right neighborhood

Alfonso De los Reyes

2019

# Business requirement



1. Our client, a local chef would like to appoint a data science agency (the “Contractor”) with the necessary expertise to design and conduct analysis to find the perfect location for a new **bakery shop** to be opened in **Singapore** in 6 months.
2. The bakery shop shouldn’t be a typical bakery like the currently existent chains in the country, the place should offer a new experience in more quiet areas in which consumers appreciate bakery and coffee and therefore ambience is important. The client is still undecided about the positioning of the place in terms of a bakery shop or a café instead.
3. Context: Singapore is dense in terms of food offering; few cities can offer the amount of variety of restaurants packed in so small spaces. According to the local authorities, there are around 8000 food service outlets in the country and therefore having a clear idea of the marketing mix is key to be successful.

* **Objective**: **Understand which neighborhoods or areas are more appropriate to open a shop.**
* **Area**: **The entire Singapore.**

**Restrictions / options**  **Measure**

1. *The place could be café or a bakery shop Frequency of bakery shops and cafes*
2. *It should be accessible Near a metro station*
3. *It should be a good area parks / gardens as measure*
4. *Avoid high rental Avoid “business” areas like downtown*

**Data sources:**

1. Postal codes from Singapore <http://download.geonames.org/export/zip/>: The entire number of postal codes that exist in Singapore, beyond the district area code. The file contains 121154 total postal codes with longitude and latitude information.
2. Foursquares data of venues.

**General methodology to be used:**

1. Take a sample from total postal codes database (320 instances).
2. Plot the sample in Singapore.
3. Use the sample data to extract the venues information from the Foursquare database.
4. Process the venues information in terms of categories and group by area.
5. Use the processed information to form clusters using K-means.
6. Plot clusters
7. Understand the clusters.
8. See if any clusters are suitable to provide a solution to the client.

# Step by step approach (Second week)

Install packages and import modules

* !conda install -c conda-forge geopy --yes;
* !conda install -c conda-forge geocoder --yes;
* !conda install -c conda-forge folium=0.5.0 --yes;

**…**

* from zipfile import ZipFile
* from sklearn.cluster import KMeans # import k-means from clustering stage
* import folium
* import numpy as np # useful for many scientific computing in Python
* import pandas as pd # primary data structure library
* pd.set\_option('display.max\_columns', None)
* pd.set\_option('display.max\_rows', None)
* import requests
* import io
* import matplotlib.cm as cm
* import matplotlib.colors as colors
* import json

Downloading zip file and extracting it

* !wget -q -O 'zipsing' <http://download.geonames.org/export/zip/SG.zip>

**…**

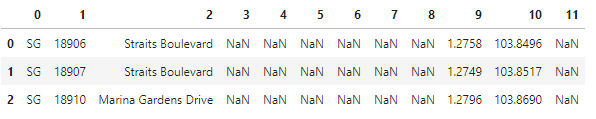
* with ZipFile('zipsing', 'r') as zip:
* zip.extract('SG.txt')

Observations: File stored in the local environment as SG.txt

Parsing the txt file and renaming columns

* data = pd.read\_csv('SG.txt',sep=' ', header=None)
* data.head(3)

**output ┐**



Observations: We need to drop and rename some columns

Dropping and renaming columns

* data.drop([0,3,4,5,6,7,8,11], axis = 1, inplace=True)
* print(data.shape)

output ┐

(121154, 4)

Observations: The file is huge, we need to take a random sample

Taking a sample of the original file, cleaning and sorting

* pcsubset = data.sample(n=320) #Postal code subset of the large database
* pcsub = pcsubset[:]

**…**

* pcsub.to\_csv('sampledata.csv') #Saving the dataframe to the local environment

Observations: The kernel died in different occasions and as the sample is random, I had issues with the output later

Sorting values, resetting index and visualizing the table

* pcsub.sort\_values(['area'], ascending=True, axis=0, inplace=True)

**…**

* pcsub = pcsub.reset\_index(drop=True)

**…**

* pcsub.head()

output ┐



Observations: Everything seems to be in order to plot the data

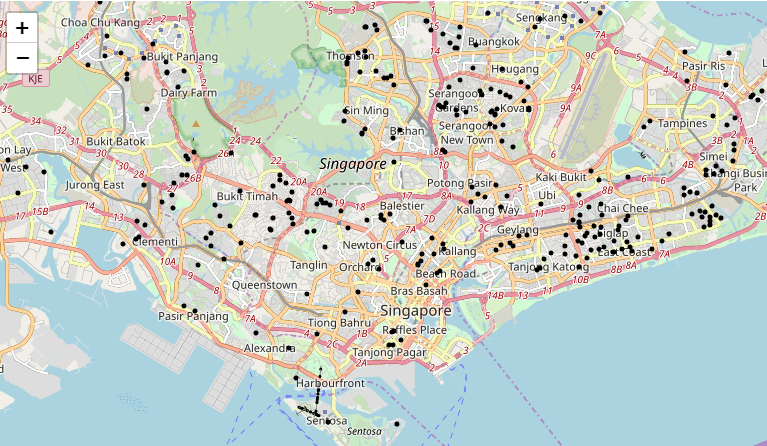
Define map coordinates and loop to add markers from random postal codes

* # Singapore latitude and longitude values
* latitude = 1.290270
* longitude = 103.8198
* singmap = folium.Map(location=[latitude, longitude], zoom\_start=12)
* fgroup = folium.map.FeatureGroup() # instantiate a feature group for the incidents in the dataframe

Plot the map

* for lat, lng, in zip(pcsub['latitude'], pcsub['longitude']):
* fgroup.add\_child(
* folium.features.CircleMarker(
* [lat, lng],
* radius=1, # define how big you want the circle markers to be
* color="black",
* fill=True,
* fill\_color="black",
* fill\_opacity=0.5
* )
* )
* # add incidents to map
* singmap.add\_child(fgroup)

output ┐



Observations: The black dots represents the 320 postal codes plotted. Why I choose to plot 320 points instead of 28 districts, because 28 districts is simply too small to cluster the data.

Defining ID / secret / Version to use in Foursquare

* CLIENT\_ID = 'HJLPKNFYSMPOPHISTM0LKEJHZC1MK2ENDRP1RWT5NAWSYMCN' # your Foursquare ID
* CLIENT\_SECRET = 'G1I1EM0WQWTCR0JH0H334RF5FJMZMTJGXSLICV01B2P2MMMU' # your Foursquare Secret
* VERSION = '20180605' # Foursquare API version

Create function to loop through postal codes and get nearby venues information

* LIMIT = 100 # limit of number of venues returned by Foursquare API
* def getNearbyVenues(names, latitudes, longitudes, radius=1000):
* 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']
* return(nearby\_venues)

Apply function using the sample data postal codes

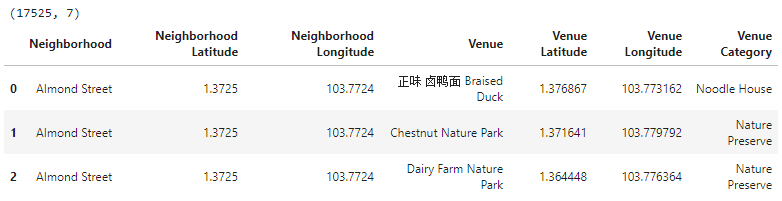
* snv = getNearbyVenues(names=pcsubset['area'], latitudes=pcsubset['latitude'], longitudes=pcsubset['longitude']) #Singapore nearby venues

Observations: This took a while and some errors were encountered with certain venues in other cases

Shape of venues data, backup and looking the table

* print(snv.shape)
* bnv = snv[:]
* bnv.head(3)

output ┐



Observations: Another huge table of 17525 rows

Saving into the local environment the Foursquare data

To not repeat the process in case something happens

* bnv.to\_csv('foursquare.csv') #Saving the dataframe as CSV in case the kernel dies

One hot encoding / Preparing data to use in the K-means clustering

* ctgrtable = pd.get\_dummies(bnv[['Venue Category']], prefix="", prefix\_sep="")
* ctgrtable.shape

output ┐

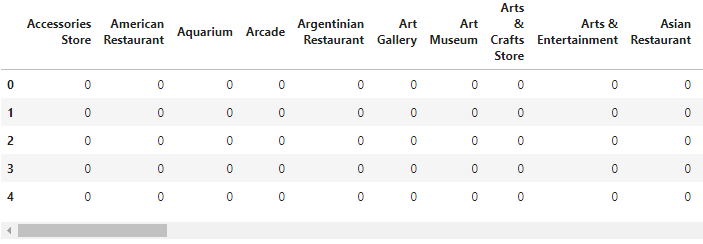
(17525, 344)

Observations: 344 categories

Quick look into the table

* ctgrtable.head(5)

output ┐



Observations: Many empty areas in the borders of the country

Grouping by Neighborhood and checking the new shape

* groupt = temptb.groupby('Neighborhood').sum().reset\_index()
* groupt.head()
* print(groupt.shape)

output ┐

(291, 344)

Observations: A dataframe of 291 rows (Areas) and 344 columns (Categories).

The groupt dataframe will be used in a paralel way for generating the top venues list but also to create the final dataframe for the clustering

1.- Preparing the dataframe for the k-mean clustering

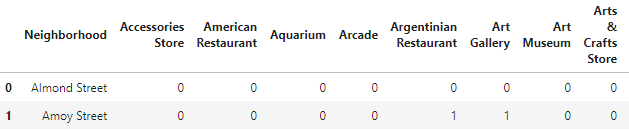
* forcluster = groupt[:] # As a backup but also is the dataframe to be used for clustering purposes

Bringing the original latitude and longitude columns

* forcluster['latitude'] = forcluster.Neighborhood.map(pcsub.set\_index('area')['latitude'].to\_dict()) # Bringing the original latitude column
* forcluster['longitude'] = forcluster.Neighborhood.map(pcsub.set\_index('area')['longitude'].to\_dict()) # Bringing the original longitude column
* print(forcluster.shape) # This dataframe includes also the latitude and longitude
* forcluster.head(2)

output ┐

(291, 344)



Observations: This dataframe is ready to be used in the K-means process

2.- Top venues per neighborhood

* num\_top\_venues = 10
* for hood in groupt['Neighborhood']:
* print("----"+hood+"----")
* temp = groupt[groupt['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')

output ┐

----Almond Street----

venue freq

0 Nature Preserve 3.0

1 Park 3.0

2 Food Court 3.0

3 Noodle House 2.0

4 Market 1.0

5 Coffee Shop 1.0

6 Seafood Restaurant 1.0

7 Flea Market 1.0

8 Parking 1.0

9 Cafeteria 1.0

.

.

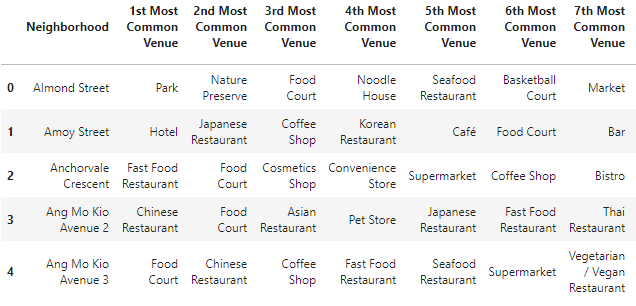
291 additional cases

Function definition and application to get a dataframe with the top 10 most common venues

* def return\_most\_common\_venues(row, num\_top\_venues):
* row\_categories = row.iloc[1:]
* row\_categories\_sorted = row\_categories.sort\_values(ascending=False)
* return row\_categories\_sorted.index.values[0:num\_top\_venues]
* num\_top\_venues = 10
* indicators = ['st', 'nd', 'rd']
* # create columns according to number of top venues
* columns = ['Neighborhood']
* for ind in np.arange(num\_top\_venues):
* try:
* columns.append('{}{} Most Common Venue'.format(ind+1, indicators[ind]))
* except:
* columns.append('{}th Most Common Venue'.format(ind+1))
* # create a new dataframe
* topv = pd.DataFrame(columns=columns)
* topv['Neighborhood'] = groupt['Neighborhood']
* for ind in np.arange(groupt.shape[0]):
* topv.iloc[ind, 1:] = return\_most\_common\_venues(groupt.iloc[ind, :], num\_top\_venues)
* topv.shape

Check new dataframe

* topv.head()

output ┐

Observations: Everything seems to be in order

Backup, rename and check shape

* topa = topv[:] # Backup
* topa.rename(columns={'Neighborhood':'area'}, inplace =True)
* print(topa.shape)

output ┐

(291, 11)

Running K-Means (Clusters are set to 10)

* # set number of clusters
* kclusters = 10
* tgc = forcluster.drop('Neighborhood', 1)
* # run k-means clustering
* kmeans = KMeans(n\_clusters=kclusters, random\_state=0).fit(tgc)
* # check cluster labels generated for each row in the dataframe
* kmeans.labels\_[0:10]

output ┐

(array([6, 3, 6, 9, 4, 2, 6, 2, 6, 0], dtype=int32)

Observations: Everything seems to be in order

Merging dataframes

Let's merge the dataframes that includes the cluster as well as the top 10 venues for each neighborhood.

* # add clustering labels
* topa.insert(0, 'Cluster Labels', kmeans.labels\_)
* tablemerged= pcsub[:]
* # merge toronto\_grouped with toronto\_data to add latitude/longitude for each neighborhood
* tablemerged = tablemerged.join(topa.set\_index('area'), on='area')

**…**

* tablemerged = tablemerged.drop('postalcode',1)

**…**

* tablemerged.head(5)

output ┐

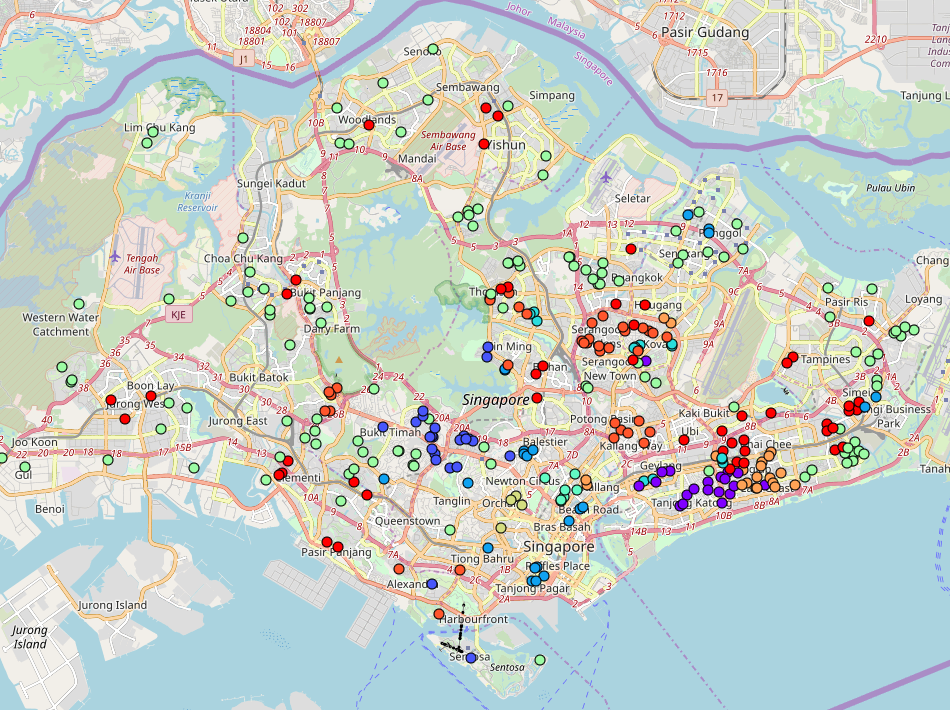


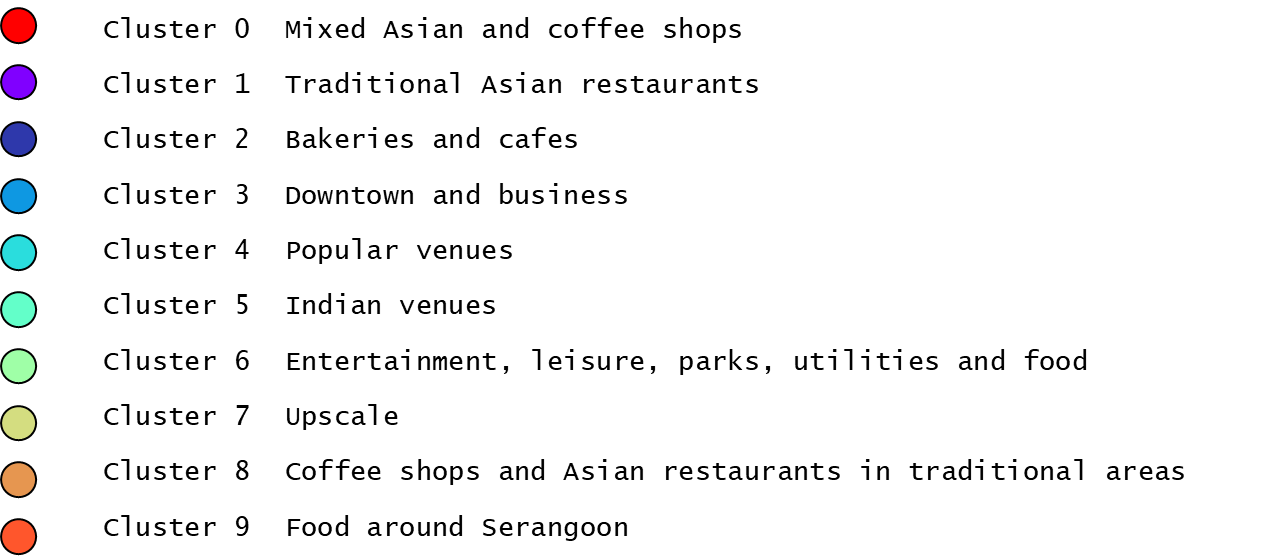
Observations: This table is ready, now we can proceed to plot the data

Plot the final map and clusters

* # create map
* map\_clusters = folium.Map(location=[latitude, longitude], zoom\_start=11)
* # set color scheme for the clusters
* x = np.arange(kclusters)
* ys = [i + x + (i\*x)\*\*2 for i in range(kclusters)]
* colors\_array = cm.rainbow(np.linspace(0, 1, len(ys)))
* rainbow = [colors.rgb2hex(i) for i in colors\_array]
* # add markers to the map
* markers\_colors = []
* for lat, lon, poi, cluster in zip(tablemerged['latitude'], tablemerged['longitude'], tablemerged['area'], tablemerged['Cluster Labels']):
* label = folium.Popup(str(poi) + ' Cluster ' + str(cluster), parse\_html=True)
* folium.CircleMarker(
* [lat, lon],
* radius=5,
* popup=label,
* color= 'black',
* weight=1,
* fill=True,
* fill\_color= rainbow[cluster -1],
* fill\_opacity=1).add\_to(map\_clusters)
* map\_clusters

output ┐





# Cluster analysis and final observations

We managed to divide postal codes in 10 clusters very distinct clusters that can be described as follows.



Cluster 0,  is located in different areas of the city and is mostly composed of mixed Asian restaurants and coffee shops.



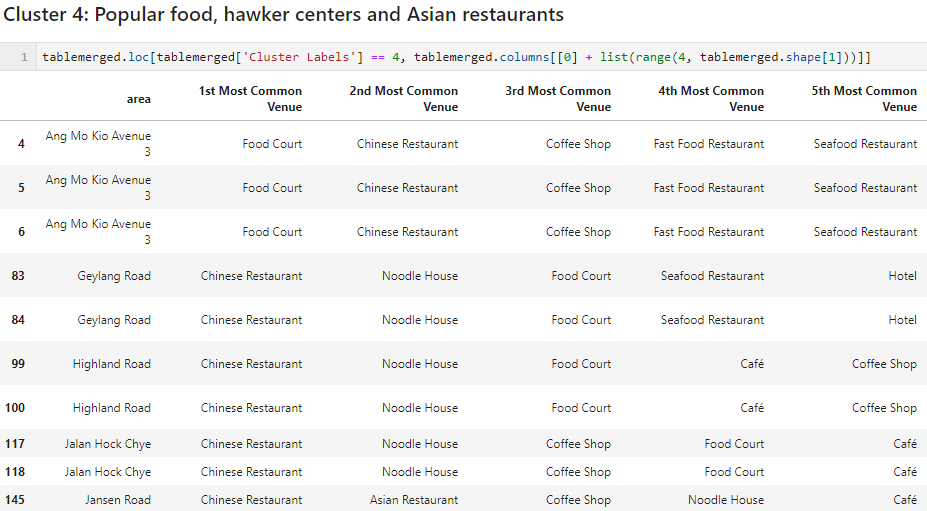
Cluster 1, is located in traditional areas of the city and is mostly composed of Chinese restaurants and Noodle houses.



Cluster 2,  Concentrated in the Bukit Timah area, a residential area composed of cafes, bakery shops, gardens, metro station and shopping mall. There is minimal public housing in this area.



Cluster 3,  Concentrated in the financial district area, composed by hotels, mixed restaurants and cafes.



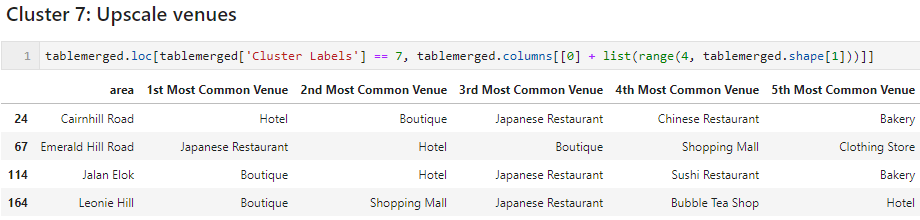
Cluster 4,  Concentrated along the Geylang area is composed of food court and fast food mostly.



Cluster 5,  Concentrated in little India mostly consists of Indian restaurants, cafes and hotels mostly.



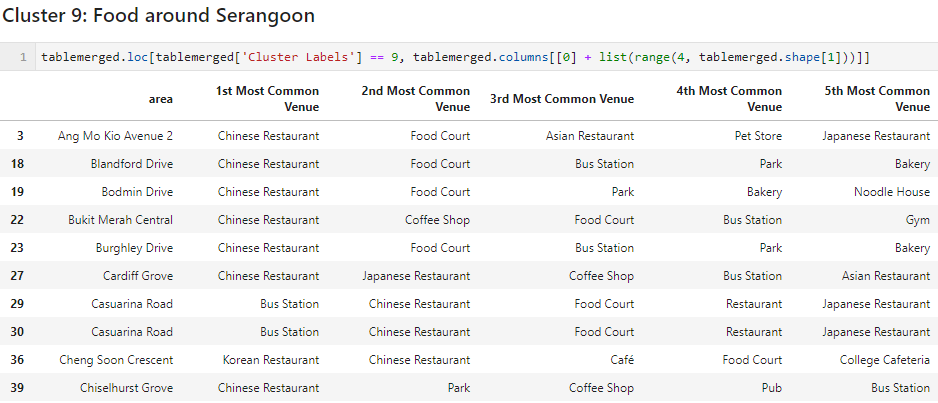
Cluster 6,  All around the city, this cluster is composed by supermarkets, food, beaches, parks, gun ranges, fast food, bus stops, etc.



Cluster 7,  Concentrated near downtown area, composed by hotels, boutiques and Japanese restaurants mostly.



Cluster 8,  Located in a traditional area of Singapore, is composed of coffee shops, cafes and Asian restaurants.



Cluster 9,  Asian restaurants concentrated in the Serangoon area

Final conclusions

According to our objective **[understand which neighborhoods or areas are more appropriate to open a shop].** Now we have a very good idea of how the different venues are distributed in the city with 2 or 3 main areas concentrating a larger number of bakery shops, cafes and similar.

Opening a bakery/café in Cluster 2 will offer a higher scale experience with a more modern experience, with younger affluent customers willing to pay more, furthermore the Cluster 2 area / Bukit Timah has some of the densest clusters of luxury condominiums and therefore potential to continue growing and is close to MRT station and is also composed of parks and leisure areas. On the other hand, cluster 8 also has plenty of coffee shops, however it is located in a much more traditional area and opening there would be an error. This area is also limited in terms of disposable income.

It is our recommendation then to open in the Bukit Timah area as shown in the map below.

