CAPSTONE PROJECT – THE BATTLE OF NEIGHBORHOODS

Table of Contents

I. Introduction/Business Problem Section

Today, '**Wine bar**' are very trendy in **Paris**. We would like to open our restaurant in the French capital, but some questions must be answered before reaching the final decision to open the store at a specific location in the city.

1. What "Customer base" are we going to target ?
    a) Are we targeting the business people during the lunch break or during the evening ?
2. Where is the "best place" to open it ?
    a) Some boroughs are more residential than others

    => For example the 14th and 16th boroughs are very residential (customers “on-place”).

    b) Some boroughs are more easier (administrative procedures) to open a new business

    => For example in the 11th borough, it is not possible anymore to open new bar.

    c) Are we near a Metro Station / Train ?
3. What are the competitor restaurants ?
    a) How many wine bars have already opened their business by borough ?
    b) Are they working correctly ?
    c) What are the feedbacks given by the users ? (satisfaction level)

All these stated questions can be alleviated by extracting relevant data. We will speak about it in the next section.

II. Data Extraction / Normalization

First of all we are going to extract the boroughs of Paris:
https://fr.geneawiki.com/index.php/Liste_des_quartiers_de_Paris

As we can see the borough can be divided into several 'Quarters'. It will allow us to have an even more refined search.

Firstly, we must extract the table containing the information about the quarters using beautifulSoup package :

[1]:
```
import urllib3
import bs4 as BeautifulSoup
import requests

import pandas as pd
import numpy as np
import geocoder # To get latitude and longitude for our postal codes

from sklearn.cluster import KMeans

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

import folium

from selenium import webdriver
```

[2]:
```
# More easy to extract the postal codes from this website instead of the wiki page.
url = 'https://fr.geneawiki.com/index.php/Liste_des_quartiers_de_Paris'
http = urllib3.PoolManager()
response = http.request('GET', url)

soup = BeautifulSoup.BeautifulSoup(response.data)

tables = soup.findAll('table')

# The Paris Boroughs table is the 2nd table in the web page
postalcodes_table = tables[1]
```

Once, we have extracted the table we must convert its content in order to exploit it. The data is already pretty well formatted, in our case we just have to remove the endings '*\n*' defined in the quarters cells:

[6]:
```
rows = postalcodes_table.find_all('tr')

headers = []
for cell in rows[0].find_all('td'):
    headers.append(cell.text.strip()) # remove \n
headers
```

[6]:
```
['Code INSEE 1', 'Code Postal', 'Arrondissements', 'Quartiers']
```

[7]:
```
# Translate headers in english and normalize it
headers[headers.index('Code INSEE 1')] = 'insee1_code'
headers[headers.index('Code Postal')] = 'postal_code'
headers[headers.index('Arrondissements')] = 'borough'
headers[headers.index('Quartiers')] = 'quarters'
headers
```

[7]:
```
['insee1_code', 'postal_code', 'borough', 'quarters']
```

[8]:
```
data = []
for row in rows[1:]: # Exclude headers line
    line = {}
    for cell_num, cell in enumerate(row.find_all('td')):
        cell = cell.text.strip()
        if cell_num <= 1: # Indexes 0 and 1
            line[headers[cell_num]] = cell
        elif cell_num == 4:
            line['quarters'] = cell.split('\n')
        else: # indexes 2 and 3 -> Join these 2 values, for example: "I - Le Louvre"
            borough = headers[2]
            if borough not in line:
                line[borough] = []
            line[borough].append(cell)
    data.append(line)
```

Once we have our data formatted, we can build it into a “usable” dataframe :

[9]:
```
df = pd.DataFrame(data)
df
```

[9]:

	borough	insee1_code	postal_code	quarters
0	[I, Le Louvre]	75101	75001	[01 - Saint-Germain-l'Auxerrois, 02 - Les Hall...
1	[II, La Bourse]	75102	75002	[05 - Gaillon, 06 - Vivienne, 07 - Le Mail, 08...
2	[III, Le Temple]	75103	75003	[09 - Les Arts-et-Métiers, 10 - Les Enfants-Ro...
3	[IV, L'Hôtel-de-Ville]	75104	75004	[13 - Saint-Merri, 14 - Saint-Gervais, 15 - L'...
4	[V, Le Panthéon]	75105	75005	[17 - Saint-Victor, 18 - Le Jardin-des-Plantes...
5	[VI, Le Luxembourg]	75106	75006	[21 - La Monnaie, 22 - L'Odéon, 23 - Notre-Dam...
6	[VII, Le Palais-Bourbon]	75107	75007	[25 - Saint-Thomas-d'Aquin, 26 - Les Invalides...
7	[VIII, L'Élysée]	75108	75008	[29 - Les Champs-Élysées, 30 - Le Faubourg-du-...
8	[IX, L'Opéra]	75109	75009	[33 - Saint-Georges, 34 - La Chaussée-d'Antin,...
9	[X, L'Enclos-Saint-Laurent]	75110	75010	[37 - Saint-Vincent-de-Paul, 38 - La Porte-Sai...
10	[XI, Popincourt]	75111	75011	[41 - La Folie-Méricourt, 42 - Saint-Ambroise,...
11	[XII, Reuilly]	75112	75012	[45 - Le Bel-Air, 46 - Picpus, 47 - Bercy, 48 ...
12	[XIII, Les Gobelins]	75113	75013	[49 - La Salpêtrière, 50 - La Gare, 51 - La Ma...
13	[XIV, L'Observatoire]	75114	75014	[53 - Le Montparnasse, 54 - Parc Montsouris 2,...
14	[XV, Vaugirard]	75115	75015	[57 - Saint-Lambert, 58 - Necker, 59 - Grenell...
15	[XVI, Passy]	75116	75016	[61 - Auteuil, 62 - La Muette, 63 - La Porte-D...
16	[XVII, Les Batignolles-Monceau]	75117	75017	[65 - Les Ternes, 66 - La Plaine-Monceau, 67 -...
17	[XVIII, La Butte-Montmartre]	75118	75018	[69 - Les Grandes-Carrières, 70 - Clignancourt...
18	[XIX, Les Buttes-Chaumont]	75119	75019	[73 - La Villette, 74 - Le Pont-de-Flandre, 75...
19	[XX, Ménilmontant]	75120	75020	[77 - Belleville, 78 - Saint-Fargeau, 79 - Le ...

This is “raw” dataframe, we can add additional values in it and formalize it next. For example, we will add the GPS location of the “quarters”.

But, there are more than 80 quarters in Paris and there is no formatted table containing this GPS information. The data extraction of this feature is pretty painful manually. So, we will have to use a “Web Browser Bot” in order to extract this information automatically from a given wiki page.

In order to keep this report readable,we will let you browse the attached notebook from “Annexes” section. Please, refer to the attached notebook defined in the to have even more details about the dataframe normalization.

In our “finalized” dataframe, we have listed every GPS locations of each quarters of Paris and “disaggregate” the boroughs of Paris by quarter as shown below:

```
[21]: paris_df.head(10)
```

[21]:

	borough	insee1_code	postal_code	quarter	latitude	longitude
0	I - Le Louvre	75101	75001	01 - Saint-Germain-l'Auxerrois	48.860112	2.340195
1	I - Le Louvre	75101	75001	02 - Les Halles	48.862541	2.344744
2	I - Le Louvre	75101	75001	03 - Le Palais-Royal	48.864912	2.337749
3	I - Le Louvre	75101	75001	04 - La Place-Vendôme	48.867495	2.329402
4	II - La Bourse	75102	75002	05 - Gaillon	48.869083	2.332867
5	II - La Bourse	75102	75002	06 - Vivienne	48.869069	2.339176
6	II - La Bourse	75102	75002	07 - Le Mail	48.867982	2.344615
7	II - La Bourse	75102	75002	08 - Bonne-Nouvelle	48.866776	2.350087
8	III - Le Temple	75103	75003	09 - Les Arts-et-Métiers	48.866253	2.356846
9	III - Le Temple	75103	75003	10 - Les Enfants-Rouges	48.864729	2.363155

Now, we can begin to explore the quarters of Paris based on the information contained into this dataframe.

III. Data Exploration & Methodology

Once, we have the GPS location of each quarters of Paris, we can fetch the venues list from the Foursquare API for each quarter:

[23]:
```
def getNearbyQuarter(names, latitudes, longitudes, radius=500, LIMIT=100):
    venues_list=[]
    for name, lat, lng in zip(names, latitudes, longitudes):
        url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={},{}&radius={}&limit={}'.forr
            CLIENT_ID,
            CLIENT_SECRET,
            VERSION,
            lat,
            lng,
            radius,
            LIMIT)

        results = requests.get(url).json()["response"]['groups'][0]['items']

        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])

    columns = ['quarter', 'quarter_latitude', 'quarter_longitude',
               'interest_point', 'interest_point_latitude', 'interest_point_longitude', 'interest_point_type']
    data = [item for venue_list in venues_list for item in venue_list]

    return pd.DataFrame(data=data, columns=columns)

paris_interest_points = getNearbyQuarter(paris_df['quarter'], latitudes=paris_df['latitude'], longitudes=paris_df['longitude'])
```

Now, we have all ‘venues‘ or ‘points of intererest’ listed by the Foursquare API concerning Paris:

[25]:

	quarter	quarter_latitude	quarter_longitude	interest_point	interest_point_latitude	interest_point_longitude	interest_point_type
0	01 - Saint-Germain-l'Auxerrois	48.860112	2.340195	Cour Carrée du Louvre	48.860360	2.338543	Pedestrian Plaza
1	01 - Saint-Germain-l'Auxerrois	48.860112	2.340195	Place du Louvre	48.859841	2.340822	Plaza
2	01 - Saint-Germain-l'Auxerrois	48.860112	2.340195	La Vénus de Milo (Vénus de Milo)	48.859943	2.337234	Exhibit
3	01 - Saint-Germain-l'Auxerrois	48.860112	2.340195	Le Fumoir	48.860424	2.340868	Cocktail Bar
4	01 - Saint-Germain-l'Auxerrois	48.860112	2.340195	Église Saint-Germain-l'Auxerrois (Église Saint...	48.859520	2.341306	Church
5	01 - Saint-Germain-l'Auxerrois	48.860112	2.340195	La Régalade Saint-Honoré	48.861620	2.341749	French Restaurant
6	01 - Saint-Germain-l'Auxerrois	48.860112	2.340195	Pont des Arts	48.858565	2.337635	Bridge
7	01 - Saint-Germain-l'Auxerrois	48.860112	2.340195	Pret A Manger	48.861811	2.341311	Sandwich Place
8	01 - Saint-Germain-l'Auxerrois	48.860112	2.340195	Coffee Crêpes	48.858841	2.340802	Coffee Shop
9	01 - Saint-Germain-l'Auxerrois	48.860112	2.340195	Boutique yam'Tcha	48.861710	2.342380	Chinese Restaurant

[26]:
```
paris_interest_points.shape
```

[26]:
```
(5606, 7)
```

We have extracted 5606 'interesting points' (which can be shops / restaurant / historical monuments, ...).

(This is not a very large sample but we will try to exploit and find cluster into this dataset).

Summary of the data extracted: This dataset will help us to visualize the distribution of the different shops/monuments in the city. Like that, we can avoid to open our store next to another opened Wine bar.

Also, it will give us an overview of an 'attractive' location position (for example: 'Pont des Arts' which is located here: *48.858565, 2.337635*) which will make a place more popular than another one.

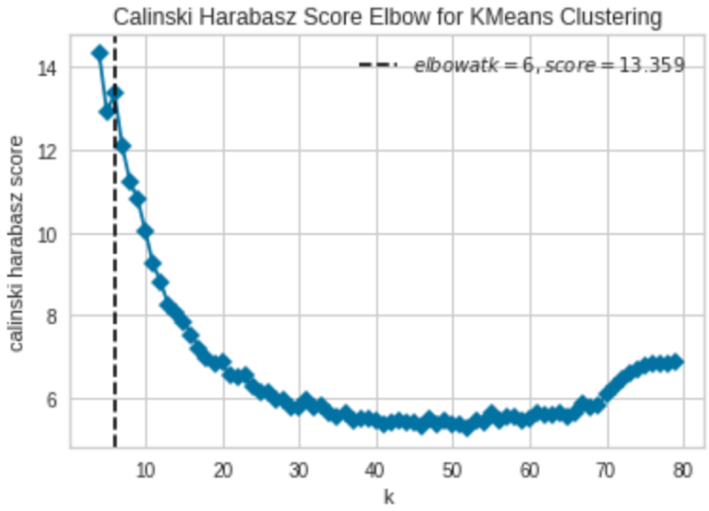
We have now extracted our dataset from Foursquare in order to analyze the restaurants distribution in Paris.

There is also another website which can be really interesting in order to add other data to our current dataset which is: https://opendata.paris.fr/pages/home/

We will use the KMeans approach to find the best place to open our Wine Bar. So first of all, we will have to convert our current dataframe into a dataframe which give the ratio of “venues” (or interesting points) per quarter.

Once we have created this new dataframe, we can find the best '*k*' using the Elbow method before building our 'best' KMeans model:

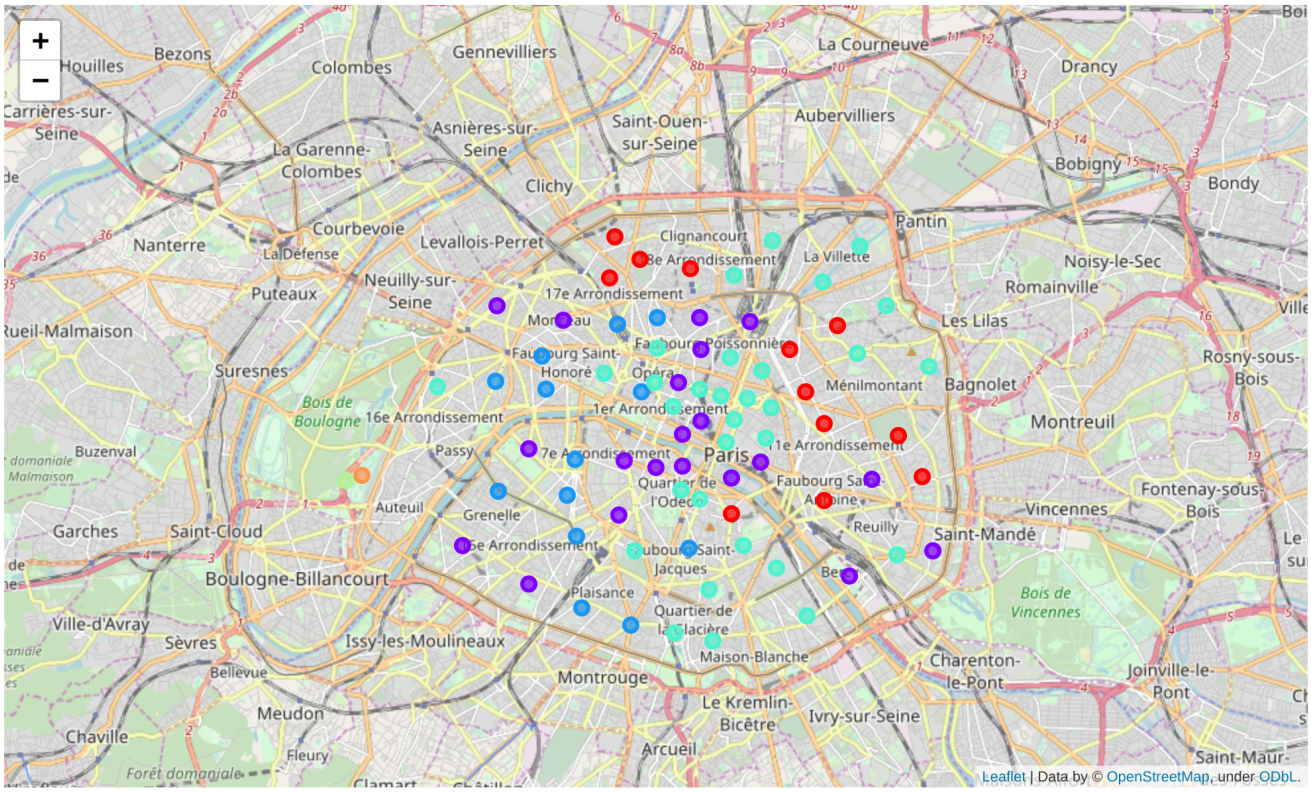
[33]:
```
# Instantiate the clustering model and visualizer
model = KMeans()
visualizer = KElbowVisualizer(
    model, k=(4,80), metric='calinski_harabasz', timings=False
)

visualizer.fit(paris_grouped_clustering)        # Fit the data to the visualizer
visualizer.show()        # Finalize and render the figure
```



[33]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6aa4f08358>
```

From the previous figure, we can see that the optimal number of clusters is 6 (the point of "inflection" of the curve).

So now we can display our clusters into the Paris map as shown below:



IV. Result Section

[36]:
```
for cluster_num in range(0, kclusters):
    cluster = paris_merged.loc[paris_merged['cluster_labels'] == cluster_num, paris_merged.columns[[1] + list(range(5, paris_me
    most_common_venues = cluster.groupby('1st Most Common Venue')['1st Most Common Venue'].count().sort_values(ascending=False)
    print('Most common venues for Cluster {}: {}'.format(cluster_num, ', '.join(most_common_venues)))
```

```
Most common venues for Cluster 0: French Restaurant, Bar
Most common venues for Cluster 1: French Restaurant
Most common venues for Cluster 2: Hotel, French Restaurant
Most common venues for Cluster 3: French Restaurant, Bar, Hotel, Japanese Restaurant, Cocktail Bar, Bakery, Wine Bar, Supermark
et, Coffee Shop, Bus Stop
Most common venues for Cluster 4: Nightclub
Most common venues for Cluster 5: Racecourse
```

First of all, we can see that Paris is very focused on "Restaurants". The "French" gastronomy encapsulates a large amount of specialities from all over France. For example, you can have Lyon Specialities (Brioche-style sausage, ...) or the "Breton crepe", the Cassoulet of Toulouse, etc... .

So, if we are going to open a Wine bar which can be assimilated by some people as a "Restaurant", we must focus on clusters were "Restaurants" are not very present.

The advantage to open a Wine bar into these clusters without restaurants are:

- to have less competitors
- to retain new customers who are not already 'accustomed' customers to nearby restaurants

So, now we will check clusters “without restaurants” (which are clusters 4 and 5).

[39]:
```
paris_merged[paris_merged['cluster_labels'].isin([4,5])]
```

[39]:

	borough	insee1_code	postal_code	quarter	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 M Com Ve
60	XVI - Passy	75116	75016	61 - Auteuil	48.852235	2.252798	4	Nightclub	Restaurant	Outdoors & Recreation	Racecourse	Nature Preserve	Plaza	C
61	XVI - Passy	75116	75016	62 - La Muette	48.853139	2.257004	5	Racecourse	Circus	Zoo Exhibit	Farmers Market	English Restaurant	Ethiopian Restaurant	E Sp

V. Discussion

=> It corresponds to the 16th borough of Paris.

The 16th borough of Paris also has a lot of famous monuments like:

```
paris_interest_points[(paris_interest_points['quarter'].isin(['61 - Auteuil', '62 - La Muette']))]
```

	quarter	quarter_latitude	quarter_longitude	interest_point	interest_point_latitude	interest_point_longitude	interest_point_type
4706	61 - Auteuil	48.852235	2.252798	Hippodrome d'Auteuil	48.851933	2.256789	Racecourse
4707	61 - Auteuil	48.852235	2.252798	Jockey Disque	48.853783	2.255724	Nightclub
4708	61 - Auteuil	48.852235	2.252798	Le Petit Jean Bouin	48.849923	2.250572	Outdoors & Recreation
4709	61 - Auteuil	48.852235	2.252798	Bel Ami, Paris	48.854830	2.255355	Restaurant
4710	61 - Auteuil	48.852235	2.252798	Wizard Trees	48.854492	2.249035	Nature Preserve
4711	61 - Auteuil	48.852235	2.252798	Cirque de Noël Christiane Bouglione	48.855720	2.254768	Circus
4712	61 - Auteuil	48.852235	2.252798	Porte de St Cloud	48.851416	2.259043	Plaza
4713	62 - La Muette	48.853139	2.257004	Hippodrome d'Auteuil	48.851933	2.256789	Racecourse
4714	62 - La Muette	48.853139	2.257004	Cirque Alexis Gruss	48.857376	2.254971	Circus

We can see that there are some different type of "venues" in these 2 quarters which can attract several type of people. As said before the data sample is not quite big and some important "venues" are not listed by Foursquare API (For example there are lot of museums in the 16th borough).

VI. Conclusion

We can see that the best place to open our new Wine bar is the 16th borough of Paris and particularly in the "62 - La Muette" quarter (as the 2nd most common venue for the "61 -Auteuil" quarter is "Restaurant" category). As shown previously there are lot of different venues "type" in this borough which can attract different type of people and also these 2 quarters are residential. So, there are people at any time of the day (unkike some boroughs which are mainly composed of offices so there are less people in the evening.

V. Annexes

This section summarizes all links used to build this report :

Content's Name	Link
My Notebook	*https://github.com/thoscc/Coursera_Capstone/blob/master/capstone_project_the_battle_of_neighborhoods.ipynb*
Paris Borough	https://fr.geneawiki.com/index.php/Liste_des_quartiers_de_Paris
Paris OpenData	https://opendata.paris.fr/pages/home/