The battle of the neighborhoods (REPORT)

Addressing a business problem using a combination of structured problem solving, data analysis & machine learning.

CONTENTS

1 Introduction/ Business Problem

Paris is one of the most important cities in the world which attracts every year millions of tourists from all over the world. It is as well very famous for its excellent cuisine both national and international. One of the main actors in in international cousine is as well Italy which has exported all over the world its food culture starting from traditional Italian Pizza and spaghetti to many different regional food always appreciated.

Is there still, room for Italian food in Paris?

We think yes in general because quality is always appreciated, but even more for some types of foods that, being very popular in Italy, even for strangers visiting the country, still have a limited presence (compared to others) in some countries like France and specifically in Paris.

We are looking in this case to home-made Ice-cream. Wea re looking for several high-traffic locations not necessary in the very centre of the town. So we focus on that borough during our analysis. We define potential neighbourhood based on the number of potential competitors for ice-cream, pastries and in general frozen sweet food which are operating right in each neighbourhood. Paris has full potential but also it is a very challenging district to open a business because of high competition. New shops should be open where we can ensure that we have enough customers, where we are not so close to direct competitors or where, if we are close to others, we are sure to provide the best quality in order to attract as many customers as possible.

The scope of the project is to advise on the business strategies and execution roadmap on setting up an Italian Ice-cream shop in Paris. The initial business problem question is “where is then better location in Paris to open one (or more) Italian home-made Ice-cream shop?”.

The City of Paris is divided for administrative reasons into 20 main boroughs (arrondissments) for administrative purposes. Each of these administrative districts (or arrondissements) are officially divided into 4 quartiers.

You will find in this report a map showing Paris arrondissements and a map presenting Paris neighbourhoods. Within each of its boroughs, the neighbourhoods are providing the typical flavour of Paris with their own culture, locations and charme. The twenty arrondissements of Paris all have their own characteristics in terms of buildings, places to go out, restaurants, and in general aspects that can influence the success of an initiative to open a new activity.

2 Data Collection and cleaning

Paris has a total of 20 boroughs and 80 neighbourhoods. In order to segment the neighbourhoods and explore them, we will use sets of data available at data.govr.fr an open platform containing numerous sets of public French data.

we will use 2 data sets:

The first consists of the list of the 20 boroughs with their respective geographical coordinates:

https://www.data.gouv.fr/fr/datasets/r/0d3553c6-45c0-4b16-82be-5ef314437d3e

The second dataset consists of the 80 neighbourhoods with their respective geographical coordinates:

https://www.data.gouv.fr/fr/datasets/r/a3b31fdc-85dc-4aeb-94c6-a8b57aebef77

Merging these 2 datasets after having them cleaned we will have a final data frame with the 20 boroughs and the 4 neighbourhoods for each of them for a total of 80 with related geographical coordinates.

3 Methodology

In this project we will then use Folium to represent all these locations on a Paris Map and the Foursquare API to provision venues information for each neighbourhood.

With this additional information we will explore neighbourhood and we will cluster them in search of the best ones where to open our chain of home-made Ice cream.

First we install all necessary libraries

```
! pip install folium
!pip install beautifulsoup4
!pip install lxml
import requests # library to handle requests
import pandas as pd # library for data analsysis
import numpy as np # library to handle data in a vectorized manner
import random # library for random number generation
from geopy.geocoders import Nominatim # module to convert an address into latitude and longitude values

from IPython.display import Image
from IPython.core.display import HTML


from IPython.display import display_html
import pandas as pd
import numpy as np

# tranforming json file into a pandas dataframe library
from pandas.io.json import json_normalize

!conda install -c conda-forge folium=0.5.0 --yes
import folium # plotting library
from bs4 import BeautifulSoup
from sklearn.cluster import KMeans
import matplotlib.cm as cm
import matplotlib.colors as colors

print('Folium installed')
print('Libraries imported.')
```

We access the first file in repository Data.Gov.Fr, consisting of the list of the 20 boroughs with their respective geographical coordinates.

```
# Retrive url and transform in Pandas Dataframe
url ="https://www.data.gouv.fr/fr/datasets/r/0d3553c6-45c0-4b16-82be-5ef314437d3e"
df =pd.read_csv(url, sep=';')
df.head()
```

We clean data and produce a simple data frame with Borough Name and geographical data.

	Borough N	Borough	Lat	long
0	3	Temple	48.862872	2.360001
1	11	Popincourt	48.859059	2.380058
2	14	Observatoire	48.829245	2.326542
3	1	Louvre	48.862563	2.336443
4	7	Palais-Bourbon	48.856174	2.312188

We access the second file in repository Data.Gov.Fr, consisting of the list of the 80 neighboroughs with their respective geographical coordinates.

```
# access file and transform in Pandas dathttps://eu-gb.dataplatform.cloud.ibm.com/data/jupyter2/runtimeenv2/v1/wdpx/service/notebook/conda2py37oce38ef488a3fbe41c7b2536e1346e781f9/dsxjpy/KRzo_DilmGMY_
url ="https://www.data.gouv.fr/fr/datasets/r/a3b31fdc-85dc-4aeb-94c6-a8b57aebef77"
df1=pd.read_csv(url, sep=';')
df1.head()
```

We clean data and produce a simple dataframe with Neighborough Name and geographical data.

	Borough N	Borough	Lat	long
0	3	Temple	48.862872	2.360001
1	11	Popincourt	48.859059	2.380058
2	14	Observatoire	48.829245	2.326542
3	1	Louvre	48.862563	2.336443
4	7	Palais-Bourbon	48.856174	2.312188

Tables merge

We merge the 2 tables obtaining a final one containing Boroughs, Neiborhoods and Geographical coordinates

	Borough N	Borough	Neighborhood	Latitude	Longitude
0	3	Temple	Archives	48.859192	2.363205
1	3	Temple	Enfants-Rouges	48.863887	2.363123
2	3	Temple	Arts-et-Métiers	48.866470	2.357083
3	3	Temple	Sainte-Avoie	48.862557	2.354852
4	11	Popincourt	Saint-Ambroise	48.862345	2.376118

Now we want to locate on map the Neighborhoods

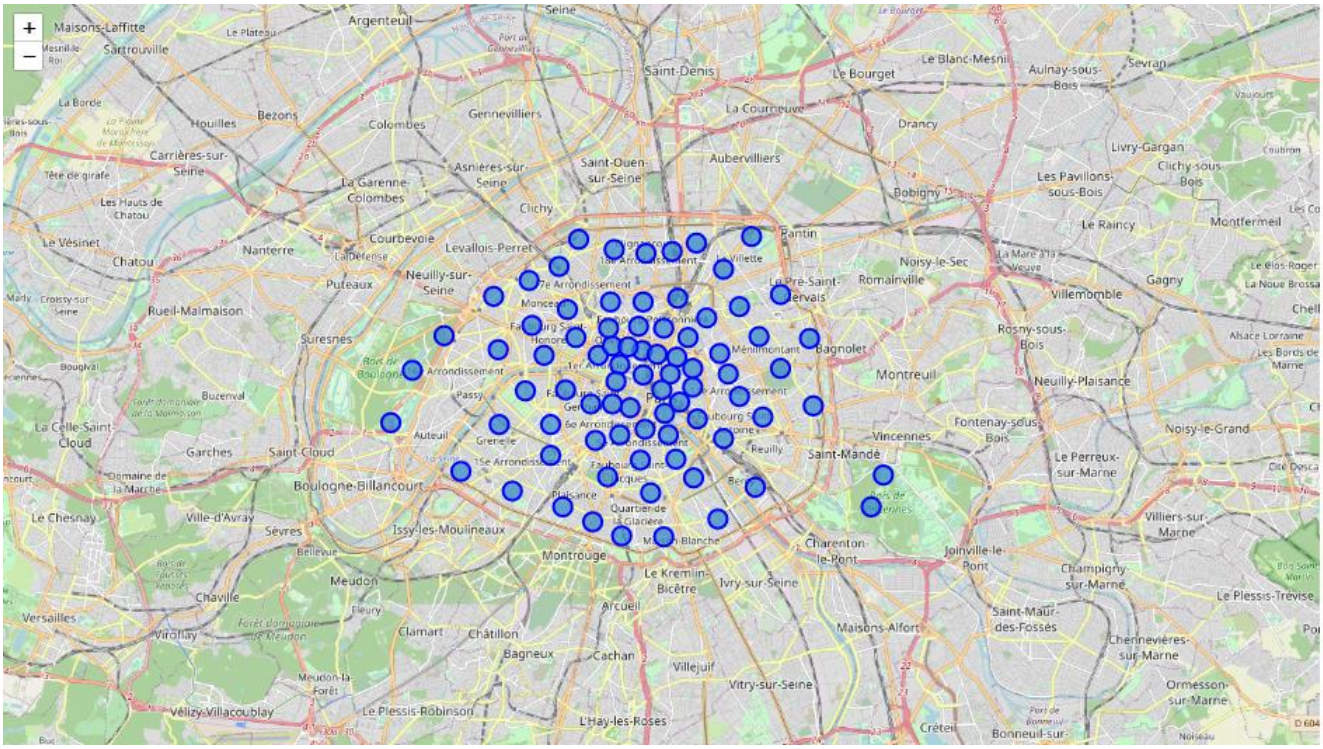
Using Geolocator we find Paris geographical coordinates and using Folium we create a map of Paris.

```
address = 'Paris, FR'

geolocator = Nominatim(user_agent="ny_explorer")
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude
print('The geograpical coordinate of Paris are {}, {}.'.format(latitude, longitude))
```

And we create a map of Paris using latitude and longitude values

Always using Folium we add Neigborhoods mark ups on the map.



Now we are ready to use Foursquare to explore venues in our nighborhoods

We access foursquare:

```
CLIENT_ID = 'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX' # your Foursquare ID
CLIENT_SECRET = 'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX' # your Foursquare Secret
VERSION = '20200605' # Foursquare API version
LIMIT = 100 # A default Foursquare API limit value
print('Your credentials:')
print('CLIENT_ID: ' + CLIENT_ID)
print('CLIENT_SECRET:' + CLIENT_SECRET)
```

We can print the name and the coordinates of the first neighborhood in the table.

```
neighborhood_latitude = df_final.loc[0, 'Latitude'] # neighborhood latitude value
neighborhood_longitude = df_final.loc[0, 'Longitude'] # neighborhood longitude value

neighborhood_name = df_final.loc[0, 'Neighborhood'] # neighborhood name

print('Latitude and longitude values of {} are {}, {}.'.format(neighborhood_name,
                                                               neighborhood_latitude,
                                                               neighborhood_longitude))
```

```
Latitude and longitude values of Archives are 48.859192412700004, 2.3632050573299996.
```

We get the top 100 venues that are in Archives within a radius of 500 meters.

```
LIMIT = 100 # limit of number of venues returned by Foursquare API
radius = 500 # define radius
url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={},{}&radius={}&limit={}'.format(
    CLIENT_ID,
    CLIENT_SECRET,
    VERSION,
    neighborhood_latitude,
    neighborhood_longitude,
    radius,
    LIMIT)

# get the result to a json file
results = requests.get(url).json()
results
```

We can analyse this first neighborhood getting familiar.

The most important item is categories and we can check the total number of venues by categories in the neighborhood Archives. Looking at the number of Ice cream shops as well.

```
nearby_venus_g=nearby_venues.groupby('categories').count()
#nearby_venus_s=nearby_venus_g.sort_values('name')
nearby_venus_s=nearby_venus_g.sort_values(by=['name'], ascending=False)
nearby_venus_s.head()
```

categories	name	lat	lng
French Restaurant	10	10	10
Hotel	6	6	6
Clothing Store	5	5	5
Bookstore	4	4	4
Art Gallery	3	3	3
Italian Restaurant	3	3	3
Bistro	3	3	3
Burger Joint	3	3	3
Plaza	3	3	3
Cocktail Bar	3	3	3

```
nearby_venus_s.tail(10)
```

categories	name	lat	lng
Creperie	1	1	1
Israeli Restaurant	1	1	1
Ice Cream Shop	1	1	1
Dessert Shop	1	1	1
History Museum	1	1	1
Diner	1	1	1
Health Food Store	1	1	1
Art Museum	1	1	1
Gastropub	1	1	1
Wine Bar	1	1	1

Now we Explore all Neighborhoods in Paris

```
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']

    return(nearby_venues)
```

Write the code to run the above function on each neighborhood and create a new dataframe called Paris_venues.

```
Paris_venues = getNearbyVenues(names=df_final['Neighborhood'],
                               latitudes=df_final['Latitude'],
                               longitudes=df_final['Longitude']
                              )
```

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Archives	48.859192	2.363205	Galerie Thaddaeus Ropac	48.860582	2.363539	Art Gallery
1	Archives	48.859192	2.363205	LECLAIREUR Sévigné	48.857341	2.363273	Clothing Store
2	Archives	48.859192	2.363205	Breizh Café	48.860613	2.361804	Creperie
3	Archives	48.859192	2.363205	Galerie Perrotin	48.860726	2.365168	Art Gallery
4	Archives	48.859192	2.363205	Musée Picasso	48.859905	2.362286	Art Museum

Check how many venues were returned for each neighborhood.

Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
Vivienne	100	100	100	100	100	100
Gros-Caillou	100	100	100	100	100	100
Monnaie	100	100	100	100	100	100
Archives	100	100	100	100	100	100
Gaillon	100	100	100	100	100	100
...	...	...	...	...	...	...
Amérique	13	13	13	13	13	13
Porte-Dauphine	10	10	10	10	10	10
Muette	8	8	8	8	8	8
Bel-Air	6	6	6	6	6	6
Picpus	2	2	2	2	2	2

For the scope of the project we continue the analysis with the neighborhoods with more than 80 venues retrieved.

So We create a new dataframe including only neigborhoods with more than 80 venues

```
Paris_venues = Paris_venues.merge(Paris_venues_tot_new, on='Neighborhood', how='inner')
#Paris_venues.to_excel(r'C:\Users\Paold\Export.xlsx', index = False)

Paris_venues.head()
```

Now we analyse each Neighborhood.

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

# add neighborhood column back to dataframe
Paris_onehot['Neighborhood'] = Paris_venues['Neighborhood']

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

	Neighborhood	Accessories Store	Adult Boutique	Afghan Restaurant	African Restaurant	Alsatian Restaurant	American Restaurant	Arcade	Arepa Restaurant	Argentinian Restaurant	...	Vegetarian / Vegan Restaurant	Venezuelan Restaurant	Video Game Store	Vietnamese Restaurant	Wine Bar	Wine Shop	Women's Store	Yoga Studio	Zoo	Zoo Exhibit
0	Archives	0.0	0.0	0.0	0.0	0.0	0.00	0.0	0.0	0.00	...	0.000000	0.0	0.0	0.00	0.010000	0.00	0.0	0.0	0.0	0.0
1	Arsenal	0.0	0.0	0.0	0.0	0.0	0.00	0.0	0.0	0.00	...	0.027397	0.0	0.0	0.00	0.013699	0.00	0.0	0.0	0.0	0.0
2	Arts-et-Métiers	0.0	0.0	0.0	0.0	0.0	0.00	0.0	0.0	0.01	...	0.010000	0.0	0.0	0.03	0.050000	0.01	0.0	0.0	0.0	0.0
3	Batignolles	0.0	0.0	0.0	0.0	0.0	0.01	0.0	0.0	0.00	...	0.010000	0.0	0.0	0.00	0.010000	0.00	0.0	0.0	0.0	0.0
4	Bercy	0.0	0.0	0.0	0.0	0.0	0.00	0.0	0.0	0.00	...	0.000000	0.0	0.0	0.00	0.025974	0.00	0.0	0.0	0.0	0.0

5 rows × 286 columns

We filter venues categories more important for us using presence of theatres museums clothing stores and others as indication of traffic + the Ice cream Shop Category.

```
Paris_grouped1= Paris_grouped[['Neighborhood','Clothing Store','Cosmetics Shop','Record Shop','Perfume Shop','Shopping Mall','Furniture / Home Store','Souvenir Shop','Garden','Ice Cream Shop']]
Paris_grouped1.head()
```

	Neighborhood	Clothing Store	Cosmetics Shop	Record Shop	Perfume Shop	Shopping Mall	Furniture / Home Store	Souvenir Shop	Garden	Ice Cream Shop
0	Archives	0.05	0.01	0.0	0.00	0.0	0.000000	0.0	0.02	0.01
1	Arts-et-Métiers	0.00	0.00	0.0	0.00	0.0	0.010000	0.0	0.00	0.00
2	Batignolles	0.00	0.00	0.0	0.00	0.0	0.010101	0.0	0.00	0.00
3	Bonne-Nouvelle	0.02	0.01	0.0	0.01	0.0	0.000000	0.0	0.00	0.01
4	Chaillot	0.00	0.01	0.0	0.00	0.0	0.000000	0.0	0.00	0.00

Print each neighborhood along with the top 5 most common venues inside the chosen categories.

```
num_top_venues = 5

for hood in Paris_grouped1['Neighborhood']:
    print("----"+hood+"----")
    temp = Paris_grouped1[Paris_grouped1['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')
```

And we put that into a pandas 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
0	Archives	Clothing Store	Garden	Ice Cream Shop	Cosmetics Shop	Souvenir Shop	Furniture / Home Store	Shopping Mall
1	Arts-et-Métiers	Furniture / Home Store	Ice Cream Shop	Garden	Souvenir Shop	Shopping Mall	Perfume Shop	Record Shop
2	Batignolles	Furniture / Home Store	Ice Cream Shop	Garden	Souvenir Shop	Shopping Mall	Perfume Shop	Record Shop
3	Bonne-Nouvelle	Clothing Store	Ice Cream Shop	Perfume Shop	Cosmetics Shop	Garden	Souvenir Shop	Furniture / Home Store
4	Chaillot	Cosmetics Shop	Ice Cream Shop	Garden	Souvenir Shop	Furniture / Home Store	Shopping Mall	Perfume Shop

```
# get counts of Clothing Store in each Neighborhood
```

NOW we Cluster Neighborhoods.

```
kclusters = 4
Paris_grouped_clustering = Paris_grouped1.drop('Neighborhood', 1)
kmeans = KMeans(n_clusters=kclusters, random_state=0).fit(Paris_grouped_clustering)
kmeans.labels_[0:10]
```

```
array([0, 2, 2, 2, 2, 0, 0, 2, 2, 0])
```

```
Paris_merged["Cluster Labels"] = pd.to_numeric(Paris_merged["Cluster Labels"])
Paris_merged['Cluster Labels'] = Paris_merged['Cluster Labels'].astype(int)
Paris_merged.head()
```

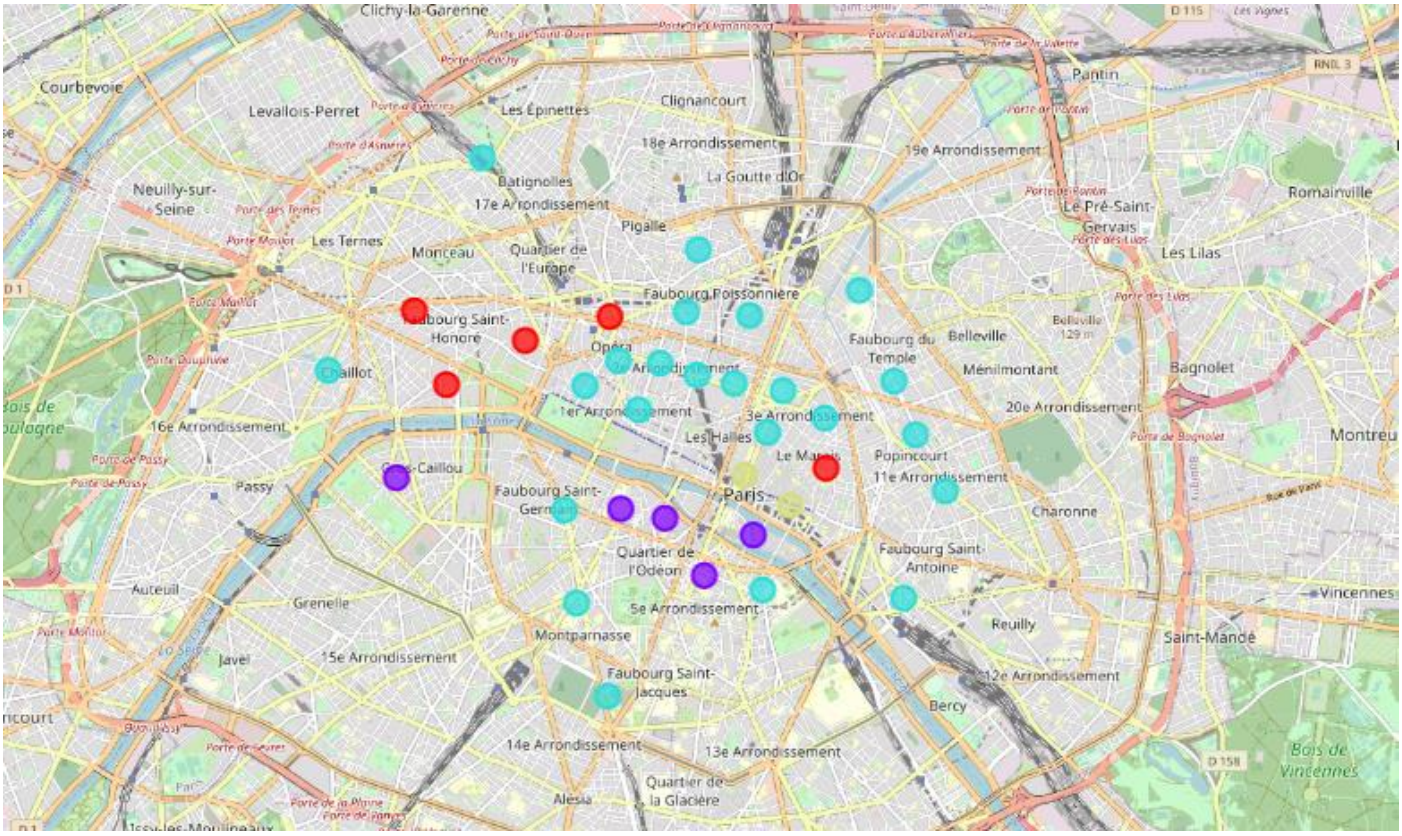
	Borough N	Borough	Neighborhood	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue
0	3	Temple	Archives	48.859192	2.363205	1	Clothing Store	Bookstore	Ice Cream Shop	Cosmetics Shop	Shopping Mall
1	3	Temple	Enfants-Rouges	48.863887	2.363123	1	Bookstore	Clothing Store	Ice Cream Shop	Shopping Mall	Boutique
2	3	Temple	Arts-et-Métiers	48.866470	2.357083	0	Boutique	Bookstore	Ice Cream Shop	Shopping Mall	Perfume Shop
3	3	Temple	Sainte-Avoie	48.862557	2.354852	0	Ice Cream Shop	Bookstore	Cosmetics Shop	Clothing Store	Shopping Mall
4	11	Popincourt	Saint-Ambroise	48.862345	2.376118	0	Bookstore	Ice Cream Shop	Shopping Mall	Boutique	Perfume Shop

And we create a Map.

```
# create map
map_clusters = folium.Map(location=[latitude, longitude], zoom_start=13)

# 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(Paris_merged['Latitude'], Paris_merged['Longitude'], Paris_merged['Neighborhood'], Paris_merged['Cluster Labels']):
    label = folium.Popup(str(poi) + ' Cluster ' + str(cluster), parse_html=True)
    folium.CircleMarker(
        [lat, lon],
        radius=10,
        popup=label,
        color=rainbow[cluster-1],
        fill=True,
        fill_color=rainbow[cluster-1],
        fill_opacity=0.7).add_to(map_clusters)
map_clusters
```



Examine clusters

Cluster 1

```
Paris_merged.loc[Paris_merged['Cluster Labels'] == 0, Paris_merged.columns[[1,2] + list(range(5, Paris_merged.shape[1]))]]
```

	Borough	Neighborhood	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
0	Temple	Archives	0	Clothing Store	Garden	Ice Cream Shop	Cosmetics Shop	Souvenir Shop	Furniture / Home Store	Shopping Mall
12	Élysée	Faubourg-du-Roule	0	Cosmetics Shop	Clothing Store	Ice Cream Shop	Garden	Souvenir Shop	Furniture / Home Store	Shopping Mall
13	Élysée	Champs-Elysées	0	Garden	Clothing Store	Cosmetics Shop	Furniture / Home Store	Ice Cream Shop	Souvenir Shop	Shopping Mall
14	Élysée	Madeleine	0	Clothing Store	Garden	Furniture / Home Store	Shopping Mall	Cosmetics Shop	Ice Cream Shop	Souvenir Shop
31	Opéra	Chaussée-d'Antin	0	Clothing Store	Garden	Souvenir Shop	Cosmetics Shop	Ice Cream Shop	Furniture / Home Store	Shopping Mall

We exclude immediately Cluster 2 and 4 where existing Ice cream shops are dominating all the categories of shops indicating traffic.

As an example in **Saint-Gervais** we have 4 ice cream shops which is the second category after French Restaurant.

[91]:

Venue Category	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude
French Restaurant	11	11	11	11	11	11
Ice Cream Shop	5	5	5	5	5	5
Clothing Store	5	5	5	5	5	5
Hotel	4	4	4	4	4	4
Pastry Shop	4	4	4	4	4	4

We exclude as well Cluster 3 and We decide to focus on the Cluster Number 1 for further Analysis

Let' analyse first neigborhood in cluster 1 **Archives**

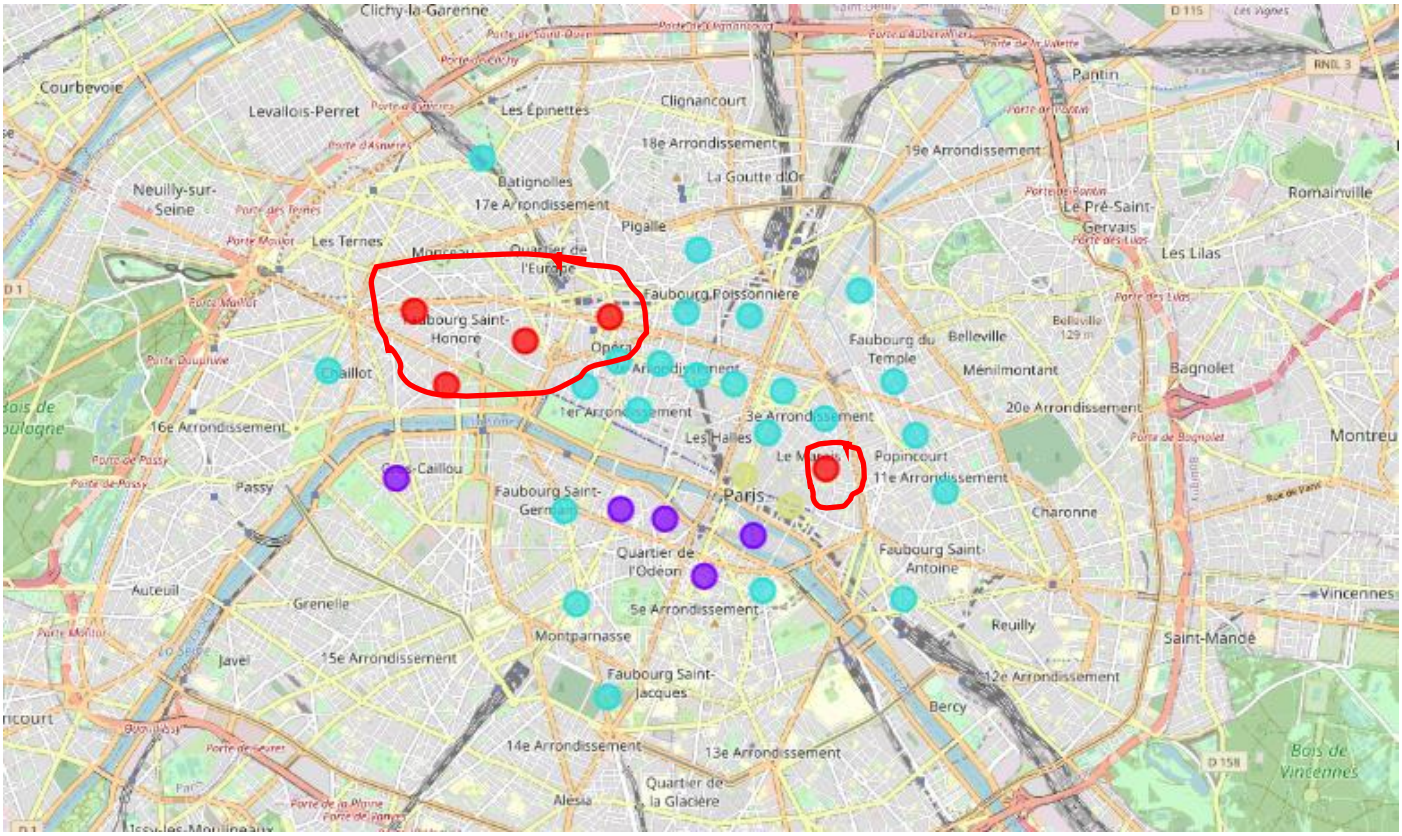
Venue Category	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude
French Restaurant	10	10	10	10	10	10
Hotel	6	6	6	6	6	6
Clothing Store	5	5	5	5	5	5
Bookstore	4	4	4	4	4	4
Art Gallery	3	3	3	3	3	3
Italian Restaurant	3	3	3	3	3	3
Bistro	3	3	3	3	3	3
Burger Joint	3	3	3	3	3	3
Plaza	3	3	3	3	3	3
Cocktail Bar	3	3	3	3	3	3
Pizza Place	2	2	2	2	2	2
Historic Site	2	2	2	2	2	2
Japanese Restaurant	2	2	2	2	2	2
Pastry Shop	2	2	2	2	2	2
Restaurant	2	2	2	2	2	2
Sandwich Place	2	2	2	2	2	2

Archives looks a good potential for the scope of this analysis since it contains several venues indicating an high level of traffic only ONE ice cream Shop

Same for **Faubourg-du-Roul** with a lot of venues indicating high traffic and one only ice cream shop¶

As well **Madeline** is a potential good place with many venues indicating potential traffic and No Ice cream shops already opened.

The cluster is therefore representative for the scope of this analysis.



4. RESULTS

After clustering the data of the respective neighborhoods, and analysing data, we have found several Neighborhoods in cluster 1 where it looks valuable to open an Italian Ice cream Shop like Archives, Enfants-Rouges Madeleine, Champs-Elysée.

5. OBSERVATION AND RECCOMENDATION

The analysis should be continued to extract a certain number of locations and the final selection should be done and validated using additional data like the price per square meter located which gives as well the idea of the cost related to the opening of a shop in the specific Area.

The completeness of Foursquare data for Paris is progressing but not yet at the level of US, Canada or certain countries in Far east.

6. CONCLUSION

Despite it can be improved, the analysis clearly shows that there is room for Italian Ice cram shops in Paris in areas where the level of traffic is more than sufficient to ensure a good level of profitability.