1 Introduction and Discussion of the Business Objective and Problem

Locations for New Fashion Stores in High Traffic Areas in Paris France

The Task At Hand

A digitally native vertical fashion retailer, with a substantial e-commerce footprint, has begun the rollout of brick and mortar stores as part of their omnichannel retail strategy. After rolling out stores in a few select cities by guessing where the best locations were to open, as part of their store expansion for Paris they've decided to be more informed and selective, and take the time to do some research.

I've been given the exciting task of assisting them to make data-driven decisions on the new locations that are most suitable for their new stores in Paris. This will be a major part of their decision-making process, the other being on the ground qualitative analysis of districts once this data and report are reviewed and studied.

The fashion brand is not what is considered high-end, they are positioned in upper end of the fast fashion market. As such, they do not seek stores in the premium upmarket strips like Avenue Montaigne, but rather, in high traffic areas where consumers go for shopping, restaurants and entertainment. Foursquare data will be very helpful in making data-driven decisions about the best of those areas.

Criteria

Qualitative data from another retailer that they know, suggests that the best locations to open new fashion retail stores may not only be where other clothing is located. This data strongly suggests that the best places are in fact areas that are near ***French Restaurants, Cafés and Wine Bars***. Parisians are very social people that frequent these place often, so opening new stores in these locations is becoming popular.

The analysis and recommendations for new store locations will focus on general districts with these establishments, not on specific store addresses. Narrowing down the best district options derived from analysis allows for either further research to be conducted, advising agents of the chosen district, or on the ground searching for specific sites by the company's personnel.

Why Data?

Without leveraging data to make decisions about new store locations, the company could spend countless hours walking around districts, consulting many real estate agents with their own district biases, and end up opening in yet another location that is not ideal.Data will provide better answers and better solutions to their task at hand.

OUTCOME / TARGET AUDIENCE

The goal is to identify the best districts - *Arrondissements* - to open new stores as part of the company's plan. The results will be translated to management in a simple form that will convey the data-driven analysis for the best locations to open stores.

2 The Data Science Workflow

Data Requirements

The main districts in Paris are divided into 20 *Arrondissements Municipaux* (administrative districts), shortened to *arrondissements*.

The data regarding the districts in Paris needs to be researched and a suitable useable source identified. If it is found but is not in a useable form, data wrangling and cleaning will have to be performed.

The cleansed data will then be used alongside Foursquare data, which is readily available. Foursquare location data will be leveraged to explore or compare districts around Paris, identifying the high traffic areas where consumers go for shopping, dining and entertainment - the areas where the fashion brand are most interested in opening new stores.

The Data Science Workflow for Part 1 & 2 includes the following:

- **Outline the initial data that is required:**
  - District data for Paris including names, location data if available, and any other details required.

- **Obtain the Data:**
  - Research and find suitable sources for the district data for Paris.
  - Access and explore the data to determine if it can be manipulated for our purposes.

- **Initial Data Wrangling and Cleaning:**
  - Clean the data and convert to a useable form as a dataframe.

The Data Science Workflow for parts 3 & 4 includes:
- **Data Analysis and Location Data:**
  - Foursquare location data will be leveraged to explore or compare districts around Paris.
  - Data manipulation and analysis to derive subsets of the initial data.
  - Identifying the high traffic areas using data visualisation and tatistical nalysis.
- **Visualization:**
  - Analysis and plotting visualizations.
  - Data visualization using various mapping libraries.
- **Discussion and Conclusions:**
  - Recomendations and results based on the data analysis.
  - Discussion of any limitations and how the results can be used, and any conclusions that can be drawn.

Import the Required Libraries

```
# Import libraries
import numpy as np # library to handle data in a vectorized manner
import json # library to handle JSON files
import pandas as pd
!conda install -c conda-forge geopy --yes
from geopy.geocoders import Nominatim # convert an address into lati
tude and longitude values
import requests # library to handle requests
from pandas.io.json import json_normalize # tranform JSON file into
a pandas dataframe
# Matplotlib and associated plotting modules
import matplotlib.pyplot as plt
import matplotlib.cm as cm
import matplotlib.colors as colors
from bs4 import BeautifulSoup
# Import k-means from clustering stage
from sklearn.cluster import KMeans
!conda install -c conda-forge folium=0.5.0 --yes
import folium # map rendering library
print('Libraries imported.')
```

Data Research and Preparation

Import the Paris District Data

Arrondissements Municipaux for Paris CSV (administrative districts)

Paris is divided into 20 Arrondissements Municipaux (or administrative districts), shortened to just arrondissements. They and normally referenced by the arrondissement number rather than a name.

Data for the arrondissements is necessary to select the most suitable of these areas for new stores.

Initially looking to get this data by scraping the relevent Wikipedia page (https://en.wikipedia.org/wiki/Arrondissements_of_Paris), fortunately, after much research, this data is available on the web and can be manipulated and cleansed to provide a meaningful dataset to use.

Data from Open|DATA
France: https://opendata.paris.fr/explore/dataset/arrondissements/table/?dataChart

Also available from
Opendatasoft: https://data.opendatasoft.com/explore/dataset/arrondissements%40parisdata/export/

```
# Download the dataset and read it into a pandas dataframe.
# The Arrondissements dataset was downloaded from Paris|DATA:  https
://opendata.paris.fr/explore/dataset/arrondissements/table/?dataChar
t
# Then placed on the GitHub repo for the project.
# https://raw.githubusercontent.com/AR-data-science/Coursera_Capston
e/master/Arrondissements_.csv

paris = pd.read_csv('https://raw.githubusercontent.com/AR-data-scien
ce/Coursera_Capstone/master/Arrondissements_.csv')
paris
```

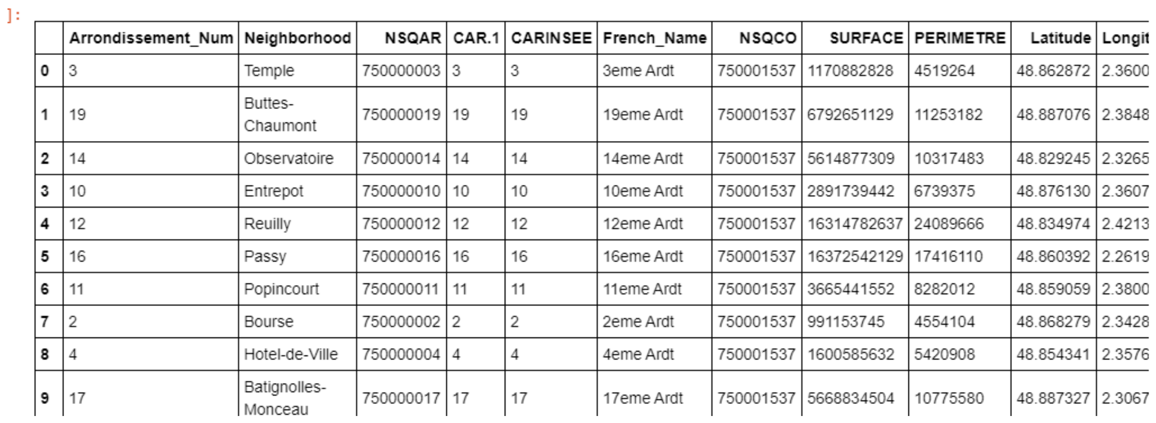
:

	CAR	NAME	NSQAR	CAR.1	CARINSEE	LAR	NSQCO	SURFACE	PERIMETRE	Geometry_X	Geometry_Y
0	3	Temple	750000003	3	3	3eme Ardt	750001537	1170882828	4519264	48.862872	2.360001
1	19	Buttes-Chaumont	750000019	19	19	19eme Ardt	750001537	6792651129	11253182	48.887076	2.384821
2	14	Observatoire	750000014	14	14	14eme Ardt	750001537	5614877309	10317483	48.829245	2.326542
3	10	Entrepot	750000010	10	10	10eme Ardt	750001537	2891739442	6739375	48.876130	2.360728
4	12	Reuilly	750000012	12	12	12eme Ardt	750001537	16314782637	24089666	48.834974	2.421325
5	16	Passy	750000016	16	16	16eme Ardt	750001537	16372542129	17416110	48.860392	2.261971
6	11	Popincourt	750000011	11	11	11eme Ardt	750001537	3665441552	8282012	48.859059	2.380058
7	2	Bourse	750000002	2	2	2eme Ardt	750001537	991153745	4554104	48.868279	2.342803
8	4	Hotel-de-Ville	750000004	4	4	4eme Ardt	750001537	1600585632	5420908	48.854341	2.357630
9	17	Batignolles-Monceau	750000017	17	17	17eme Ardt	750001537	5668834504	10775580	48.887327	2.306777

Exploring, Wrangling and Cleaning the Data

Rename the columns 'Geometry_X' and 'Geometry_Y', "CAR' to 'Arrondissement' etc...

```
# Rename the necessary columns 'Geometry_X and Geometry_Y' etc...

# District : name of the central District for the Arrondissement
# Arrondissement : the Arrondissement or district number which is used
to identify it
# Arrondissement_Fr : the descriptive French label for each Arrondissem
ent

paris.rename(columns={'NAME': 'Neighborhood ', 'CAR': 'Arrondissement_N
um', 'Geometry_X': 'Latitude', 'Geometry_Y': 'Longitude',  'LAR': 'Fren
ch_Name'}, inplace=True)
paris
```

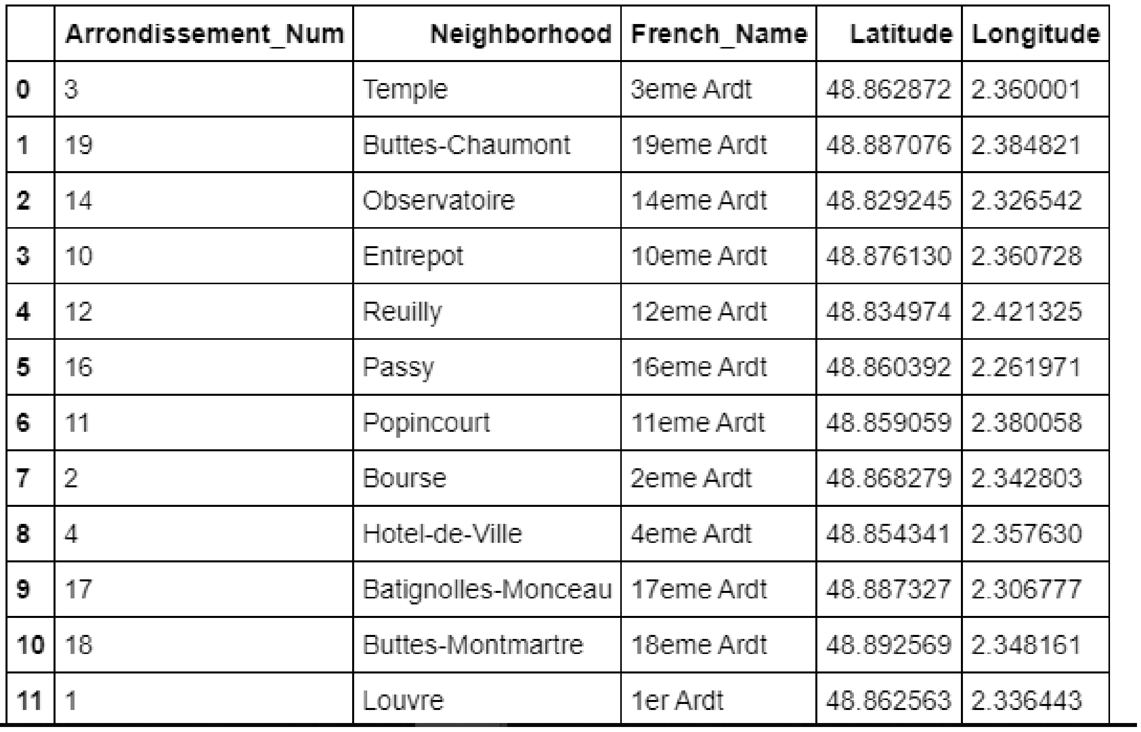


]:

	Arrondissement_Num	Neighborhood	NSQAR	CAR.1	CARINSEE	French_Name	NSQCO	SURFACE	PERIMETRE	Latitude	Longit
0	3	Temple	750000003	3	3	3eme Ardt	750001537	1170882828	4519264	48.862872	2.3600
1	19	Buttes-Chaumont	750000019	19	19	19eme Ardt	750001537	6792651129	11253182	48.887076	2.3848
2	14	Observatoire	750000014	14	14	14eme Ardt	750001537	5614877309	10317483	48.829245	2.3265
3	10	Entrepot	750000010	10	10	10eme Ardt	750001537	2891739442	6739375	48.876130	2.3607
4	12	Reuilly	750000012	12	12	12eme Ardt	750001537	16314782637	24089666	48.834974	2.4213
5	16	Passy	750000016	16	16	16eme Ardt	750001537	16372542129	17416110	48.860392	2.2619
6	11	Popincourt	750000011	11	11	11eme Ardt	750001537	3665441552	8282012	48.859059	2.3800
7	2	Bourse	750000002	2	2	2eme Ardt	750001537	991153745	4554104	48.868279	2.3428
8	4	Hotel-de-Ville	750000004	4	4	4eme Ardt	750001537	1600585632	5420908	48.854341	2.3576
9	17	Batignolles-Monceau	750000017	17	17	17eme Ardt	750001537	5668834504	10775580	48.887327	2.3067

Clean up the dataset to remove unnecessary columns¶

```
# Clean up the dataset to remove unnecessary columns.
# Some of the columns are for mapping software - not required here.

paris.drop(['NSQAR','CAR.1','CARINSEE','NSQCO','SURFACE', 'PERIMETRE
' ], axis=1, inplace=True)
paris
```

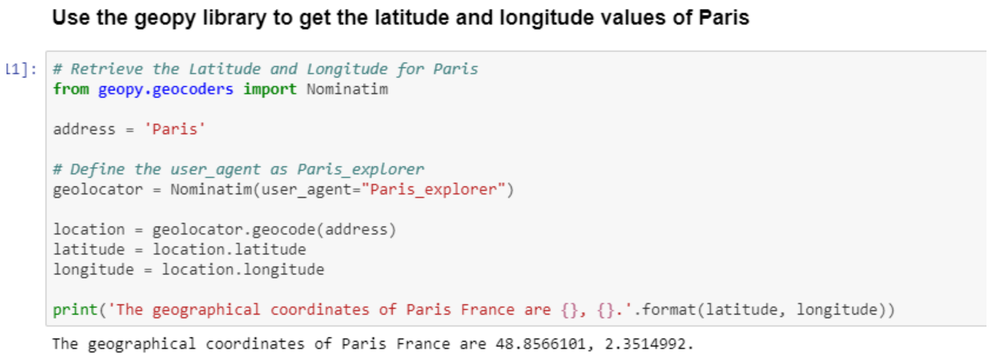


	Arrondissement_Num	Neighborhood	French_Name	Latitude	Longitude
0	3	Temple	3eme Ardt	48.862872	2.360001
1	19	Buttes-Chaumont	19eme Ardt	48.887076	2.384821
2	14	Observatoire	14eme Ardt	48.829245	2.326542
3	10	Entrepot	10eme Ardt	48.876130	2.360728
4	12	Reuilly	12eme Ardt	48.834974	2.421325
5	16	Passy	16eme Ardt	48.860392	2.261971
6	11	Popincourt	11eme Ardt	48.859059	2.380058
7	2	Bourse	2eme Ardt	48.868279	2.342803
8	4	Hotel-de-Ville	4eme Ardt	48.854341	2.357630
9	17	Batignolles-Monceau	17eme Ardt	48.887327	2.306777
10	18	Buttes-Montmartre	18eme Ardt	48.892569	2.348161
11	1	Louvre	1er Ardt	48.862563	2.336443

3 Methodology and Exploratory Data Analysis

The Data Science Workflow for parts 3 & 4 includes:

- **Data Analysis and Location Data:**
  - Foursquare location data will be leveraged to explore or compare districts around Paris.
  - Data manipulation and analysis to derive subsets of the initial data.
  - Identifying the high traffic areas using data visualization and statistical analysis.
- **Visualization:**
  - Analysis and plotting visualizations.
  - Data visualization using various mapping libraries.
- **Discussion and Conclusions:**
  - Recomendations and results based on the data analysis.
  - Discussion of any limitations and how the results can be used, and any conclusions that can be drawn.

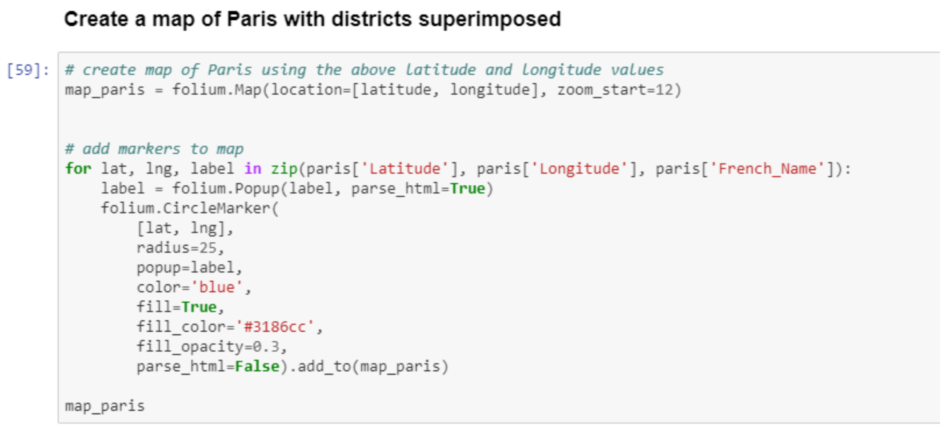


Use the geopy library to get the latitude and longitude values of Paris

1]:

```
# Retrieve the Latitude and Longitude for Paris
from geopy.geocoders import Nominatim

address = 'Paris'

# Define the user_agent as Paris_explorer
geolocator = Nominatim(user_agent="Paris_explorer")

location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude

print('The geographical coordinates of Paris France are {}, {}.'.format(latitude, longitude))
```

```
The geographical coordinates of Paris France are 48.8566101, 2.3514992.
```



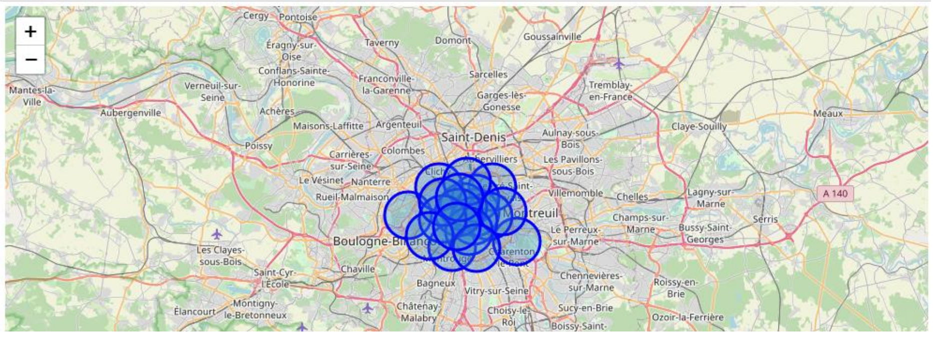
Create a map of Paris with districts superimposed

[59]:

```
# create map of Paris using the above Latitude and Longitude values
map_paris = folium.Map(location=[latitude, longitude], zoom_start=12)


# add markers to map
for lat, lng, label in zip(paris['Latitude'], paris['Longitude'], paris['French_Name']):
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius=25,
        popup=label,
        color='blue',
        fill=True,
        fill_color='#3186cc',
        fill_opacity=0.3,
        parse_html=False).add_to(map_paris)

map_paris
```

```
map_paris
```

|:



Use the Foursquare API to explore the Arrondissements of Paris (Neighborhoods)

Define Foursquare Credentials and Version

```
CLIENT_ID = 'RU3Y3XCL1D3X4IFWHEPI3VYYAEEGSWMVQTWP2PHZ1DEL1E2R' # your Foursquare ID
CLIENT_SECRET = 'GMMPCVSMWDJDXSOTEO22G3FN4H2BUFSZO05SYEGRGW4N5AKL' # your Foursquare Secret
VERSION = '20180605' # Foursquare API version

print('Your credentails:')
print('CLIENT_ID: ' + CLIENT_ID)
print('CLIENT_SECRET:' + CLIENT_SECRET)
```

```
Your credentails:
CLIENT_ID: RU3Y3XCL1D3X4IFWHEPI3VYYAEEGSWMVQTWP2PHZ1DEL1E2R
CLIENT_SECRET:GMMPCVSMWDJDXSOTEO22G3FN4H2BUFSZO05SYEGRGW4N5AKL
```

Exploratory data analysis

Explore the first district in our dataframe to become familiar with the data (use the French descriptive arrondissement name)

```
# Explore the first Neighborhood in our dataframe.
# Get the Neighborhood's French name.

paris.loc[0, 'French_Name']
paris.loc[0, 'French_Name']
```

```
'3eme Ardt'
```

The first arrondissement is identified as *3eme Ardt*

```
# Get the Neighborhood's latitude and longitude values.

neighborhood_latitude = paris.loc[0, 'Latitude'] # Neighborhood latitude value
neighborhood_longitude = paris.loc[0, 'Longitude'] # Neighborhood longitude value

neighborhood_name = paris.loc[0, 'French_Name'] # Neighborhood name

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

```
Latitude and longitude values of the neighborhood 3eme Ardt are 48.86287238, 2.3600009859999997.
```

Get the top 100 venues that are in the neighborhood *3eme Ardt* 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)
url # displays the URL
```

```
'https://api.foursquare.com/v2/venues/explore?&client_id=RU3Y3XCL1D3X4IFWHEPI3VYYAEEGSWMVQTWP2PHZ1DEL1E2R&client_secret=GMMPCVS
MWDJDXSOTEO22G3FN4H2BUFSZO05SYEGRGW4N5AKL&v=20180605&ll=48.86287238,2.3600009859999997&radius=500&limit=100'
```

```
In [17]: # Send the GET request and examine the resutls

         results = requests.get(url).json()
         results
```

```
Out[17]: {'meta': {'code': 200, 'requestId': '5d80b83b8ad62e002c9f7de6'},
          'response': {'suggestedFilters': {'header': 'Tap to show:',
            'filters': [{'name': 'Open now', 'key': 'openNow'}]},
           'headerLocation': 'Enfants-Rouges',
           'headerFullLocation': 'Enfants-Rouges, Paris',
           'headerLocationGranularity': 'neighborhood',
           'totalResults': 137,
           'suggestedBounds': {'ne': {'lat': 48.8673723845, 'lng': 2.3668285468065267},
            'sw': {'lat': 48.8583723755, 'lng': 2.353173425193473}},
           'groups': [{'type': 'Recommended Places',
             'name': 'recommended',
             'items': [{'reasons': {'count': 0,
                'items': [{'summary': 'This spot is popular',
                  'type': 'general',
                  'reasonName': 'globalInteractionReason'}]},
               'venue': {'id': '4d974096a2c654814aa6d353',
                'name': 'Mmmozza',
                'location': {'address': '57 rue de Bretagne',
                 'lat': 48.86391016055883,
                 'lng': 2.360590696334839
```

Define the function that extracts the category of the venue

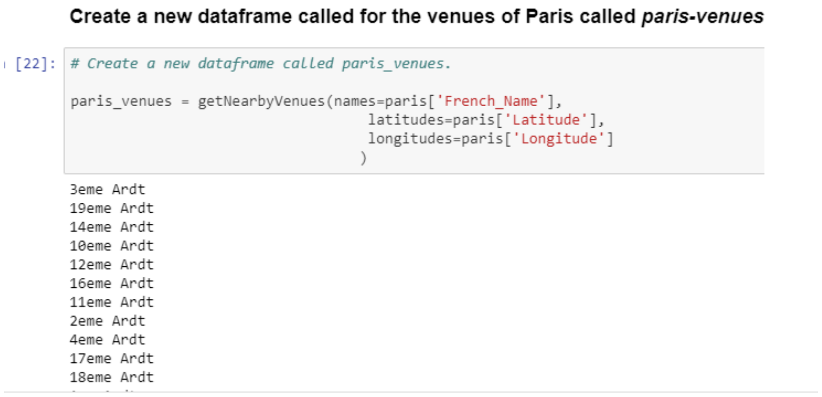
```
In [18]: # define a function that extracts the category of the venue

         def get_category_type(row):
             try:
                 categories_list = row['categories']
             except:
                 categories_list = row['venue.categories']

             if len(categories_list) == 0:
                 return None
             else:
                 return categories_list[0]['name']
```

Structure the json file into a pandas dataframe

```
|: # clean the json and structure it into a pandas dataframe.

   venues = results['response']['groups'][0]['items']

   nearby_venues = json_normalize(venues) # flatten JSON

   # filter columns
   filtered_columns = ['venue.name', 'venue.categories', 'venue.location.lat', 'venue.location.lng']
   nearby_venues =nearby_venues.loc[:, filtered_columns]

   # filter the category for each row
   nearby_venues['venue.categories'] = nearby_venues.apply(get_category_type, axis=1)

   # clean columns
   nearby_venues.columns = [col.split(".")[-1] for col in nearby_venues.columns]

   nearby_venues.head(20)
```

)ut[19]:

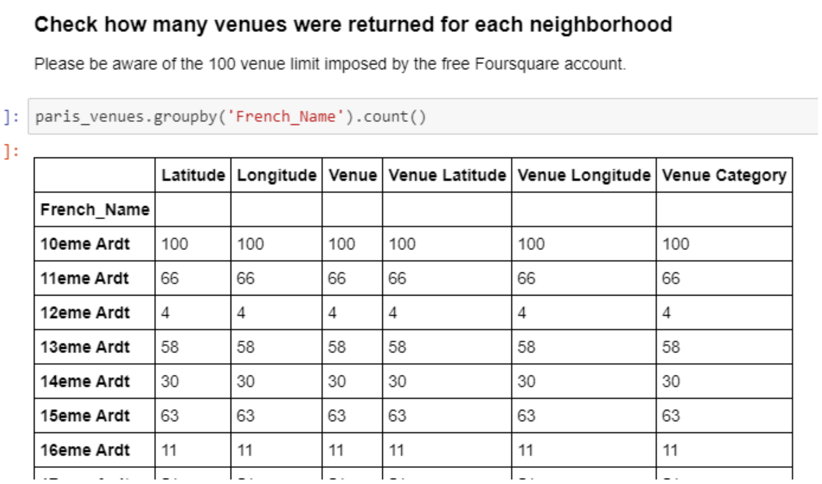
	name	categories	lat	lng
0	Mmmozza	Sandwich Place	48.863910	2.360591
1	Square du Temple	Park	48.864475	2.360816
2	Marché des Enfants Rouges	Farmers Market	48.862806	2.361996
3	Chez Alain Miam Miam	Sandwich Place	48.862781	2.362064
4	Chez Alain Miam Miam	Sandwich Place	48.862369	2.361950
5	Fromagerie Jouannault	Cheese Shop	48.862947	2.362530
6	Les Enfants Rouges	Wine Bar	48.863013	2.361260
7	Okomusu	Okonomiyaki Restaurant	48.861453	2.360879
8	Hôtel Jules & Jim	Hotel	48.863496	2.357395
9	Musée de la Chasse et de la Nature	Museum	48.861507	2.358624
10	Bontemps	Dessert Shop	48.863956	2.360725



Create a new dataframe called for the venues of Paris called *paris-venues*

```
[22]: # Create a new dataframe called paris_venues.

paris_venues = getNearbyVenues(names=paris['French_Name'],
                                   latitudes=paris['Latitude'],
                                   longitudes=paris['Longitude']
                                  )
```

```
3eme Ardt
19eme Ardt
14eme Ardt
10eme Ardt
12eme Ardt
16eme Ardt
11eme Ardt
2eme Ardt
4eme Ardt
17eme Ardt
18eme Ardt
```



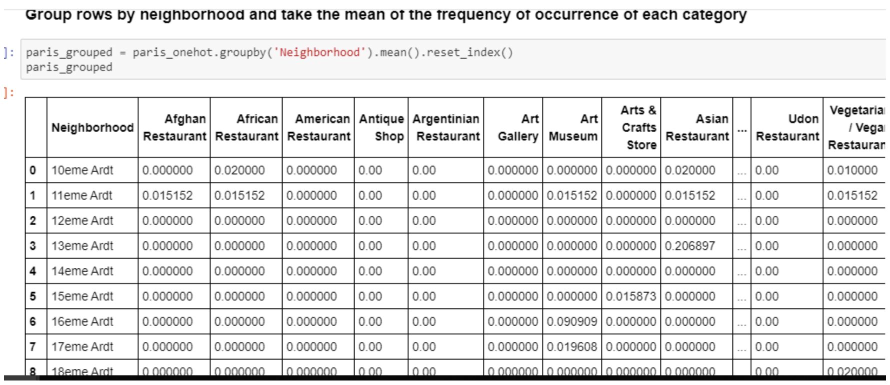
Check how many venues were returned for each neighborhood

Please be aware of the 100 venue limit imposed by the free Foursquare account.

```
]: paris_venues.groupby('French_Name').count()
```

]:

	Latitude	Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
French_Name						
10eme Ardt	100	100	100	100	100	100
11eme Ardt	66	66	66	66	66	66
12eme Ardt	4	4	4	4	4	4
13eme Ardt	58	58	58	58	58	58
14eme Ardt	30	30	30	30	30	30
15eme Ardt	63	63	63	63	63	63
16eme Ardt	11	11	11	11	11	11

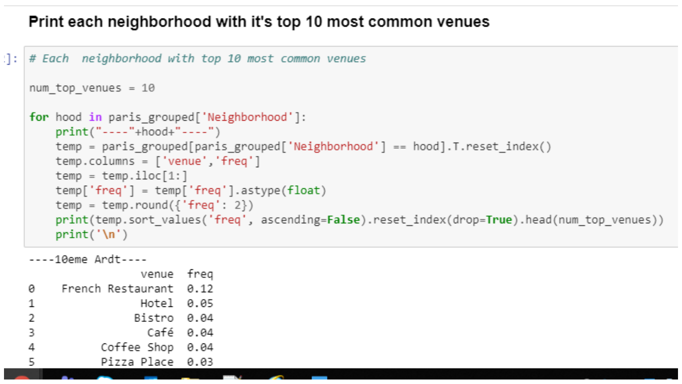


Group rows by neighborhood and take the mean of the frequency of occurrence of each category

```
]: paris_grouped = paris_onehot.groupby('Neighborhood').mean().reset_index()
paris_grouped
```

]:

	Neighborhood	Afghan Restaurant	African Restaurant	American Restaurant	Antique Shop	Argentinian Restaurant	Art Gallery	Art Museum	Arts & Crafts Store	Asian Restaurant	...	Udon Restaurant	Vegetaria / Vega Restauran
0	10eme Ardt	0.000000	0.020000	0.000000	0.00	0.00	0.000000	0.000000	0.000000	0.020000	...	0.00	0.010000
1	11eme Ardt	0.015152	0.015152	0.000000	0.00	0.00	0.000000	0.015152	0.000000	0.015152	...	0.00	0.015152
2	12eme Ardt	0.000000	0.000000	0.000000	0.00	0.00	0.000000	0.000000	0.000000	0.000000	...	0.00	0.000000
3	13eme Ardt	0.000000	0.000000	0.000000	0.00	0.00	0.000000	0.000000	0.000000	0.206897	...	0.00	0.000000
4	14eme Ardt	0.000000	0.000000	0.000000	0.00	0.00	0.000000	0.000000	0.000000	0.000000	...	0.00	0.000000
5	15eme Ardt	0.000000	0.000000	0.000000	0.00	0.00	0.000000	0.000000	0.015873	0.000000	...	0.00	0.000000
6	16eme Ardt	0.000000	0.000000	0.000000	0.00	0.00	0.000000	0.090909	0.000000	0.000000	...	0.00	0.000000
7	17eme Ardt	0.000000	0.000000	0.000000	0.00	0.00	0.000000	0.019608	0.000000	0.000000	...	0.00	0.000000
8	18eme Ardt	0.000000	0.000000	0.000000	0.00	0.00	0.000000	0.000000	0.000000	0.000000		0.00	0.020000



Print each neighborhood with it's top 10 most common venues

```
]: # Each  neighborhood with top 10 most common venues

num_top_venues = 10

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

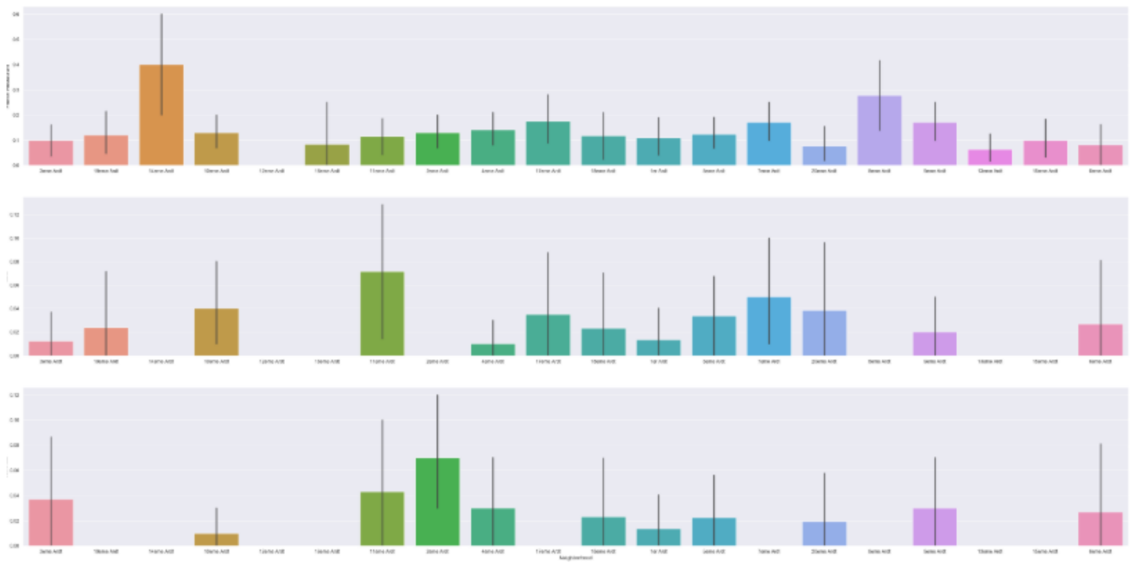
```
----10eme Ardt----
                venue  freq
0    French Restaurant  0.12
1                Hotel  0.05
2               Bistro  0.04
3                 Café  0.04
4          Coffee Shop  0.04
5          Pizza Place  0.03
```

The business types criteria specified by the client! *'French Restaurants', 'Cafés' and 'Wine Bars'*¶

Let's look at their frequency of occurance for all the Paris neighborhoods, isolating the categorical venues

These are the venue types that the client wants to have an abundant density of in the ideal store locations. I've used a violin plot from the seaborn library - it is a great way to visualise frequency distribution datasets, they display a density estimation of the underlying distribution.

Frequency Distribution for top 3 venue categories for each neighborhood



The Neighborhoods

So as we can see from the analysis there are 8 neighborhoods to open new stores - according to the criteria that they have the 3 specified venues in a great frequency (*French Restaurants, Cafés and Wine Bars).* They are as follows:
Neighborhoods

- **3eme Ardt**
- **10eme Ardt**
- **11eme Ardt**
- **4eme Ardt**
- **18eme Ardt**
- **18eme Ardt**
- **5eme Ardt**
- **9eme Ardt**
- **6eme Ardt**

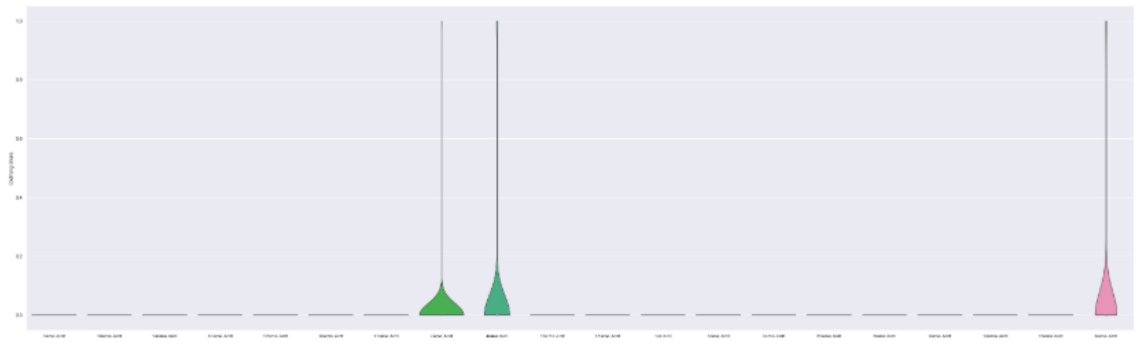
Let's take this further with some exploration and Inferential Analysis

We have the 8 neighborhoods that all include the venue category criteria.
But if we included the 'Clothing_Store" venue category into the analysis, then we might be able to make some inferences based on the data, and domain knowledge of marketing and the industry, to focus the list.

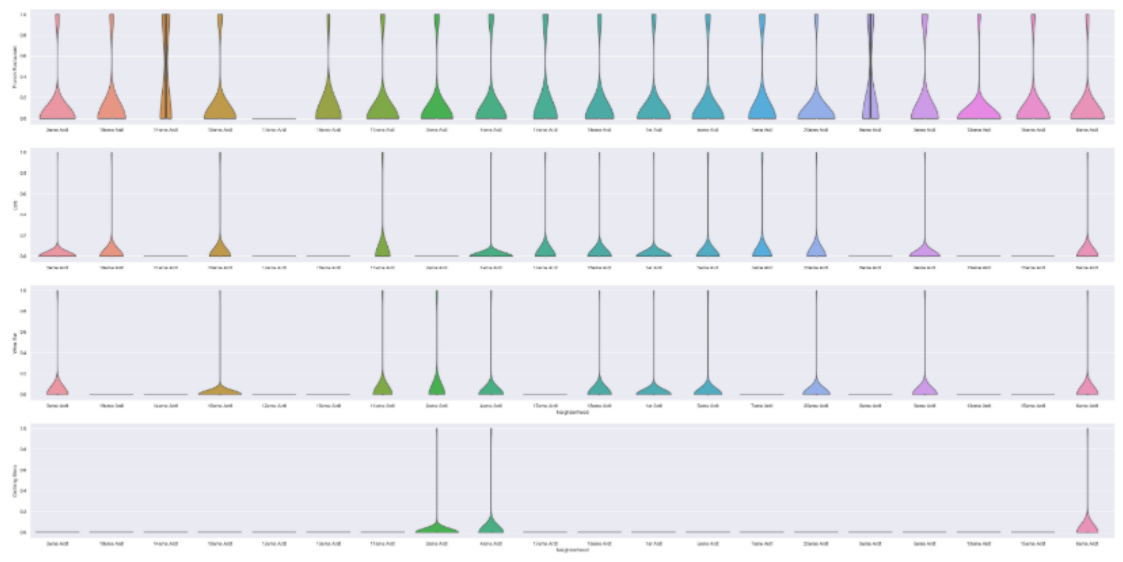
Let's look at the venue category - *''Clothing Store*'

```
# Add the Clothing_Store to explore this category
import seaborn as sns

fig = plt.figure(figsize=(50,15))
sns.set(font_scale=1.1)

ax = plt.subplot(1,1,1)
sns.violinplot(x="Neighborhood", y="Clothing Store", data=paris_onehot, cut=0);
plt.xlabel("")

ax.text(1.0, 1.1, 'Frequency of Clothing stores for each neighborhood', fontsize=60)
plt.savefig ("Distribution_Frequency_Clothing_Venues.png", dpi=240)
plt.show()
```

Frequency of Clothing stores for each neighborhood



Frequency distribution for the top 3 venue categories for each neighborhood (includes clothing)



4 Inferences and Discussion

Chosen Neighborhoods - Results

Inferential analysis using the data, as well as domain knowledge of retail and marketing, allow the list to be focused to just 3 neighborhoods from the previous 8.

The reasoning being that if the 3 criteria have been met - identifying neighborhoods that are lively with Restaurants, Cafés and Wine Bars - adding Clothing Stores into the mix of stores in the area is a significant bonus. Having some of the same category of stores in the same area - especially in fashion retail - is very desirable as a retailer.

So we can increase the criteria to include *Restaurants, Cafés, Wine Bars and Clothing Stores* - which narrows down and focuses the suggested districts for new stores to be located, and at the same time provides better locations for the brand.

So the final 3 prospective neighborhoods for new store locations are where 4 criteria are met:

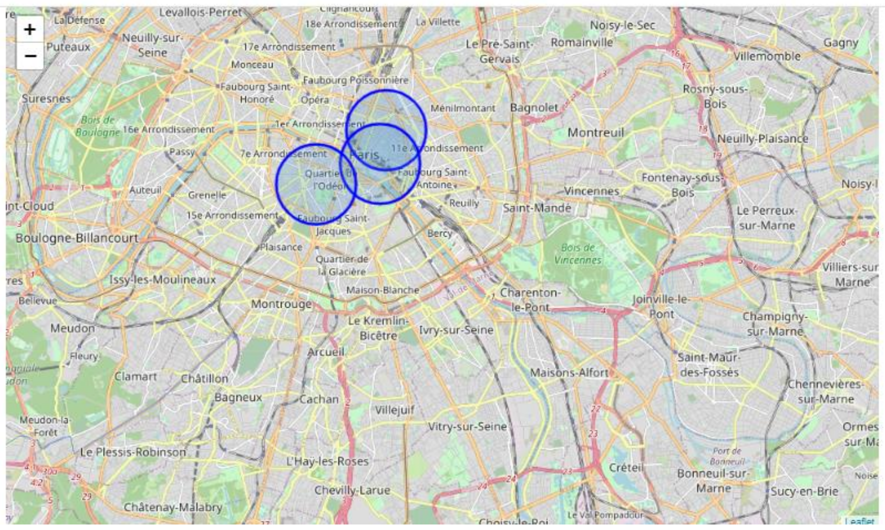
- 3eme Ardt : Arrondissement 3, Temple

- 4eme Ardt : Arrondissement 4, Hotel-de-Ville

- 6eme Ardt : Arrondissement 6, Luxembourg¶

Let's look at the 3 districts on a Paris map

Where are our chosen districts? Let's visualise them on a map of Paris



Observations

I guess it's not a surprise that these districts are all very centrally located in the circular arrangement of Paris's arrondissements. Locations fitting the criteria for popular venues would normally be in central locations in many cities of the world.

From this visualization it is clear that on a practical level, with no data to base decisions on, the circle of the 20 districts is very large, and researching and then visiting them all would be a daunting and time-consuming task. We have narrowed the search area down significantly from 20 potential districts to 3 that should suit the client's retail business.

Inferences

We have made inferences from the data in making the location recommendations, but that is exactly the point. There is no right or wrong answer or conclusion for the task at hand. The job of data analysis here is to steer a course for the location selection of new stores (i) to meet the criteria of being in neighborhoods that are lively with abundant leisure venues, and (ii) to narrow the search down to just a few of the main areas that are best suited to match the criteria.

Conclusions

There are many ways this analysis could have been performed based on different methodology and perhaps different data sources. I chose the method I selected as it was a straightforward way to narrow down the options, not complicating what is actually simple in many ways – meeting the criteria for the surrounding venues, and in my case, domain knowledge I have on the subject. I originally intended to use the clustering algorithms to cluster the data, but as it progressed it became obvious that this only complicated the task at hand. The analysis and results are not an end point, but rather a starting point that will guide the next part of the process to find specific store locations. The next part will involve domain knowledge of the industry, and perhaps, of the city itself. But the data analysis and resulting recommendations have greatly narrowed down the best district options based on data and what we can infer from it.

Without leveraging data to make focused decisions, the process could have been drawn out and resulted in new stores opening in sub-standard areas for this retailer. Data has helped to provide a better strategy and way forward, these data-driven decisions will lead to a better solution in the end.