clustring_cities_morocco

June 4, 2020

1 Morocco Cities Clustering

1.1 Introduction and description

The goal of this project it to **cluster** the cities of my country **Morocco**. The clustering algorithm will use **geographical** data such as *population*, *number of hotels* and *number and type of industries**. This clustering could serve for serveral purposes:

- Say I had to move from my current city, I would like to choose another city which is similar to my current city.
- For a foreign tourist, Say you visited a city *A* and you liked it but didn't like city *B*. In future visit to Morocco, you'll would like to avoid all the cities in the *B* cluster and try to discover more cities in the *A* cluster. Better application for this would be a **Recommender system** but can also use *clustering*.

1.2 Data description

The first step for the data preparation is to get a list of **cities** in Morocco. Luckily, this wikipedia page has a list of all the imporant cities in a table.

[3]:
```
import pandas as pd
import matplotlib.pyplot as plt
import folium
from geopy import Nominatim
import requests
import numpy as np
```

[2]:
```
morocco_cities_link = "https://en.wikipedia.org/wiki/List_of_cities_in_Morocco"
```

[3]:
```
morocco_cities = requests.get(morocco_cities_link).content
morocco_cities = pd.read_html(morocco_cities)[0]
```

[4]:
```
morocco_cities.head()
```

[4]:
```
   Rank           City  Population(2014 census)[5][6]  \
0     1  Casablanca[b]                        3359818
1     2         Fez[c]                        1112072
2     3     Tangier[d]                         947952
```

```
3     4   Marrakesh[e]                          928850
4     5        Salé[f]                          890403

                      Region
0          Casablanca-Settat
1                Fès-Meknès
2  Tanger-Tetouan-Al Hoceima
3             Marrakesh-Safi
4         Rabat-Salé-Kénitra
```

[5]:
```
#rename the long population column
morocco_cities.rename(columns = {'Population(2014 census)[5][6]':'Population'},␣
↪inplace=True)
#set index as the City
morocco_cities.set_index(morocco_cities['City'],inplace=True)

#dropping the rank colum
morocco_cities.drop(['City','Rank'], axis=1, inplace=True)
morocco_cities.head()
```

[5]:
```
               Population                      Region
City
Casablanca[b]     3359818          Casablanca-Settat
Fez[c]            1112072                Fès-Meknès
Tangier[d]         947952  Tanger-Tetouan-Al Hoceima
Marrakesh[e]       928850             Marrakesh-Safi
Salé[f]            890403         Rabat-Salé-Kénitra
```

[6]:
```
morocco_cities.index = morocco_cities.index.map(lambda x : x.split(sep = '[')[0])
morocco_cities.head()
```

[6]:
```
            Population                      Region
City
Casablanca     3359818          Casablanca-Settat
Fez            1112072                Fès-Meknès
Tangier         947952  Tanger-Tetouan-Al Hoceima
Marrakesh       928850             Marrakesh-Safi
Salé            890403         Rabat-Salé-Kénitra
```

The table is *bare bone* for now as we only have the

- City name: as index
- Population:
- Region

As a first step, we will add the **goelocalisation** positions

[7]:
```
def gps_coordinates(description):
    """
    get the gps (latitude, longitude)
    from the description using the foursquare agent
    """
    geolocator = Nominatim(user_agent='foursquare_agent')

    #getting the location
    while True:
        location = geolocator.geocode(description)
        if location is not None:
            break

    return location.latitude, location.longitude
```

[8]:
```
morocco_cities['latitude'] = np.zeros(len(morocco_cities))
morocco_cities['longitude'] = np.zeros(len(morocco_cities))

for i in range(len(morocco_cities)):
    #region
    region = morocco_cities.index[i]
    #coordinate
    lat, long = gps_coordinates(region)

    #
    morocco_cities.iloc[i,2] = lat
    morocco_cities.iloc[i,3] = long
```

[9]:
```
morocco_cities.head()
```

[9]:

City	Population	Region	latitude	longitude
Casablanca	3359818	Casablanca-Settat	33.595063	-7.618777
Fez	1112072	Fès-Meknès	34.034653	-5.016193
Tangier	947952	Tanger-Tetouan-Al Hoceima	35.777103	-5.803792
Marrakesh	928850	Marrakesh-Safi	31.625826	-7.989161
Salé	890403	Rabat-Salé-Kénitra	34.044889	-6.814017

2 showing the cities

[10]:
```
morocco_position = gps_coordinates('Morocco')
```

[11]:
```
morocco = folium.Map(location = morocco_position, zoom_start=6)
for city, lat, long in zip(morocco_cities.index, morocco_cities.latitude,␣
 ↪morocco_cities.longitude):
    folium.features.CircleMarker(
```

```
        [lat, long],
        fill_color='blue',
        radius=3,
        fill=True
    ).add_to(morocco)
morocco
```

```
        ---------------------------------------------

        AttributeError Traceback (most recent call last)

        <ipython-input-11-7cf56e3ec4f1> in <module>
          1 morocco = folium.Map(location = morocco_position, zoom_start=6)
          2 for city, lat, long in zip(morocco_cities.index, morocco_cities.
 ↪latitude, morocco_cities.longitude):
    ----> 3     folium.features.CircleMarker(
          4         [lat, long],
          5         fill_color='blue',


        AttributeError: module 'folium.features' has no attribute 'CircleMarker'
```

Now we need more information about each city. We will use the *foursqure* API to select some useful informations. In order to get all the possible venue **categories**, I will save search results of each **query** on a panda Dataframe. The categories will be decided then by their frequency

-

[12]:
```
CLIENT_ID = '4VCZ1KOOIRKHSLQTMSPJML3GXASFVKA35MAV5VZGGHISQZE5' # your Foursquare␣
 ↪ID
CLIENT_SECRET = 'CXOCOCPCNAK5K52M3QAATVPA1HFENUWGAENFYT3EXSDDXOPN' # 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: 4VCZ1KOOIRKHSLQTMSPJML3GXASFVKA35MAV5VZGGHISQZE5
CLIENT_SECRET:CXOCOCPCNAK5K52M3QAATVPA1HFENUWGAENFYT3EXSDDXOPN
```

[13]:
```
# 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']
```

[14]:
```
def get_venues_for_city(city_latitude, city_longitude, city_name, radius=500,␣
↪limit = 500):
    """
    Function to execute the foursquare request and get all the venues for a given
    city characterised by it latitude and longitude
    """
    url = 'https://api.foursquare.com/v2/venues/explore?
↪&client_id={}&client_secret={}&v={}&ll={},{}&radius={}&limit={}'.format(
    CLIENT_ID,
    CLIENT_SECRET,
    VERSION,
    city_latitude,
    city_longitude,
    radius,
    limit)

    #getting the request as a json data
    result = requests.get(url).json()

    #gettingt the venues
    venues = result['response']['groups'][0]['items']

    # normalize the json data
    nearby_venues = pd.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.to_csv(f"cities_data/{city_name}.csv")
    nearby_venues.head()
```

```
    return nearby_venues

casablanca = get_venues_for_city(morocco_cities.iloc[0,2], morocco_cities.
 ↪iloc[0,3],'Casablanca')
```

[15]:
```
casablanca.head()
```

[15]:

	name	categories	lat	lng
0	Six PM	Hotel Bar	33.595940	-7.618684
1	Hyatt Regency Casablanca	Hotel	33.596195	-7.618708
2	Casa Jose	Tapas Restaurant	33.597823	-7.615341
3	Le Rouget de l'Isle	French Restaurant	33.592591	-7.622857
4	Le Riad Restaurant	Moroccan Restaurant	33.593936	-7.614676

Now I will do the same for all the cities to get an idea on all the *categories*

[16]:
```
for row in morocco_cities.iterrows():
    city_name = row[0]
    latitude  = row[1]['latitude']
    longitude = row[1]['longitude']
    #some cities produces a problem in request
    print("city name: ", city_name)
    if (city_name not in ['Safi', 'Nador','Fquih Ben Salah','Errachidia','Lqliaa'
                          ,'Taroudant','Tan-Tan','Souk El Arbaa',
                         'Lahraouyine','Drargua']):
        get_venues_for_city(latitude, longitude, city_name)
```

```
city name:  Casablanca
city name:  Fez
city name:  Tangier
city name:  Marrakesh
city name:  Salé
city name:  Meknes
city name:  Rabat
city name:  Oujda
city name:  Kenitra
city name:  Agadir
city name:  Tetouan
city name:  Temara
city name:  Safi
city name:  Mohammedia
city name:  Khouribga
city name:  El Jadida
city name:  Beni Mellal
city name:  Aït Melloul
city name:  Nador
city name:  Dar Bouazza
city name:  Taza
```

```
city name:  Settat
city name:  Berrechid
city name:  Khemisset
city name:  Inezgane
city name:  Ksar El Kebir
city name:  Larache
city name:  Guelmim
city name:  Khenifra
city name:  Berkane
city name:  Taourirt
city name:  Bouskoura
city name:  Fquih Ben Salah
city name:  Dcheira El Jihadia
city name:  Oued Zem
city name:  El Kelaa Des Sraghna
city name:  Sidi Slimane
city name:  Errachidia
city name:  Guercif
city name:  Oulad Teima
city name:  Ben Guerir
city name:  Tifelt
city name:  Lqliaa
city name:  Taroudant
city name:  Sefrou
city name:  Essaouira
city name:  Fnideq
city name:  Sidi Kacem
city name:  Tiznit
city name:  Tan-Tan
city name:  Ouarzazate
city name:  Souk El Arbaa
city name:  Youssoufia
city name:  Lahraouyine
city name:  Martil
city name:  Ain Harrouda
city name:  Suq as-Sabt Awlad an-Nama
city name:  Skhirat
city name:  Ouazzane
city name:  Benslimane
city name:  Al Hoceima
city name:  Beni Ansar
city name:  M'diq
city name:  Sidi Bennour
city name:  Midelt
city name:  Azrou
city name:  Drargua
```

[17]:
```
#getting the list of all items
from pathlib import Path
#path for the categories
categories_path = Path("cities_data/")

#list of dfs
cities_df = [pd.read_csv(p,index_col=0) for p in categories_path.iterdir()]

#concatenate all the categoeis
df_categories = pd.concat(cities_df, axis=0)
```

[18]:
```
#getting the histogram of the categories
categories = df_categories['categories'].value_counts()
categories
```

[18]:
```
Café                   67
Hotel                  35
Moroccan Restaurant    22
Coffee Shop            20
Diner                  19
                       ..
Garden                  1
Pharmacy                1
Salad Place             1
Lounge                  1
Playground              1
Name: categories, Length: 114, dtype: int64
```

We see the best venues are:

1. Café
2. Hotel
3. Moroccan Restaurant
4. Coffee Shop
5. Diner

Now for each city, we will add the count of thes categories

[19]:
```
selected_categories = ['Café', 'Hotel', 'Moroccan Restaurant', 'Coffee Shop',␣
↪'Diner']
```

[20]:
```
for cat in selected_categories:
    morocco_cities[cat] = np.zeros(len(morocco_cities))
morocco_cities
```

[20]:
```
             Population                     Region   latitude  longitude  \
City
Casablanca      3359818          Casablanca-Settat  33.595063  -7.618777
```

```
Fez              1112072                Fès-Meknès  34.034653  -5.016193
Tangier           947952  Tanger-Tetouan-Al Hoceima  35.777103  -5.803792
Marrakesh         928850             Marrakesh-Safi  31.625826  -7.989161
Salé              890403         Rabat-Salé-Kénitra  34.044889  -6.814017
...                  ...                        ...        ...        ...
M'diq             56227  Tanger-Tetouan-Al Hoceima  35.683360  -5.323216
Sidi Bennour      55815          Casablanca-Settat  32.650779  -8.424209
Midelt            55304             Drâa-Tafilalet  32.680347  -4.739897
Azrou             54350                 Fès-Meknès  33.436117  -5.221913
Drargua           50946                Souss-Massa  30.382030  -9.476421

             Café  Hotel  Moroccan Restaurant  Coffee Shop  Diner
City
Casablanca    0.0    0.0                 0.0          0.0    0.0
Fez           0.0    0.0                 0.0          0.0    0.0
Tangier       0.0    0.0                 0.0          0.0    0.0
Marrakesh     0.0    0.0                 0.0          0.0    0.0
Salé          0.0    0.0                 0.0          0.0    0.0
...           ...    ...                 ...          ...    ...
M'diq         0.0    0.0                 0.0          0.0    0.0
Sidi Bennour  0.0    0.0                 0.0          0.0    0.0
Midelt        0.0    0.0                 0.0          0.0    0.0
Azrou         0.0    0.0                 0.0          0.0    0.0
Drargua       0.0    0.0                 0.0          0.0    0.0

[67 rows x 9 columns]
```

[21]:
```
#now we need the fill all the missing data

#first we need to drop the cities without data (problem with foursquare)
unkown_cities = ['Safi', 'Nador','Fquih Ben Salah','Errachidia','Lqliaa'
                         ,'Taroudant','Tan-Tan','Souk El Arbaa',
                        'Lahraouyine','Drargua']
#morocco_cities.drop(unkown_cities, axis=0,inplace=True)
#loop over all the saved data
for city in categories_path.iterdir():
    #load the categories
    categories = pd.read_csv(city,index_col=0)
    categories = categories['categories'].value_counts()
    #saving the count for each categorie
    for cat in selected_categories:
        if cat in categories.index:
            morocco_cities.loc[city.stem,cat]= categories[cat]
```

[22]:
```
morocco_cities.head()
```

[22]:
```
            Population                      Region   latitude  longitude  Café  \
City
Casablanca     3359818          Casablanca-Settat  33.595063  -7.618777   4.0
Fez            1112072                 Fès-Meknès  34.034653  -5.016193   0.0
Tangier         947952  Tanger-Tetouan-Al Hoceima  35.777103  -5.803792   4.0
Marrakesh       928850             Marrakesh-Safi  31.625826  -7.989161   6.0
Salé            890403         Rabat-Salé-Kénitra  34.044889  -6.814017   1.0

            Hotel  Moroccan Restaurant  Coffee Shop  Diner
City
Casablanca    6.0                  3.0          1.0    1.0
Fez           0.0                  1.0          1.0    0.0
Tangier       3.0                  1.0          0.0    5.0
Marrakesh    11.0                 12.0          0.0    0.0
Salé          0.0                  0.0          0.0    0.0
```

Finally we're getting rid on region as it will not have an effect on the dataset

[23]:
```
#morocco_cities.drop('Region',axis=1, inplace=True)
morocco_cities.dropna(inplace=True)
morocco_cities
```

[23]:
```
              Population                      Region   latitude  longitude  \
City
Casablanca       3359818          Casablanca-Settat  33.595063  -7.618777
Fez              1112072                 Fès-Meknès  34.034653  -5.016193
Tangier           947952  Tanger-Tetouan-Al Hoceima  35.777103  -5.803792
Marrakesh         928850             Marrakesh-Safi  31.625826  -7.989161
Salé              890403         Rabat-Salé-Kénitra  34.044889  -6.814017
...                  ...                        ...        ...        ...
M'diq             56227  Tanger-Tetouan-Al Hoceima  35.683360  -5.323216
Sidi Bennour      55815          Casablanca-Settat  32.650779  -8.424209
Midelt            55304             Drâa-Tafilalet  32.680347  -4.739897
Azrou             54350                 Fès-Meknès  33.436117  -5.221913
Drargua           50946                Souss-Massa  30.382030  -9.476421

              Café  Hotel  Moroccan Restaurant  Coffee Shop  Diner
City
Casablanca     4.0    6.0                  3.0          1.0    1.0
Fez            0.0    0.0                  1.0          1.0    0.0
Tangier        4.0    3.0                  1.0          0.0    5.0
Marrakesh      6.0   11.0                 12.0          0.0    0.0
Salé           1.0    0.0                  0.0          0.0    0.0
...            ...    ...                  ...          ...    ...
M'diq          3.0    0.0                  0.0          0.0    0.0
Sidi Bennour   2.0    0.0                  0.0          1.0    1.0
Midelt         1.0    1.0                  0.0          1.0    0.0
```

```
Azrou          1.0    1.0                 0.0          0.0    0.0
Drargua        0.0    0.0                 0.0          0.0    0.0

[67 rows x 9 columns]
```

3 saving the data

[24]:
```
morocco_cities.to_csv("morocan_cities.csv",index=True)
```

4 Clustering Part 2

Introduction and description

The goal of this project it to cluster the cities of my country Morocco. The clustering algorithm will use geographical data such as population, number of hotels and number and type of industries*. This clustering could serve for serveral purposes:

```
Say I had to move from my current city, I would like to choose another city which is similar to

For a foreign tourist, Say you visited a city
```

and you liked it but didn't like city . In future visit to Morocco, you'll would like to avoid all the cities in the cluster and try to discover more cities in the cluster. Better application for this would be a Recommender system but can also use clustering.

[4]:
```
data = pd.read_csv("morocan_cities.csv" ,index_col=0)
data.head()
```

[4]:
```
            Population                     Region   latitude  longitude  Café  \
City
Casablanca     3359818          Casablanca-Settat  33.595063  -7.618777   4.0
Fez            1112072                Fès-Meknès  34.034653  -5.016193   0.0
Tangier         947952  Tanger-Tetouan-Al Hoceima  35.777103  -5.803792   4.0
Marrakesh       928850             Marrakesh-Safi  31.625826  -7.989161   6.0
Salé            890403         Rabat-Salé-Kénitra  34.044889  -6.814017   1.0

            Hotel  Moroccan Restaurant  Coffee Shop  Diner
City
Casablanca    6.0                  3.0          1.0    1.0
Fez           0.0                  1.0          1.0    0.0
Tangier       3.0                  1.0          0.0    5.0
Marrakesh    11.0                 12.0          0.0    0.0
Salé          0.0                  0.0          0.0    0.0
```

4.1 Exploratory data analysis

Let's plot some variables to see if there is a relation between the variables. But first I need to drop the `Region` field as it not a numeric variable.

[5]:
```
data.drop(['Region'], axis=1, inplace=True)
data.head()
```

[5]:

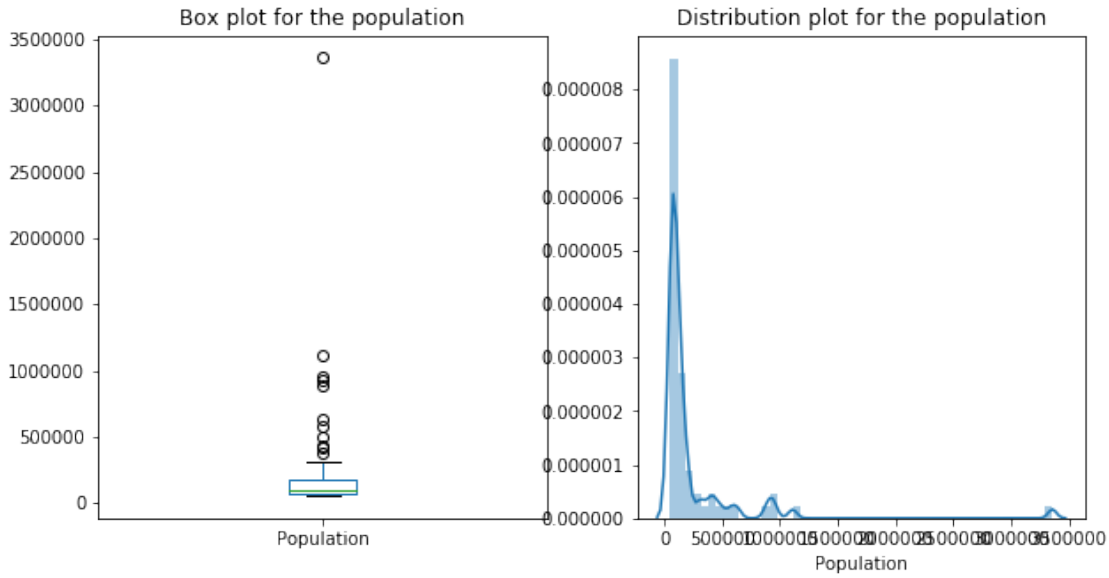
City	Population	latitude	longitude	Café	Hotel	Moroccan Restaurant	Coffee Shop	Diner
Casablanca	3359818	33.595063	-7.618777	4.0	6.0	3.0	1.0	1.0
Fez	1112072	34.034653	-5.016193	0.0	0.0	1.0	1.0	0.0
Tangier	947952	35.777103	-5.803792	4.0	3.0	1.0	0.0	5.0
Marrakesh	928850	31.625826	-7.989161	6.0	11.0	12.0	0.0	0.0
Salé	890403	34.044889	-6.814017	1.0	0.0	0.0	0.0	0.0

4.2 Population

[7]:
```
import seaborn as sns
fig, axs = plt.subplots(1,2,figsize=(10,5))

#box plot
data['Population'].plot(kind='box',ax=axs[0])
sns.distplot(data.Population, ax=axs[1])
axs[0].set_title("Box plot for the population")
axs[1].set_title("Distribution plot for the population")
```

[7]:
```
Text(0.5, 1.0, 'Distribution plot for the population')
```



We can see the data is not well distributed as the city of `Casablanca` has almost all the m

4.3 Features venues variables

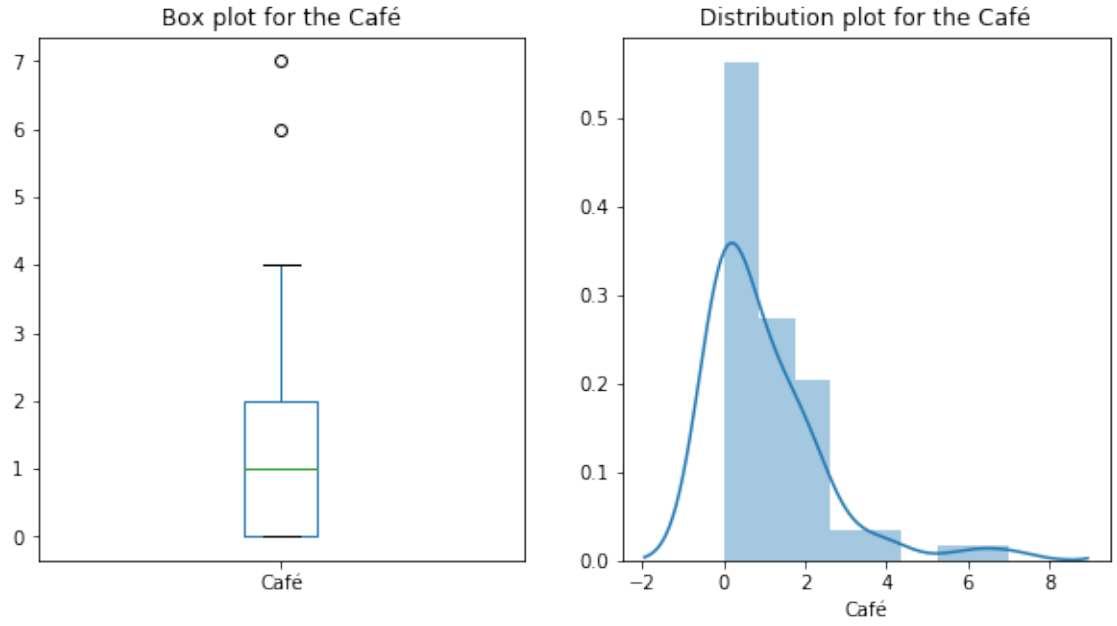
Let's plot and see the values on the venues variables. This is data that was gathered for each town using the **foursquare** api

[13]:
```
venues = ['Café','Hotel', 'Moroccan Restaurant', 'Coffee Shop', 'Diner']
```

[10]:
```
fig, axs = plt.subplots(1,2,figsize=(10,5))

#box plot
data['Café'].plot(kind='box',ax=axs[0])
sns.distplot(data['Café'], ax=axs[1])
axs[0].set_title("Box plot for the Café")
axs[1].set_title("Distribution plot for the Café")
```

[10]:
```
Text(0.5, 1.0, 'Distribution plot for the Café')
```

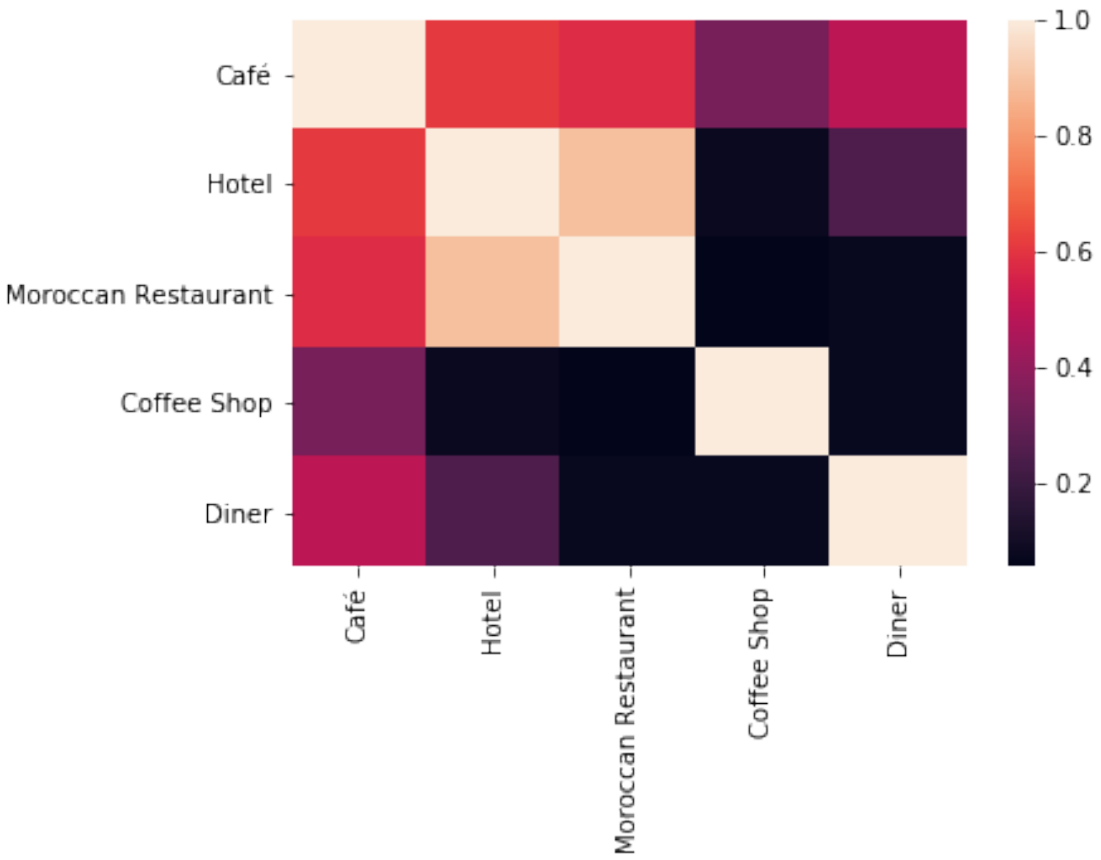


4.4 Correlation

Let plot and search if there is a correlation between the venues

[14]:
```
corr = data[venues].corr()
sns.heatmap(corr)
```

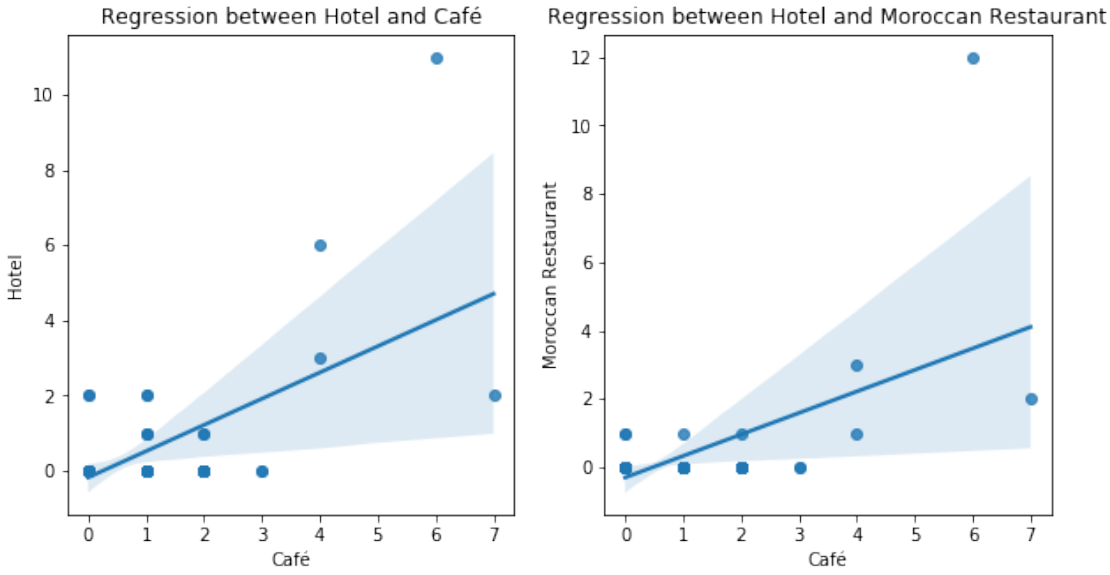
[14]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f413f4ef790>
```



We can remark that there is a strong correlation between Café and Hotel and Moroccan Restaurant to test that let's plot the the regression line between those two variables

[15]:
```
fig, axs = plt.subplots(1,2,figsize=(10,5))
sns.regplot(data['Café'], data['Hotel'],ax=axs[0])
axs[0].set_title('Regression between Hotel and Café')
sns.regplot(data['Café'], data['Moroccan Restaurant'],ax=axs[1])
axs[1].set_title('Regression between Hotel and Moroccan Restaurant')
```

[15]:
```
Text(0.5, 1.0, 'Regression between Hotel and Moroccan Restaurant')
```



4.5 Data preparation for clustering

The first step for the clustering is normalize the data

[ ]:
```
from sklearn.preprocessing import StandardScaler
normalizer = StandardScaler()
```