Evaluation of the neighborhoods in Oslo, Norway in order to develop a business activity.

I. Introduction

The development of businesses brings diverse commercial activities of interest such as capital investment, job creation and increases the activity of the area, often improving the quality of life of the inhabitants in the area. However, it is necessary that the business activity to be developed is economically profitable and sometimes the best option is to choose a neighborhood that already has an activity developed at least to some extent. Therefore, it is important to evaluate the presence of commercial activity in the different neighborhoods of Oslo.

II. Data

To study the feasibility of this project, first of all we evaluated the presence of shops in the area. To do this, we used data from the Wikipedia webpage, which was submitted to a scrapping process.

1. List of boroughs in Oslo

https://en.wikipedia.org/wiki/List_of_boroughs_of_Oslo

2. List of exact locations and commercial activity

From the obtained data GEOLOCATOR is used in order to determine the precise location of each one of the neighborhood, and to extract the sort of businesses present in the vicinity with FOURSQUARE API.

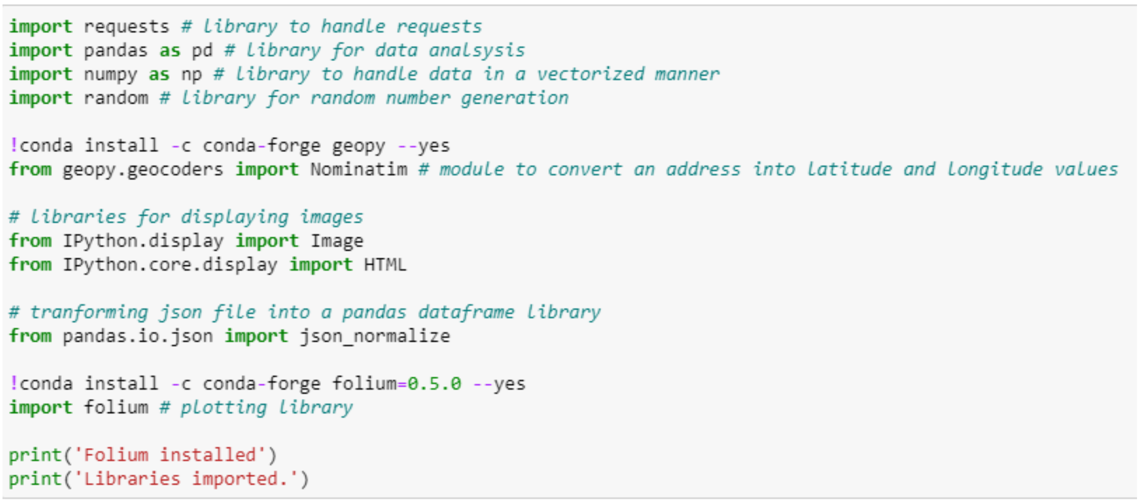
https://developer.foursquare.com/docs

The information was compared in regard to the commercial activity of the different boroughs in Oslo by clustering analysis. From the neighborhoods, we will use Foursquare to obtain information regarding the presence of businesses which will be subjected to a clustering analysis.

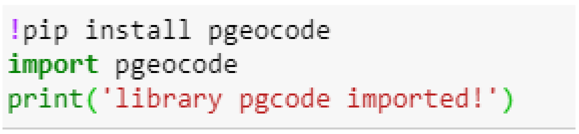
III. METHODOLOGY

First of all we need to scrap the data of interest from a Wikipedia web page.

Import necessary Libraries



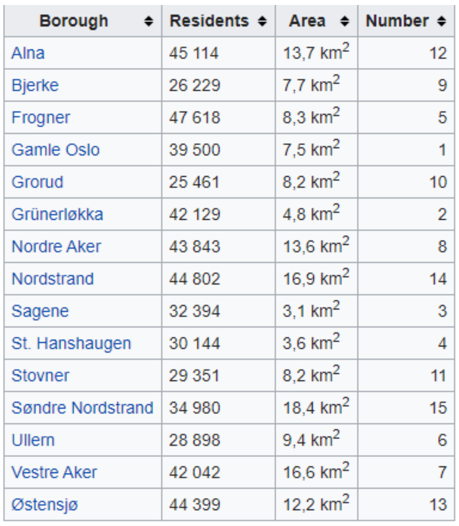
```
import requests # library to handle requests
import pandas as pd # library for data analsysis
import numpy as np # library to handle data in a vectorized manner
import random # library for random number generation

!conda install -c conda-forge geopy --yes
from geopy.geocoders import Nominatim # module to convert an address into latitude and longitude values

# libraries for displaying images
from IPython.display import Image
from IPython.core.display import HTML

# 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

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



```
!pip install pgeocode
import pgeocode
print('library pgcode imported!')
```

Figure 1. Imported libraries

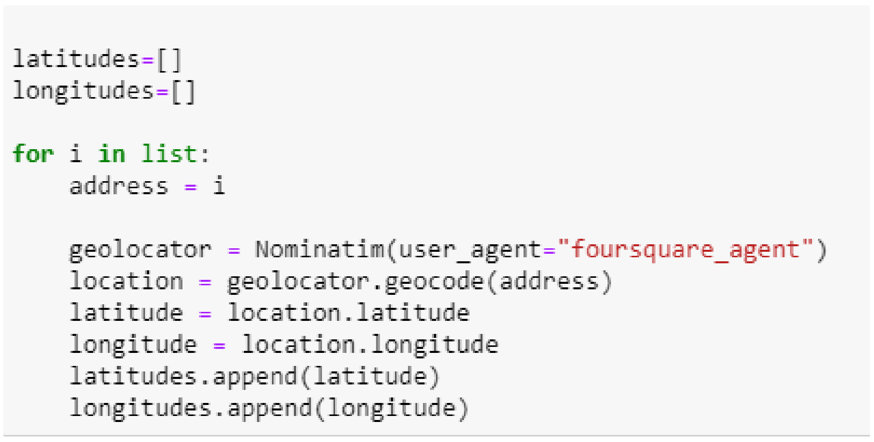
In this study we explored, segmented and clustered the data based on the neighborhood in which they are located.



Borough	Residents	Area	Number
Alna	45 114	13,7 km^2	12
Bjerke	26 229	7,7 km^2	9
Frogner	47 618	8,3 km^2	5
Gamle Oslo	39 500	7,5 km^2	1
Grorud	25 461	8,2 km^2	10
Grünerløkka	42 129	4,8 km^2	2
Nordre Aker	43 843	13,6 km^2	8
Nordstrand	44 802	16,9 km^2	14
Sagene	32 394	3,1 km^2	3
St. Hanshaugen	30 144	3,6 km^2	4
Stovner	29 351	8,2 km^2	11
Søndre Nordstrand	34 980	18,4 km^2	15
Ullern	28 898	9,4 km^2	6
Vestre Aker	42 042	16,6 km^2	7
Østensjø	44 399	12,2 km^2	13

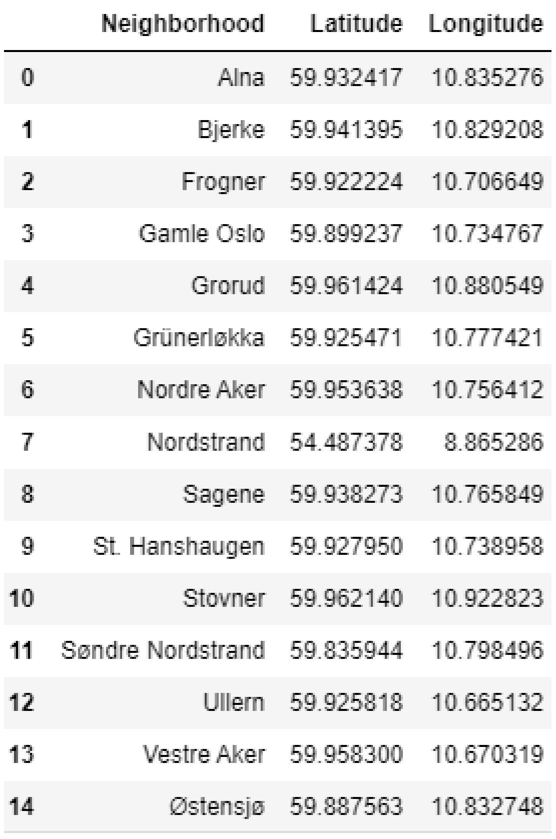
Figure 2. Table in Wikipedia. This table contains the information we need, we are interested in Borough column.

Oslo is organized in 14 neighborhoods some of them in the suburbs of Oslo. But in order to obtain the exact locations we used GEOLOCATOR in a loop for obtaining the longitudes and latitudes.



```
latitudes=[]
longitudes=[]

for i in list:
    address = i

    geolocator = Nominatim(user_agent="foursquare_agent")
    location = geolocator.geocode(address)
    latitude = location.latitude
    longitude = location.longitude
    latitudes.append(latitude)
    longitudes.append(longitude)
```

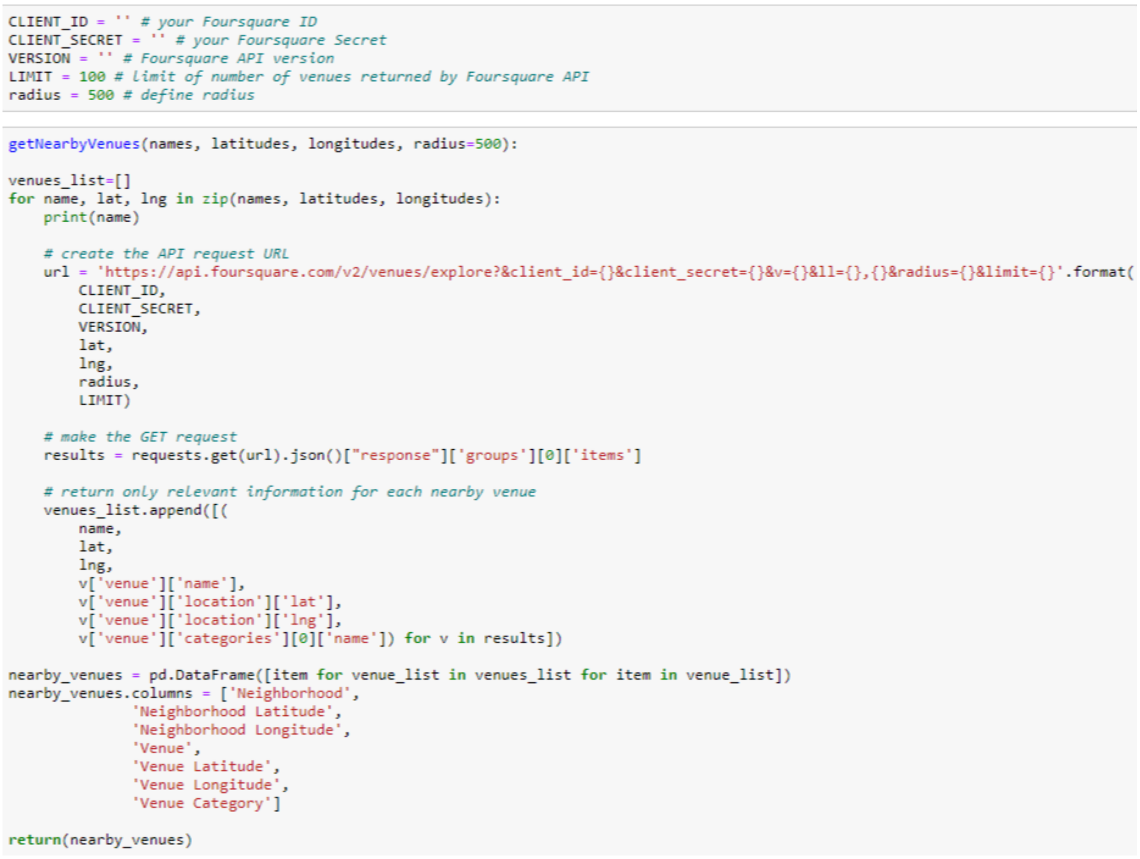
Figure 3. Code written in Python to obtain location by name.



	Neighborhood	Latitude	Longitude
0	Alna	59.932417	10.835276
1	Bjerke	59.941395	10.829208
2	Frogner	59.922224	10.706649
3	Gamle Oslo	59.899237	10.734767
4	Grorud	59.961424	10.880549
5	Grünerløkka	59.925471	10.777421
6	Nordre Aker	59.953638	10.756412
7	Nordstrand	54.487378	8.865286
8	Sagene	59.938273	10.765849
9	St. Hanshaugen	59.927950	10.738958
10	Stovner	59.962140	10.922823
11	Søndre Nordstrand	59.835944	10.798496
12	Ullern	59.925818	10.665132
13	Vestre Aker	59.958300	10.670319
14	Østensjø	59.887563	10.832748

Figure 4. Dataset with Neighborhood and exact location.

Now, we use Foursquare in order to look for the businesses in a radio of 500 meters.

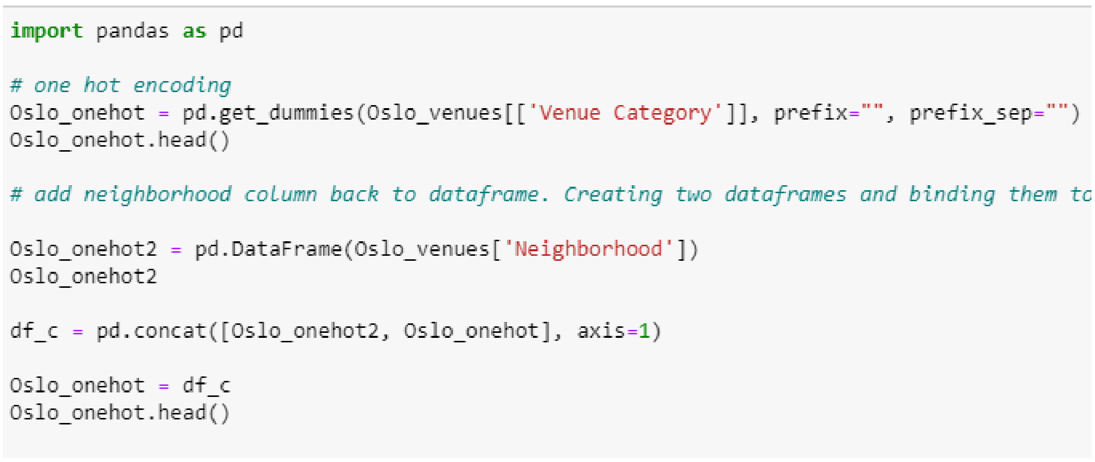


```
CLIENT_ID = '' # your Foursquare ID
CLIENT_SECRET = '' # your Foursquare Secret
VERSION = '' # Foursquare API version
LIMIT = 100 # Limit of number of venues returned by Foursquare API
radius = 500 # define radius
```

```
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)
```

Figure 5. Example of Foursquare request URL in Python.

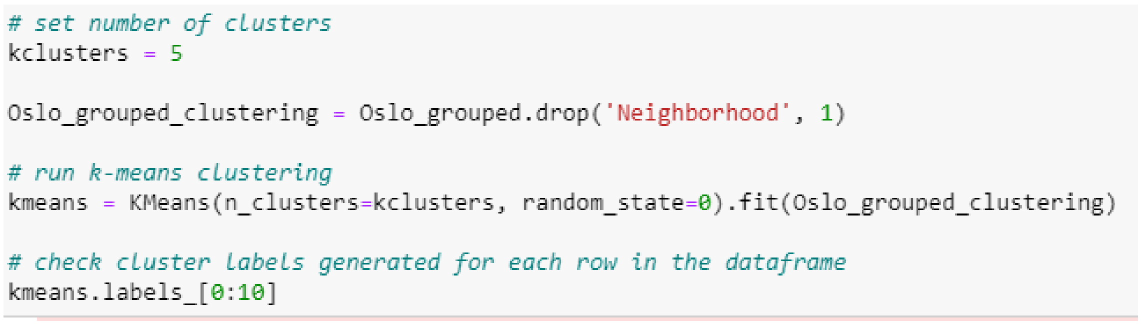
Then, we get the venues for all neighborhoods in our dataset and count the number of venues per neighbourhood. Now, we need to use the One Hot Encoding that is a common technique used to work with categorical features.



```
import pandas as pd

# one hot encoding
Oslo_onehot = pd.get_dummies(Oslo_venues[['Venue Category']], prefix="", prefix_sep="")
Oslo_onehot.head()

# add neighborhood column back to dataframe. Creating two dataframes and binding them to

Oslo_onehot2 = pd.DataFrame(Oslo_venues['Neighborhood'])
Oslo_onehot2

df_c = pd.concat([Oslo_onehot2, Oslo_onehot], axis=1)

Oslo_onehot = df_c
Oslo_onehot.head()
```

Figure 6. Obtaining category information.

We cluster the information using k-means algorithm, starting by 5 clusters.



```
# set number of clusters
kclusters = 5

Oslo_grouped_clustering = Oslo_grouped.drop('Neighborhood', 1)

# run k-means clustering
kmeans = KMeans(n_clusters=kclusters, random_state=0).fit(Oslo_grouped_clustering)

# check cluster labels generated for each row in the dataframe
kmeans.labels_[0:10]
```

Figure 7. Clustering analysis.

We use Folium in order to visualize the data.

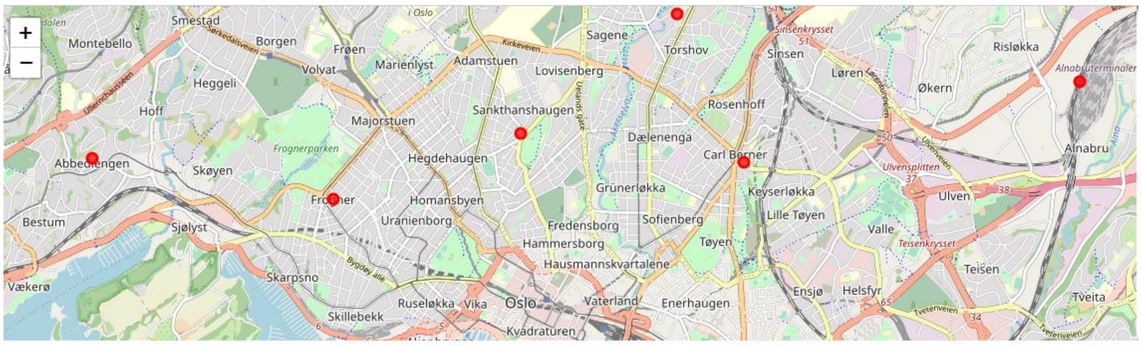


Figure 8. An example of the data obtained and visualized with Folium.

IV. RESULTS

The results show that the neighbourhoods around the center of Oslo have a higher commercial activity being a good choice in order to start a small business. The activity by itself will provide a a greater concentration of potential customers.