Capstone Project Report: The Battle of Neighborhoods

Leveraging data science to analyze the potential of starting business activities in the most promising neighborhoods of Munich, Germany

Christina Psylla, 09.08.21

Capstone Project Report: The Battle of Neighborhoods

Leveraging data science to analyze the potential of starting business activities in the most promising neighborhoods of Munich, Germany

Table of Contents

1. Introduction: Business Problem
2. Data
3. Methodology
4. Analysis
5. Results and Discussion
6. Conclusion

1. Introduction: Business Problem

In this report, we are trying to find a borough in Munich, where we could facilitate a new business, namely a greek restaurant. According to published data, Munich is host to the largest greek community in Germany. Therefore, we strongly believe that opening a greek restaurant in a suitable neighborhood in Munich will be a very profitable action, and it will comfort a lot of immigrants from Greece, who are homesick for greek traditional food. For this reason we will analyze and cluster Munich's neighborhoods in terms of venues# categories and price per square meter.

2. Data

In order to find the most promising borough for opening a greek restaurant in Munich we will utilize following data:

a. Information about the venues in all boroughs of Munich, which is gathered by web scraping 'https://www.muenchen.de/leben/service/postleitzahlen.html'. The Geocoder Python package (https://geocoder.readthedocs.io/index.html) will be used to provide us with the latitude and logitude coordinate for all neighborhoods in Munich. Longitude and latitude are then used with the Foursquare API to provide information about the nearby venues.

b. Average price per m² of the apartments in Munich: We will use web scraping to read and save the information from 'https://de.statista.com/statistik/daten/studie/260438/umfrage/mietpreise-in-muenchen-nach-bezirken/' into a dataframe. itude and longitude are used as input for FourSquare to source information about the boroughs.

3. Methodology

The steps will be as follow: Firstly, we will read the postal codes of each neighborhood into a dataframe and transform the structure of the dataframe, as several postal codes correspond to the same borough. Then we will utilize folium map and Munich's geographical coordinates in order to visualize the districts of Munich. Following, we analyze with Foursquare API the venues and venues' categories and we visualize their distribution on the map with color code, in order to gain insights into the lifestyle of each district and the popularity of different venues' categories such as modern and italian restaurants. We assume that a neighborhood, where mediterranean and modern restaurants are popular could be a target for our business. Nevertheless, we should also take into account the rent cost of each district for our decision, so that we start our business in a borough, where a greek restaurant would be popular, but also where the rent price is not too high. Finally, we perform a clustering and exploratory analysis to obtain information about the rent price per square meter and the number of venues.

4. Analysis

We start with importing and installing all necessary libraries for our analysis.

In [15]:
```
!pip install pandas
!pip install requests
!pip install folium
!conda install -c conda-forge folium=0.5.0 --yes
```

```
Requirement already satisfied: pandas in /opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages (1.3.1)
Requirement already satisfied: numpy>=1.17.3 in /opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages (from pandas) (1.20.3)
Requirement already satisfied: python-dateutil>=2.7.3 in /opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages (from pandas) (2.8.
2)
Requirement already satisfied: pytz>=2017.3 in /opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages (from pandas) (2021.1)
Requirement already satisfied: six>=1.5 in /opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages (from python-dateutil>=2.7.3->pand
as) (1.16.0)
Requirement already satisfied: requests in /opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages (2.26.0)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages (from requests) (3.1)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages (from requests) (1.2
6.6)
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages (from requests) (2021.5.
30)
Requirement already satisfied: charset-normalizer~=2.0.0 in /opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages (from requests)
(2.0.0)
Requirement already satisfied: folium in /opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages (0.12.1)
Requirement already satisfied: requests in /opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages (from folium) (2.26.0)
Requirement already satisfied: jinja2>=2.9 in /opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages (from folium) (3.0.1)
Requirement already satisfied: numpy in /opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages (from folium) (1.20.3)
Requirement already satisfied: branca>=0.3.0 in /opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages (from folium) (0.4.2)
Requirement already satisfied: MarkupSafe>=2.0 in /opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages (from jinja2>=2.9->folium)
(2.0.1)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages (from requests->foliu
m) (1.26.6)
Requirement already satisfied: charset-normalizer~=2.0.0 in /opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages (from requests->f
olium) (2.0.0)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages (from requests->folium) (3.1)
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages (from requests->folium)
(2021.5.30)
Collecting package metadata (current_repodata.json): done
Solving environment: done
```

In [16]:
```
import numpy as np # library to handle data in a vectorized manner

import pandas as pd # library for data analsysis
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)

import json # library to handle JSON files

#!conda install -c conda-forge geopy --yes # uncomment this line if you haven't completed the Foursquare API lab
from geopy.geocoders import Nominatim # convert an address into latitude 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.cm as cm
import matplotlib.colors as colors

# import k-means from clustering stage
from sklearn.cluster import KMeans

#!conda install -c conda-forge folium=0.5.0 --yes # uncomment this line if you haven't completed the Foursquare API lab
import folium # map rendering library

print('Libraries imported.')
```

```
Libraries imported.
```

Fig. 1 : Import of necessary libraries

We will read each postal code in a dataframe using following website: 'https://www.muenchen.de/int/en/living/postal-codes.html'

In [17]:
```
url = 'https://www.muenchen.de/int/en/living/postal-codes.html'
munich_data_list = pd.read_html(url)
munich_data = munich_data_list[0]
munich_data
```

Out[17]:

	District	Postal Code
0	Allach-Untermenzing	80995, 80997, 80999, 81247, 81249
1	Altstadt-Lehel	80331, 80333, 80335, 80336, 80469, 80538, 80539
2	Au-Haidhausen	81541, 81543, 81667, 81669, 81671, 81675, 81677
3	Aubing-Lochhausen-Langwied	81243, 81245, 81249
4	Berg am Laim	81671, 81673, 81735, 81825
5	Bogenhausen	81675, 81677, 81679, 81925, 81927, 81929
6	Feldmoching-Hasenbergl	80933, 80935, 80995
7	Hadern	80689, 81375, 81377
8	Laim	80686, 80687, 80689
9	Ludwigsvorstadt-Isarvorstadt	80335, 80336, 80337, 80469
10	Maxvorstadt	80333, 80335, 80539, 80636, 80797, 80798, 8079...
11	Milbertshofen-Am Hart	80807, 80809, 80937, 80939
12	Moosach	80637, 80638, 80992, 80993, 80997
13	Neuhausen-Nymphenburg	80634, 80636, 80637, 80638, 80639
14	Obergiesing	81539, 81541, 81547, 81549
15	Pasing-Obermenzing	80687, 80689, 81241, 81243, 81245, 81247
16	Ramersdorf-Perlach	81539, 81549, 81669, 81671, 81735, 81737, 81739
17	Schwabing-Freimann	80538, 80801, 80802, 80803, 80804, 80805, 8080...
18	Schwabing-West	80796, 80797, 80798, 80799, 80801, 80803, 8080...
19	Schwanthalerhöhe	80335, 80339
20	Sendling	80336, 80337, 80469, 81369, 81371, 81373, 81379
21	Sendling-Westpark	80686, 81369, 81373, 81377, 81379
22	Thalkirchen-Obersendling-Fürstenried-Forstenri...	81379, 81475, 81476, 81477, 81479
23	Trudering-Riem	81735, 81825, 81827, 81829
24	Untergiesing-Harlaching	81543, 81545, 81547

Fig. 2 : Dataframe containing the postal codes of Munich

We can see that several districts are assigned to more than one postal codes, therefore we split the above dataframe.

In [18]:
```
munich_data_cleaned = pd.DataFrame(columns=['District', 'Postal Code'])
munich_data_cleaned.head()
```

Out[18]:

	District	Postal Code

In [19]:
```
items = []
for idx, codes in enumerate(munich_data['Postal Code']):
    code_list = codes.split(',')
    district = munich_data['District'][idx]
    for element in code_list:
        element = element.replace(' ', '')
        items.append({'District': district, 'Postal Code': element})
```

In [20]:
```
munich_data_cleaned = munich_data_cleaned.append(items)
munich_data_cleaned.head()
```

Out[20]:

	District	Postal Code
0	Allach-Untermenzing	80995
1	Allach-Untermenzing	80997
2	Allach-Untermenzing	80999
3	Allach-Untermenzing	81247
4	Allach-Untermenzing	81249

In [21]:
```
#Clean data
muc_data_cleaned = pd.DataFrame(columns=['District', 'Postal Code'])
muc_data_cleaned.head()
```

Out[21]:

	District	Postal Code

Fig. 3 : Splitting the dataframe

We use our credentials for Foursquare API in order to fetch latitude and longitude data for each postal code and we create a new dataframe, which contains the geographical coordinates of each district.

In [23]:
```
# create new dataframe containing latitude and longitude values
munich_data_ll = pd.DataFrame(columns=['District', 'Postal Code', 'Latitude', 'Longitude'])

# Loop
items = []
for idx, district in enumerate(munich_data_cleaned['District']):
    code = munich_data_cleaned['Postal Code'][idx]
    address = district + ', ' + code # to get format of address

    geolocator = Nominatim(user_agent="ny_explorer")
    location = geolocator.geocode(address)
    latitude = location.latitude
    longitude = location.longitude
    items.append({'District': district,
                  'Postal Code': code,
                  'Latitude': latitude,
                  'Longitude': longitude})
```

In [24]:
```
munich_data_ll = munich_data_ll.append(items)
munich_data_ll.head()
```

Out[24]:

	District	Postal Code	Latitude	Longitude
0	Allach-Untermenzing	80995	48.195157	11.462973
1	Allach-Untermenzing	80997	48.195157	11.462973
2	Allach-Untermenzing	80999	48.195157	11.462973
3	Allach-Untermenzing	81247	48.195157	11.462973
4	Allach-Untermenzing	81249	48.195157	11.462973

Fig. 4 : Dataframe containing geographical coordinates

We can now create Munich's map utilizing the folium library.

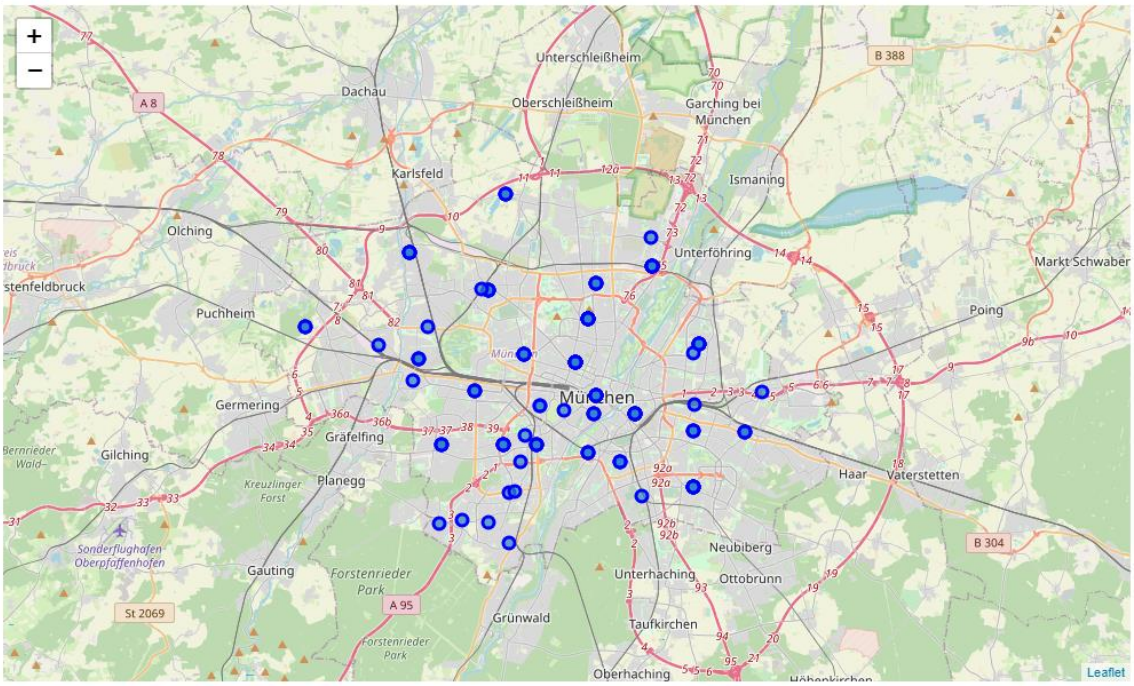

Fig. 5 : Map of Munich using Folium library

We explore the nearby in each borough venues with the Foursquare API and we create the Munich venues dataframe.

```
In [30]: munich_venues.groupby('District').count()
```

Out[30]:

	Latitude	Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
District						
Allach-Untermenzing	45	45	45	45	45	45
Altstadt-Lehel	210	210	210	210	210	210
Au-Haidhausen	210	210	210	210	210	210
Berg am Laim	29	29	29	29	29	29
Bogenhausen	66	66	66	66	66	66
Feldmoching-Hasenbergl	3	3	3	3	3	3
Hadern	30	30	30	30	30	30
Laim	66	66	66	66	66	66
Ludwigsvorstadt-Isarvorstadt	120	120	120	120	120	120
Maxvorstadt	270	270	270	270	270	270
Milbertshofen-Am Hart	76	76	76	76	76	76
Moosach	122	122	122	122	122	122
Neuhausen-Nymphenburg	150	150	150	150	150	150
Obergiesing	52	52	52	52	52	52
Pasing-Obermenzing	57	57	57	57	57	57
Ramersdorf-Perlach	77	77	77	77	77	77
Schwabing-Freimann	94	94	94	94	94	94
Schwabing-West	176	176	176	176	176	176
Schwanthalerhöhe	60	60	60	60	60	60
Sendling	157	157	157	157	157	157
Sendling-Westpark	45	45	45	45	45	45
Thalkirchen-Obersendling-Fürstenried-Forstenried-Solln	61	61	61	61	61	61
Trudering-Riem	40	40	40	40	40	40
Untergiesing-Harlaching	90	90	90	90	90	90

Fig. 6 : Munich venues dataframe

We are specifically interested in greek restaurants and taverns, therefore we search for these categories across Munich's districts. We create a Munich map with markers representing greek restaurants and tavernas and we visualize it:

In [64]:
```
#venues of interest
muc_venueint = munich_venues[(munich_venues['Venue Category'].str.contains('Greek')==True)  | (munich_venues['Venue Category'].str.contai
ns('Taverna')==True)]

#all other venues
muc_venueother =  munich_venues[(munich_venues['Venue Category'].str.contains('Greek')==False)  | (munich_venues['Venue Category'].str.co
ntains('Taverna')==False)]
```

In [58]:
```
#Dataframes with specific favourite venues
muc_venuegreek = munich_venues[munich_venues['Venue Category'].str.match('Greek')]
muc_venuetaverna = munich_venues[munich_venues['Venue Category'].str.match('Taverna')]
```

In [82]:
```
muc_venuetaverna.head(10)
```

Out[82]:

	District	Latitude	Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
1473	Pasing-Obermenzing	48.143502	11.465118	Taverna Kypros	48.145074	11.459974	Taverna
1493	Pasing-Obermenzing	48.165049	11.474079	Taverna Naxos	48.164144	11.479353	Taverna
1682	Schwabing-West	48.168271	11.569873	Georgios	48.165728	11.564714	Taverna
1704	Schwabing-West	48.168271	11.569873	Georgios	48.165728	11.564714	Taverna
1726	Schwabing-West	48.168271	11.569873	Georgios	48.165728	11.564714	Taverna
1748	Schwabing-West	48.168271	11.569873	Georgios	48.165728	11.564714	Taverna
1770	Schwabing-West	48.168271	11.569873	Georgios	48.165728	11.564714	Taverna
1792	Schwabing-West	48.168271	11.569873	Georgios	48.165728	11.564714	Taverna
1814	Schwabing-West	48.168271	11.569873	Georgios	48.165728	11.564714	Taverna
1836	Schwabing-West	48.168271	11.569873	Georgios	48.165728	11.564714	Taverna

In [81]:
```
muc_venuegreek.head(10)
```

Out[81]:

	District	Latitude	Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
496	Bogenhausen	48.158487	11.636682	Pyrsos	48.154944	11.637708	Greek Restaurant
508	Bogenhausen	48.158487	11.636682	Pyrsos	48.154944	11.637708	Greek Restaurant
520	Bogenhausen	48.158487	11.636682	Pyrsos	48.154944	11.637708	Greek Restaurant
532	Bogenhausen	48.158487	11.636682	Pyrsos	48.154944	11.637708	Greek Restaurant
544	Bogenhausen	48.154782	11.633484	Pyrsos	48.154944	11.637708	Greek Restaurant
550	Bogenhausen	48.158487	11.636682	Pyrsos	48.154944	11.637708	Greek Restaurant
560	Feldmoching-Hasenbergl	48.218462	11.520409	Seehaus Feldmoching	48.216965	11.517452	Greek Restaurant
561	Feldmoching-Hasenbergl	48.218462	11.520409	Seehaus Feldmoching	48.216965	11.517452	Greek Restaurant
562	Feldmoching-Hasenbergl	48.218462	11.520409	Seehaus Feldmoching	48.216965	11.517452	Greek Restaurant
593	Laim	48.139551	11.502166	Potlatsch	48.139821	11.500279	Greek Restaurant

Fig. 7 : Venues of interest

Out[21]:

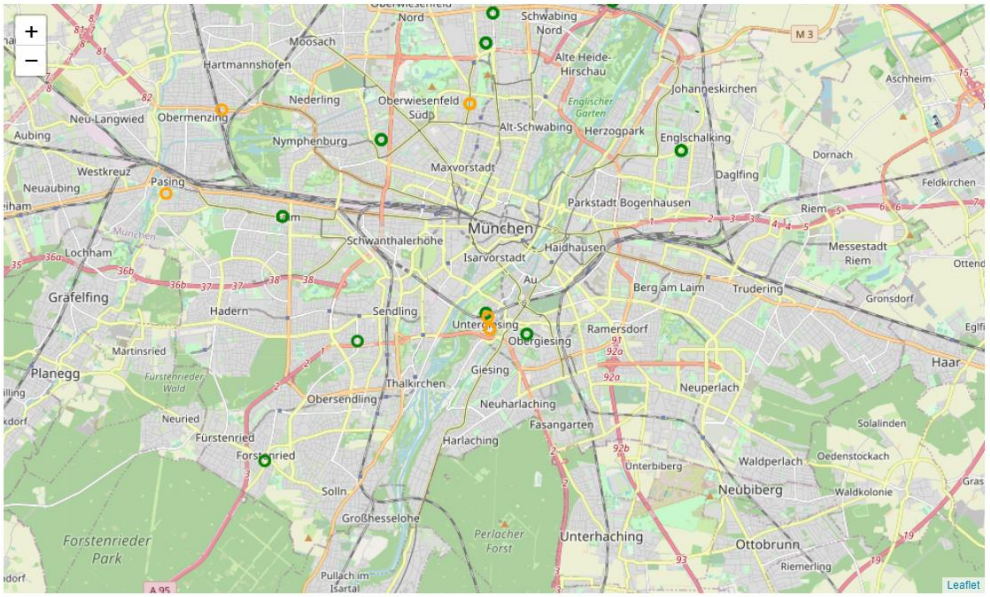

Fig. 8 : Visualization of the venues of interest

We are nevertheless interested in other venue categories too, so we check the mean of the frequency of occurrence of each venue category per district and we print the first 5 most common venues. We are aiming for obtaining information related to similarities between districts, so we split Munich's districts into 5 clusters.

We create a dataframe which includes the clusters and the most common venues and then we visualize the clusters with folium.

In [84]:
```
munich_grouped = munich_onehot.groupby('District').mean().reset_index()
munich_grouped.head(10)
```

Out[84]:

	District	ATM	Afghan Restaurant	American Restaurant	Arcade	Art Museum	Asian Restaurant	Athletics & Sports	Austrian Restaurant	Auto Dealership	Automotive Shop	BBQ Joint	Bagel Shop	Bakery	Bank
0	Allach-Untermenzing	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.200000	0.0
1	Altstadt-Lehel	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.142857	0.0	0.0	0.142857	0.0
2	Au-Haidhausen	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.142857	0.0	0.0	0.142857	0.0
3	Aubing-Lochhausen-Langwied	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.000000	0.0
4	Berg am Laim	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.250000	0.0	0.0	0.000000	0.0
5	Bogenhausen	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.166667	0.0
6	Feldmoching-Hasenbergl	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.333333	0.0	0.0	0.000000	0.0
7	Hadern	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.333333	0.0
8	Laim	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.000000	0.0
9	Ludwigsvorstadt-Isarvorstadt	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.250000	0.0	0.0	0.000000	0.0

Out[35]:

	District	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	8th Most Common Venue	9th Most Common Venue	10th Most Common Venue
0	Allach-Untermenzing	Sporting Goods Shop	Drugstore	Supermarket	Italian Restaurant	Bakery	Market	Men's Store	Metro Station	Miscellaneous Shop	Mobile Phon Shop
1	Altstadt-Lehel	Drugstore	Playground	Supermarket	Automotive Shop	Sporting Goods Shop	Bakery	ATM	Motel	Men's Store	Metro Statior
2	Au-Haidhausen	Supermarket	Bakery	Playground	Italian Restaurant	Automotive Shop	Drugstore	ATM	Miscellaneous Shop	Mobile Phone Shop	Modern European Restaurant
3	Aubing-Lochhausen-Langwied	Drugstore	Italian Restaurant	Sporting Goods Shop	ATM	Motel	Men's Store	Metro Station	Miscellaneous Shop	Mobile Phone Shop	Modern European Restaurant
4	Berg am Laim	Supermarket	Drugstore	Automotive Shop	ATM	Motel	Men's Store	Metro Station	Miscellaneous Shop	Mobile Phone Shop	Modern European Restaurant
5	Bogenhausen	Sporting Goods Shop	Bakery	Playground	Supermarket	Italian Restaurant	Drugstore	Men's Store	Metro Station	Miscellaneous Shop	Mobile Phon Shop
6	Feldmoching-Hasenbergl	Supermarket	Drugstore	Automotive Shop	ATM	Motel	Men's Store	Metro Station	Miscellaneous Shop	Mobile Phone Shop	Modern European Restaurant
7	Hadern	Playground	Drugstore	Bakery	ATM	Movie Theater	Metro Station	Miscellaneous Shop	Mobile Phone Shop	Modern European Restaurant	Motel
8	Laim	Sporting Goods Shop	Supermarket	Italian Restaurant	Modern European Restaurant	Manti Place	Market	Men's Store	Metro Station	Miscellaneous Shop	Mobile Phon Shop
9	Ludwigsvorstadt-Isarvorstadt	Playground	Supermarket	Automotive Shop	Drugstore	ATM	Motel	Men's Store	Metro Station	Miscellaneous Shop	Mobile Phon Shop
10	Maxvorstadt	Plaza	Burrito Place	Fountain	Church	Department Store	Men's Store	Gourmet Shop	German Restaurant	ATM	Miscellaneou Shop

Fig. 9 : Munich most common venues per district dataframe

In [36]:
```
#Cluster neighborhoods

num_clusters = 5

X = munich_grouped.drop('District', 1)

kmeans = KMeans(n_clusters=num_clusters, random_state=0).fit(X)
```

```
/opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages/ipykernel/__main__.py:5: FutureWarning: In a future version of pandas all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only
```

In [37]:
```
#Add clustering labels
district_venues_sorted.insert(0, 'Cluster Labels', kmeans.labels_)

munich_merged = munich_data_ll

# merge labels and data about venues to district data and latitude plus longitude data to have all in one dataframe
munich_merged = munich_merged.join(district_venues_sorted.set_index('District'), on='District')

munich_merged.head()
```

Out[37]:

	District	Postal Code	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 Most Common Venue	8th Most Common Venue	9th Most Common Venue
0	Allach-Untermenzing	80995	48.195157	11.462973	4	Sporting Goods Shop	Drugstore	Supermarket	Italian Restaurant	Bakery	Market	Men's Store	Metro Station	Miscellaneous Shop
1	Allach-Untermenzing	80997	48.195157	11.462973	4	Sporting Goods Shop	Drugstore	Supermarket	Italian Restaurant	Bakery	Market	Men's Store	Metro Station	Miscellaneous Shop
2	Allach-Untermenzing	80999	48.195157	11.462973	4	Sporting Goods Shop	Drugstore	Supermarket	Italian Restaurant	Bakery	Market	Men's Store	Metro Station	Miscellaneous Shop
3	Allach-Untermenzing	81247	48.195157	11.462973	4	Sporting Goods Shop	Drugstore	Supermarket	Italian Restaurant	Bakery	Market	Men's Store	Metro Station	Miscellaneous Shop
4	Allach-Untermenzing	81249	48.195157	11.462973	4	Sporting Goods Shop	Drugstore	Supermarket	Italian Restaurant	Bakery	Market	Men's Store	Metro Station	Miscellaneous Shop

Fig.10 : Clustering neighborhoods in Munich

Out[31]:

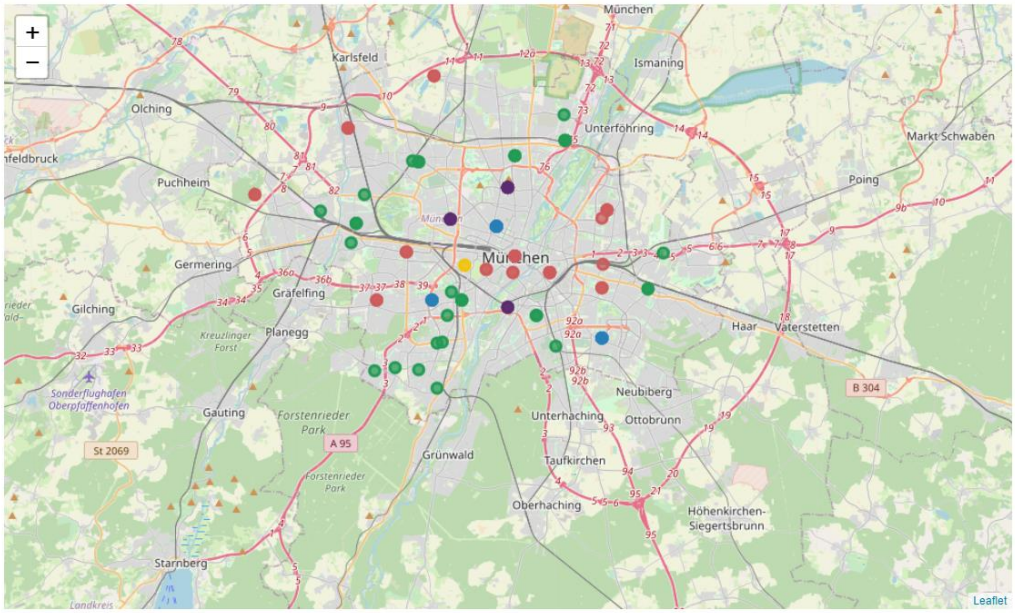

Fig.11 : Visualization of Munich's clusters

Now we can examine the different clusters:

```
In [39]: # first: examine the green cluster (number zero)
         cluster0 = munich_merged.loc[munich_merged['Cluster Labels'] == 0, munich_merged.columns[[1] + list(range(5, munich_merged.shape[1]))]]
         cluster0['1st Most Common Venue'].value_counts()

Out[39]: Bookstore      9
         Coffee Shop    9
         Irish Pub      8
         Name: 1st Most Common Venue, dtype: int64

In [40]: # next: examine the red cluster (number one)
         cluster1 = munich_merged.loc[munich_merged['Cluster Labels'] == 1, munich_merged.columns[[1] + list(range(5, munich_merged.shape[1]))]]
         cluster1['1st Most Common Venue'].value_counts()

Out[40]: Supermarket            14
         Drugstore               7
         Playground              7
         Sporting Goods Shop     6
         Name: 1st Most Common Venue, dtype: int64

In [41]: # next: examine the blue cluster (number two)
         cluster2 = munich_merged.loc[munich_merged['Cluster Labels'] == 2, munich_merged.columns[[1] + list(range(5, munich_merged.shape[1]))]]
         cluster2['1st Most Common Venue'].value_counts()

Out[41]: Boutique    12
         Creperie     8
         Name: 1st Most Common Venue, dtype: int64

In [42]: # examine the purple cluster (number three)
         cluster3 = munich_merged.loc[munich_merged['Cluster Labels'] == 3, munich_merged.columns[[1] + list(range(5, munich_merged.shape[1]))]]
         cluster3['1st Most Common Venue'].value_counts()

Out[42]: Plaza                9
         Gourmet Shop          7
         Fountain              7
         Sporting Goods Shop   6
         Men's Store           5
         Name: 1st Most Common Venue, dtype: int64

In [43]: # examine the yellow cluster (number four)
         cluster4 = munich_merged.loc[munich_merged['Cluster Labels'] == 4, munich_merged.columns[[1] + list(range(5, munich_merged.shape[1]))]]
         cluster4['1st Most Common Venue'].value_counts()

Out[43]: Sporting Goods Shop    8
         Drugstore              3
         Hotel                  2
         Name: 1st Most Common Venue, dtype: int64
```

Fig.11 : Obtaining information about different clusters in Munich

We can see that there are neighborhoods in Munich, where mediterranean restaurants are very popular, so we will focus on such districts.

Nevertheless, Munich is a really expensive city in terms of rent prices, so below we will explore rent prices per district by utilizing the website: 'https://www.tz.de/leben/wohnen/uebersicht-muenchner-mieten-preise-nach-postleitzahlen-tz-6133643.html'

```
In [45]: #Get rent price per square meter
df_mucPrice.rename(columns={'PLZ':'PostalCode', 'Miete':'PricePerm2'}, inplace = True)
df_mucPrice["PricePerm2"] = df_mucPrice["PricePerm2"] /100
df_mucPrice.head(20)
```

Out[45]:

	PostalCode	PricePerm2	Trend	Kaufpreis	Trend.1
1	80995	14.10	1,1%	5000	6,8%
2	80997	13.25	-1,9%	5430	12,4%
3	80999	13.05	5,2%	5880	11,4%
4	81247	14.55	2,5%	6520	1,4%
5	81249	13.25	5,6%	5100	3,4%
7	80331	22.30	3,5%	k.A.	k.A.
8	80333	19.10	1,9%	9120	20,8%
9	80335	19.55	2,9%	8690	5,8%
10	80336	18.15	0,0%	8960	9,0%
11	80469	2.06	4,8%	8370	0,5%
12	80538	20.25	4,1%	10 900	23,4%
13	80539	21.45	-17,3%	k.A.	k.A.
15	81541	17.95	-0,3%	6030	4,7%
16	81543	15.55	-2,5%	5700	14%
17	81667	19.30	0,5%	7640	3,2%
18	81669	17.50	2,9%	5900	7,1%
19	81671	14.50	7,4%	7240	9,2%
20	81675	17.95	-3,0%	8880	22,0%
21	81677	19.45	-2,0%	5780	3,8%
23	81243	13.95	6,9%	5220	13,0%

Fig.12: Price per square meter for each Munich district

5. Results & Discussion

The above analysis demonstrates, that the most suitable districts in Munich for opening a greek restaurant are located in cluster 3, because the frequency of occurence of gourmet shops is high:

Plaza 9 Gourmet Shop 7 Fountain 7 Sporting Goods Shop 6 Men's Store 5

Such districts are Neu Langwied, Sendling, Obersendling, Pasing, Obermenzing, Perlach, Maxvorstadt and Neuperlach.

Our findings from the rent price exploratory analysis suggest, that the top 5 neighborhoods with the lowest rent price per m2 in the purple cluster are:

7. Neu Langwied
8. Allach
9. Aubing
10. Pasing
11. Moosach

6. Conclusion

Based on our analysis and results, we choose Pasing-Obermenzing as the district that will host our new greek restaurant because there already exist 3 greek tavernas and 0 greek restaurants.

That means that a lot of Greek people live in this neighborhood and since there is no greek restaurant operating there we will have the monopoly of the business.

In terms of price Pasing is not as cheap as Neu Langwied, but also not as pricey as Moosach or Maxvorstadt, therefore it is the most suitable district for our new operations.