Coursera
IBM Data Science Specialization

Final Capstone Project Report

OR

Berlin or Munich? - Clustering the Neighborhoods of Berlin and Munich

Submitted by	Rekha Remadevi
Submission date	09.03.2021
Course of Study	IBM Data Science Professional Certificate.

1 INTRODUCTION

It is quite difficult to find two more dramatically contrasting cities within the same country than Berlin and Munich.

Munich and Berlin are two of the best cities in Germany, and both have a lot to offer. Berlin is not only Germany's capital and largest city; it is also the cultural hub of the nation. One of the most fascinating cities in Europe, Berlin is vibrant and edgy and is Germany's centre for fashion, art and culture. Berlin's nightlife is famously impressive, known as the techno capital of the world. Berlin is a city of culinary delight offering a wide variety of food ranging from traditional German food to American burgers, at an affordable price.

Munich is the wealthy capital of Bavaria and the gateway to the Alps. It is said to be one of the most beautiful and charming cities in all of Germany and is filled with museums and beautiful architecture. It is most famous for being the centre of Oktoberfest festivities, which attracts over 6 million visitors every year.

2 BUSINESS PROBLEM

A person wants to relocate to Germany, and he is considering moving and buying an apartment within the neighborhoods of Berlin or Munich. In order to make a decision, he aims to obtain some information about the neighborhoods/ districts in Berlin and Munich. This information also helps tourists to choose their destinations depending on the experiences that the neighborhoods have to offer and what they would like to experience. This study will help stakeholders make informed decisions and address any concerns they have including the different kinds of cuisines, provision stores and what the city has to offer.

3 DATA DESCRIPTION

In terms of data, it is of paramount importance to obtain geographical location data for both Berlin and Munich. Postal codes in each city serve as a starting point. Using Postal codes, one can find out the neighborhoods, boroughs, venues and their most popular venue categories.

3.1 Berlin

To derive our solution, We scrape our data from http://www.places-in-germany.com/14356-places-within-a-radius-of-15km-around-berlin.html

This www.Places-in-Germany.com page has information about some states in Germany

1. Borough: Name of Neighbourhood
2. Zip code: Postal codes for Belin

The data will be scraped with a tool called *Beautiful Soup* and directly transformed to *Pandas* and *Data Frame*. It needs some cleaning while gathering only the boroughs of the city.

3.2 Munich

The postal code and district names of all districts in Munich are required to solve the task. The data published at https://www.muenchen.de/int/en/living/postal-codes.html is used in order to fetch the necessary data. The data is fetched by using the *pandas library* and the built-in *pd. read HTML ()* function. This function scrapes the data available on the website and stores all tables in data frames.

1. District: Districts name
2. Postal Code: Postal code for Districts in Munich

3.3 Geodata

The *python geopy* library is used for getting the latitude and longitude values. This library only requires the name of a neighborhood and accepts also a postal code and returns the latitude and longitude values for the given address.

3.4 Venue data

As a next step, the available top 100 venues shall be fetched for each postal code. For this problem, we will get the services of *Foursquare API* to explore the data of two cities, in terms of their neighborhoods. The data also include the information about the places around each neighborhood like restaurants, hotels, coffee shops, parks, theaters, art galleries, museums and many more. We selected one Borough from each city to analyze their neighborhoods.

We will use machine learning technique, *Clustering* to segment the neighborhoods with similar objects on the basis of each neighborhood data. These objects will be given priority on the basis of foot traffic (activity) in their respective neighborhoods. This will help to locate the tourist's areas and hubs, and then we can judge the similarity or dissimilarity between two cities on that basis.

4 METHODOLOGY

By using *Python libraries*, we will be creating our model. The required packages are the following:

```
import pandas as pd
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
import numpy as np
import json
from geopy.geocoders import Nominatim
from pandas.io.json import json_normalize
from bs4 import BeautifulSoup
import requests
import geopy as geo
import geopandas as gpd
import folium
from sklearn.cluster import KMeans

import matplotlib.cm as cm
import matplotlib.colors as colors
```

4.1 Data Collection

To collect data for Berlin, We scrape our data from http://www.places-in-germany.com/14356-places-within-a-radius-of-15km-around-berlin.html using the following code:

```
url='http://www.places-in-germany.com/14356-places-within-a-radius-of-15km-around-berlin.html'
req=requests.get(url)
soup=BeautifulSoup(req.text,"html.parser")
table = soup.find_all('table')
df=pd.read_html(str(table), header=0)[0]
```

The data look like this

	Distance	Route	Postal code / Place	Population
0	1.2 km (0.8 miles)	NaN	10115 Mitte	79582
1	2.1 km (1.3 miles)	NaN	10119 Prenzlauer Berg	140881
2	2.8 km (1.7 miles)	NaN	10115 Mitte	333534
3	2.9 km (1.8 miles)	NaN	13347 Gesundbrunnen	82110
4	3.3 km (2.0 miles)	NaN	10243 Friedrichshain-Kreuzberg	269398
...	...	...	...	...
93	14.6 km (9.1 miles)	NaN	12459 Köpenick	59201
94	14.8 km (9.2 miles)	NaN	12524 Altglienicke	26101
95	14.9 km (9.3 miles)	NaN	16341 Schwanebeck bei Bernau bei Berlin	-
96	15.0 km (9.3 miles)	NaN	12305 Lichtenrade	49451
97	15.0 km (9.3 miles)	NaN	12157 Steglitz-Zehlendorf	293955

Figure 4-1 – Overview of Postal codes in the neighborhoods of Berlin.

By using the page https://www.muenchen.de/int/en/living/postal-codes.html, Munich data has been collected. The data is fetched by using the *pandas library* and the built-in *pd. read HTML ()* function. This function scrapes the data available on the website and stores all tables in data frames. The following codes has been used for this.

```
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
```

The extracted data look like this

	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

Figure 4-2 – Overview of Postal codes and Districts names in Munich.

4.2 Data preprocessing

We split all the places according to boroughs for Berlin

```
df.rename({'Postal code / Place':'Borough'},axis=1, inplace=True)
```

```
berlin=df.Borough.str.split(expand=True) # Spliting Zip-Code and Place name
```

```
# concatenating Borough again
borough=[]
for name, values in berlin.iterrows():
    #print(name, values[0], values[1],values[2])
    if values[2] is None:
        borough.append(values[1])
    else:
        borough.append(values[1] + ' ' + values[2])
```

```
berlin['Borough']=borough
```

```
berlin.rename({0:'Zipcode'}, axis=1, inplace=True)
```

We split all the places according to their postal codes for Munich

```
address = 'Munich'

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

```
The geograpical coordinate of Munich are 48.1371079, 11.5753822.
```

4.3 Feature Selection

For both the datasets, only the borough, neighbourhood, postal codes and geolocations (latitude and longitude) are needed.

```
#Keeping only coloumn with Zip Code and Borough
berlin=berlin[['Zipcode', 'Borough']]
berlin.shape
```

```
(98, 2)
```

```
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})
```

4.4 Visualizing the Neighborhoods of Berlin and Munich

Using the *Folium package*, it is possible to visualize the maps of Berlin and Munich with the neighbourhoods.

Neighbourhood Map of Berlin:

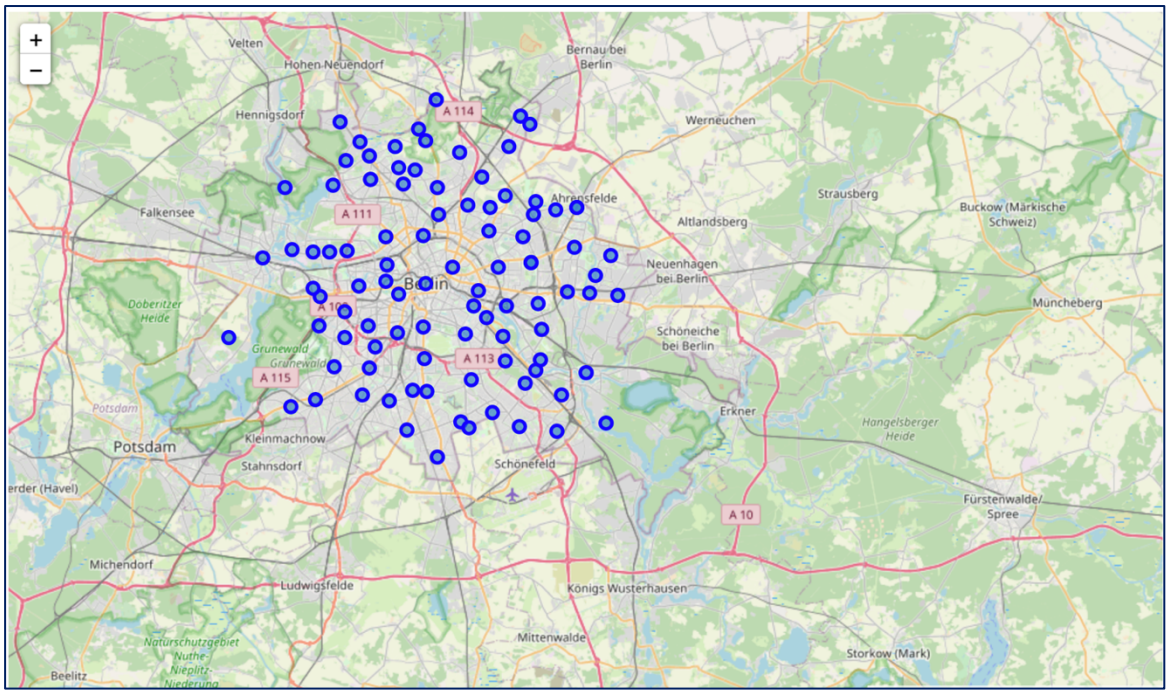

Figure 4-3 - Overview Berlin Boroughs

Neighbourhood Map of Munich:

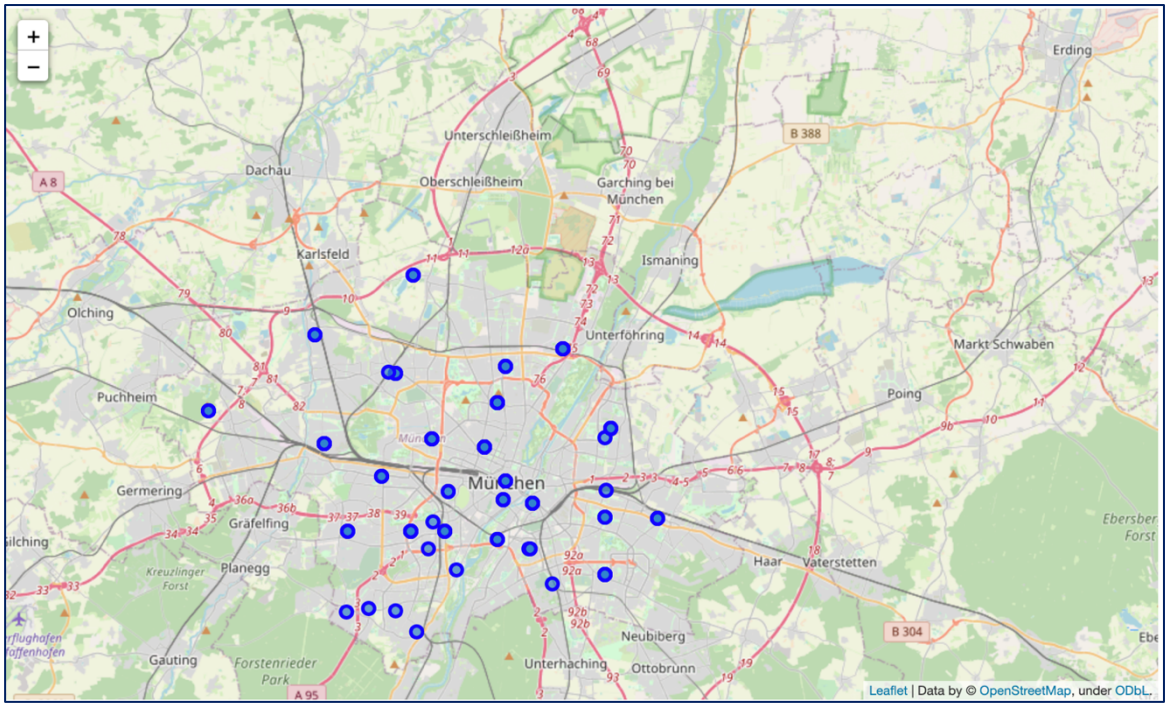

Figure 4-4 - Figure 4: District wise map of Munich

Visualization of the neighbourhoods is available now and with the help of *Foursquare API,* it is easy to collect information pertaining to each neighbourhood including that of the name of the neighbourhood, geo-coordinates, venue and venue categories within a 500m radius of each neighborhood

```
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={}&limi
            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)
```

Resulting data look like this

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Mitte	52.517012	13.388822	Dussmann das KulturKaufhaus	52.518312	13.388708	Bookstore
1	Mitte	52.517012	13.388822	Dussmann English Bookshop	52.518223	13.389239	Bookstore
2	Mitte	52.517012	13.388822	Cookies Cream	52.516569	13.388008	Vegetarian / Vegan Restaurant
3	Mitte	52.517012	13.388822	Freundschaft	52.518294	13.390344	Wine Bar
4	Mitte	52.517012	13.388822	Komische Oper	52.515968	13.386701	Opera House

Figure 4-5 - Overview of Latitude and Longitude in Berlin

4.5 One Hot Coding

One Hot Encoding is a process by which categorical variables are converted into a form that could be provided to ML algorithms to do a better job in prediction. Since we are trying to find out what are the different kinds of venue categories present in each neighbourhood and then calculate the top 10 common venues to base our similarity on, we use the One Hot Encoding to work with our categorical datatype of the venue categories.

```
berlin_venue_cat = pd.get_dummies(venues_berlin[['Venue Category']], prefix="", prefix_sep="")
berlin_venue_cat['Neighborhood'] = venues_berlin['Neighborhood']
# moving neighborhood column to the first column
fixed_columns = [berlin_venue_cat.columns[-1]] + list(berlin_venue_cat.columns[:-1])
berlin_venue_cat = berlin_venue_cat[fixed_columns]
#Grouping and Calculating to the mean
berlin_grouped = berlin_venue_cat.groupby('Neighborhood').mean().reset_index()
berlin_grouped.head()
```

	Neighborhood	ATM	Adult Boutique	African Restaurant	American Restaurant	Argentinian Restaurant	Art Gallery	Art Museum	Asian Restaurant	Austrian Restaurant	...	Volleyball Court	Warehouse Store	Waterfall	Wa
0	Adlershof	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0	
1	Ahrensfelde	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0	
2	Ahrensfelde bei	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0	
3	Alt-Hohenschönhausen	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.111111	0.0	...	0.0	0.0	0.0	
4	Alt-Treptow	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0	

Figure 4-6 - Overview of venues in Berlin.

4.6 Top Venues in the Neighbourhoods

In this step, we will rank and label top venue categories in the neighbourhoods of Berlin and Munich.

Let's make a function to get the topmost common venue categories

```
def return_most_common_venues(row, num_top_venues):
    row_categories = row.iloc[1:]
    row_categories_sorted = row_categories.sort_values(ascending=False)

    return row_categories_sorted.index.values[0:num_top_venues]
```

```
num_top_venues = 15
indicators = ['st', 'nd', 'rd']

# create columns according to number of top venues
columns = ['Neighborhood']
for ind in np.arange(num_top_venues):
    try:
        columns.append('{}{} Most Common Venue'.format(ind+1, indicators[ind]))
    except:
        columns.append('{}th Most Common Venue'.format(ind+1))

# create a new dataframe
neighborhoods_venues_sorted = pd.DataFrame(columns=columns)
neighborhoods_venues_sorted['Neighborhood'] = berlin_grouped['Neighborhood']

for ind in np.arange(berlin_grouped.shape[0]):
    neighborhoods_venues_sorted.iloc[ind, 1:] = return_most_common_venues(berlin_grouped.iloc[ind, :], num_top_venues)

neighborhoods_venues_sorted.head()
```

4.7 Model Building- K means

By using *KMeans Clustering* Machine learning algorithm to cluster similar neighbourhoods together, we will be going with the number of clusters as 5.

```
# Set number of clusters
k_num_clusters = 5

berlin_grouped_clustering = berlin_grouped.drop('Neighborhood', 1)

# run k-means clustering
kmeans = KMeans(n_clusters=k_num_clusters, random_state=0).fit(berlin_grouped_clustering)
kmeans
```

Join Berlin grouped with combined data on neighbourhood to add latitude & longitude for each neighbourhood to prepare it for plotting.

	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	8th Most Common Venue	9th Most Common Venue	10th Most Common Venue	11th Most Common Venue	12th Mc Comm Veni
0	Adlershof	Supermarket	Italian Restaurant	Trattoria/Osteria	Greek Restaurant	Pizza Place	Drugstore	Steakhouse	Furniture / Home Store	Gaming Cafe	Fried Chicken Joint	Fish Market	Fren Restaura
1	Ahrensfelde	Supermarket	Gas Station	Zoo Exhibit	Gay Bar	Garden Center	Garden	Gaming Cafe	Furniture / Home Store	Fried Chicken Joint	French Restaurant	Fountain	Fo Co
2	Ahrensfelde bei	Supermarket	Gas Station	Zoo Exhibit	Gay Bar	Garden Center	Garden	Gaming Cafe	Furniture / Home Store	Fried Chicken Joint	French Restaurant	Fountain	Fo Co
3	Alt-Hohenschönhausen	Tram Station	Supermarket	Indian Restaurant	Drugstore	Asian Restaurant	Coffee Shop	Greek Restaurant	Big Box Store	Fried Chicken Joint	Food Court	Fountain	Fren Restaura
4	Alt-Treptow	Bakery	Italian Restaurant	Nightclub	Mexican Restaurant	Tapas Restaurant	Big Box Store	Garden Center	Bus Stop	Snack Place	Café	Outdoor Sculpture	Seafo Restaura

Figure 4-7 - Overview of top 15 venues in Berlin

4.8 Visualizing the clustered Neighbourhoods

Data is processed, collected and compiled. Missing data were removed. Finally, the model is built.

Using *Folium package* clustered neighbourhoods map are built.

We drop all the *NaN values* to prevent data skew

```
Berlin_merged_nonan =Berlin_merged.dropna(subset=['Cluster Labels'])
```

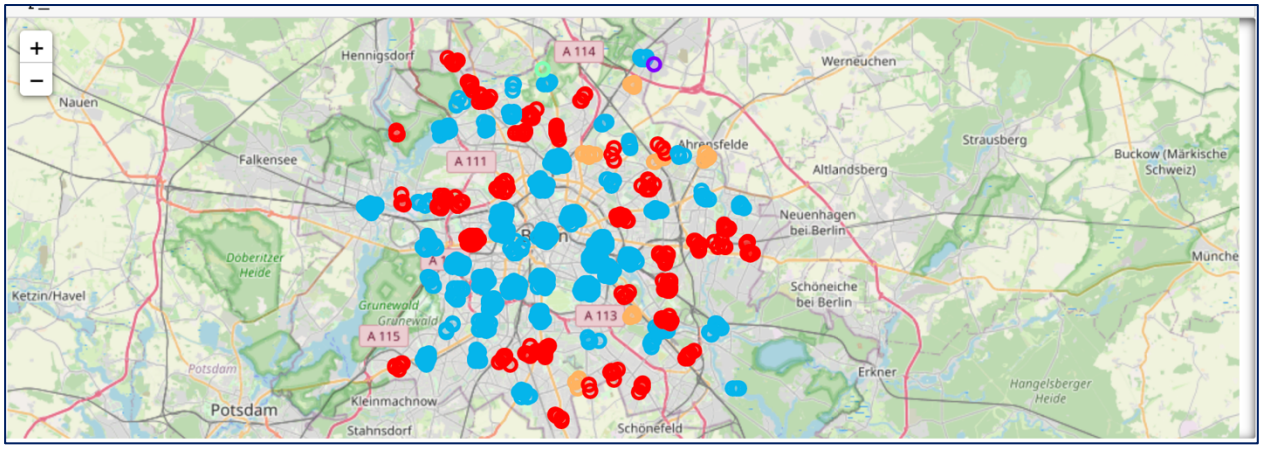

Figure 4-8 - Map of clustered neighbourhoods of Berlin

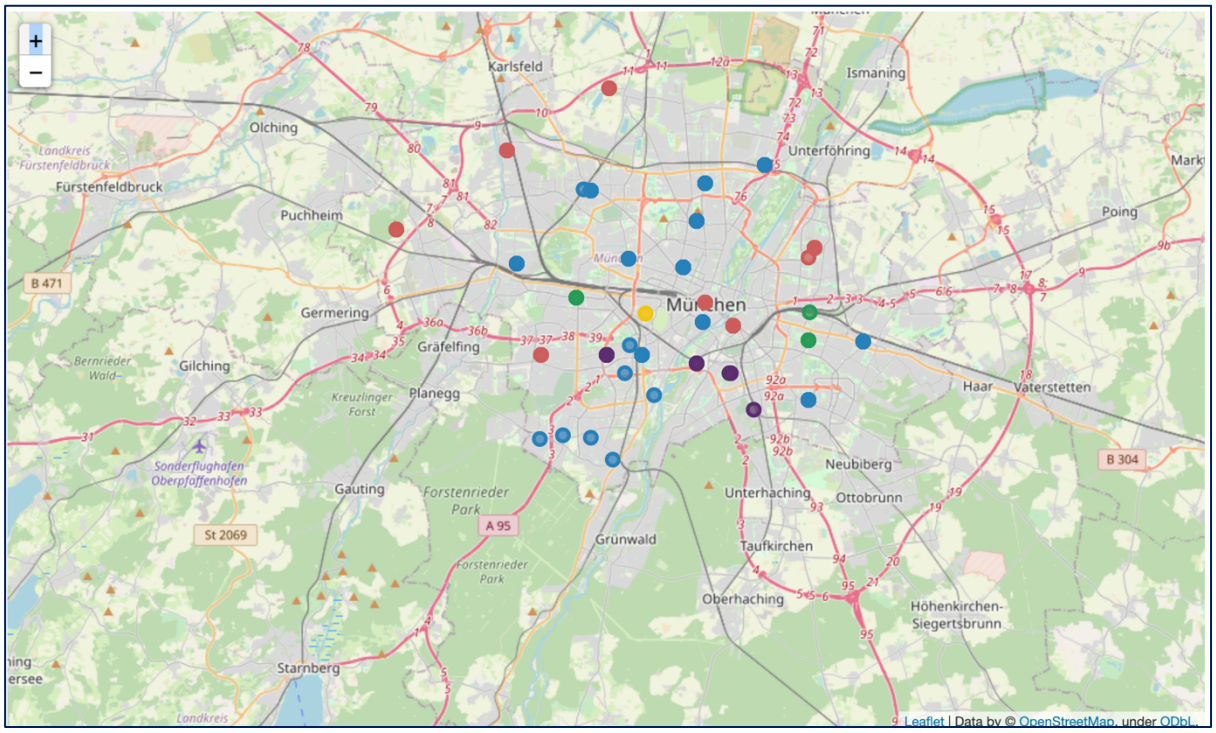

Figure 4-9 -Map of Clustered neighbourhood of Munich

4.9 Examining Clusters

We could examine our clusters by expanding on our code using the *Cluster Labels column*:

```
cluster0 = Berlin_merged_nonan.loc[Berlin_merged_nonan['Cluster Labels'] == 0, Berlin_merged_nonan.columns[[1] + list(r
cluster0['1st Most Common Venue'].value_counts()
```

```
cluster1 = Berlin_merged_nonan.loc[Berlin_merged_nonan['Cluster Labels'] == 1, Berlin_merged_nonan.columns[[1] + list(r
cluster1['1st Most Common Venue'].value_counts()
```

```
cluster2 = Berlin_merged_nonan.loc[Berlin_merged_nonan['Cluster Labels'] == 2, Berlin_merged_nonan.columns[[1] + list(r
cluster2['1st Most Common Venue'].value_counts()
```

```
cluster3 = Berlin_merged_nonan.loc[Berlin_merged_nonan['Cluster Labels'] == 3, Berlin_merged_nonan.columns[[1] + list(r
cluster3['1st Most Common Venue'].value_counts()
```

```
cluster4 = Berlin_merged_nonan.loc[Berlin_merged_nonan['Cluster Labels'] == 4, Berlin_merged_nonan.columns[[1] + list(r
cluster4['1st Most Common Venue'].value_counts()
```

5 RESULT AND DISCUSSION

The neighbourhoods of Berlin are very multicultural. Berlin seems to take a step further in this direction by having a lot of supermarkets, restaurants, bars, coffee shops, ice cream shops, drug store and clothing store. It has a lot of shopping options too with that of the fish markets, garden centre, gaming café, dessert shop, bookstores and sporting goods shop. The main modes of transport seem to be trams and buses. For leisure, the neighbourhoods are set up to have lots of parks, zoos, operas, and historic sites. After a further exploration of the clusters, each cluster can get a most common venue. Therefore, the clusters can be named by its most common venue.

In our case, in the cluster number zero the most common venues are Supermarket and Bakery. It has the greatest number of tram stations, metro stations and bus stops. This cluster also has café, soccer ffields, climbing gym and German restaurantss. This cloud especially fit to families having kids. The cluster number three has only one shop. In the last cluster, the most common venue is supermarket.

Overall, the city of Berlin offers a multicultural, diverse and certainly an entertaining experience.

Munich is only one third the size of Berlin in terms of city area. It has a wide variety of cuisines and eateries including Italian, Currywurst joint, Asian, Chinese etc. There are a lot of hangout spots including many Restaurants and Bars. Munich has a lot of Sporting goods shop. As one can see, the blue cluster is the most common cluster in Munich and therefore Munich seems to have a lot of similar districts in the city. In our case, the cluster number zero has most Sporting goods shop. In cluster number one, drugstores and supermarkets are the most common venues and it's therefore the Drugstore+Supermarket Cluster. The cluster number two has a lot of hotels and plazas. This cluster is my favourite, since it has a lot of coffee shops. In the the cluster number three, the most common venues are Afghan restaurants. And the last cluster is smallest cluster and it has one restaurant.

6 CONCLUSION

The purpose of this project was to explore the cities of Berlin and Munich and see how attractive it is to potential migrants and tourists. We explored both the cities based on their postal codes and then extrapolated the common venues present in each of the neighbourhoods finally concluding with clustering similar neighbourhoods together.

The neighbourhoods of Berlin and Munich have more like similar venues. We could see that each of the neighbourhoods in both the cities have a wide variety of experiences to offer which is unique in its own way. The dissimilarity exists in terms of some different venues and facilities but not to a larger extent. The cultural diversity is quite evident which also gives the feeling of a sense of inclusion. Overall, it's up to the stakeholders to decide which experience they would prefer more, and which would suit their tastes better.

7 REFERENCES:

1. Munich Neighbourhood clustering using the K – means Algorithm

https://medium.com/@brus.patrick63/munich-neighborhood-clustering-using-k-means-cde98a6e3199?sk=d55c0f9773160e217f3da572bb594c60

2. IBM Capstone Project — The Battle of Neighbourhoods in Berlin: Restaurants

https://medium.com/@lyan.fu.fly/ibm-capstone-project-the-battle-of-neighborhoods-in-berlin-restaurants-dd326f1bfacb

3. A Tale of Two Cities: Clustering Neighbourhoods of London and Paris using Machine Learning

https://medium.com/analytics-vidhya/a-tale-of-two-cities-clustering-neighborhoods-of-london-and-paris-5328f69cd8b6

4. Foursquare API