Apartment Hunting in Munich

IBM Data Science - Capstone Project

Achim Peichl

Problem Description

The housing market in Munich is getting more and more expensive and so are the rents. Because of job opportunities and/or the high quality of living in Munich (the alps are nearby and also beautiful lakes) there are always new people looking out for an attractive opportunity to rent an apartment in Munich. The purpose of this project is to collect data from different sources, combine them and use them to analyze wich boroughs can be considered in attractive. In order to do that we try to find boroughs that are similar to boroughs that are considered to be popular (rent and living quality are considered to be an indication fro attractiveness).

Data Collection

For this project I have used a variety of different data sources. First I used wikipedia to get info on the boroughs/neighborhoods of Munich, then I used different sources to get the zip codes and also shape files for the boroughs and zip codes. After that I collected data for rent and living quality in Munich. The foursquare API was used to get info on all the venues available for Munich. Another source of data was the official website from the city of Munich(Crime, Population, Healthcare, Education and so on).

Munich boroughs:
https://de.wikipedia.org/wiki/Liste_der_Stadtteile_Münchens

Geodata on the boroughs:
https://www.opengov-muenchen.de/dataset/verwaltungseinheiten-der-landeshauptstadt-muenchen

Munich ZIP codes:
https://www.muenchen.de/leben/service/postleitzahlen.html

Geodata on the ZIP codes:
https://public.opendatasoft.com/explore/dataset/postleitzahlen-deutschland/table

Rent Data:
https://www.miet-check.de/mietpreise/plz/muenchen/6562/

Living quality:
https://suedbayerische-immobilien.de/Wohnqualitaet-Muenchen

Venues:
Foursquare data query via the API

Different data sets on healthcare, population, education, crime and so on:
https://www.opengov-muenchen.de

Methodology

At first I created a pandas dataframe containing the boroughs of Munich, Germany, and their neighborhoods and ZIP codes. After that I retrieved the geodata for the boroughs and ZIP codes and added them to the data frame.
For the rent data created a new data frame with the purpose of matching the data on rent and the ZIP codes. To achieve this I first create a dictionary containing the rent as the value and the ZIP code as the key. I then used that dictionary to replace the ZIP code values in a dataframe with the rent values for the corresponding ZIP code. By connecting the ZIP codes to the boroughs we we can then calculate the average rent for each Borough.
In the next step I created a CSV file containing info on the living quality of each borough and then loaded that CSV file into a data frame.
Then I used the foursquare API to collect data on the venues in all of the ZIP codes of Munich. This data was then aggregated on the borough level.

A first try at clustering the ZIP codes (and subsequently the boroughs) did not return the desired results, as in the clusters did not correlate in any (noticeable)

form with either the data on rent or the living quality, so I decided to feed more data into the clustering algorithm.

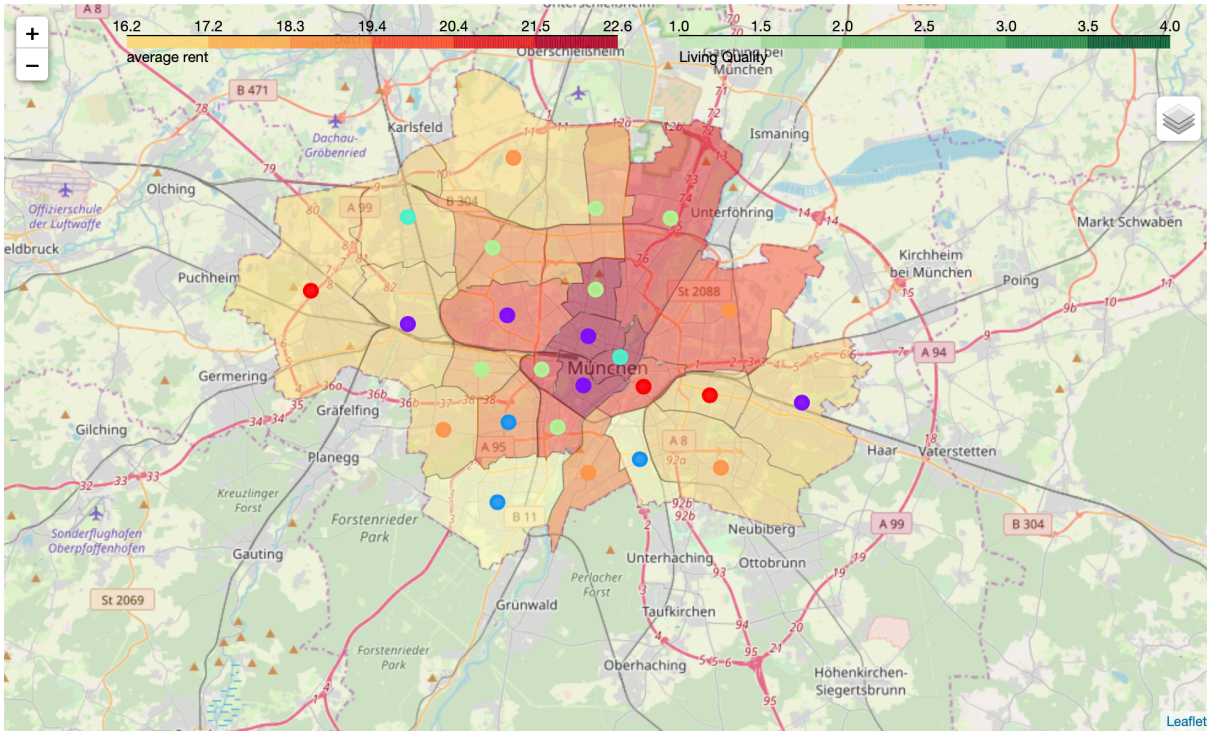
Additional data was collected from the official website for the city of Munich. This data included as mentioned above, healthcare, population, education, Crime and so on. I wrote a function to clean the data (provided by CSV files) and join it into one dataframe.

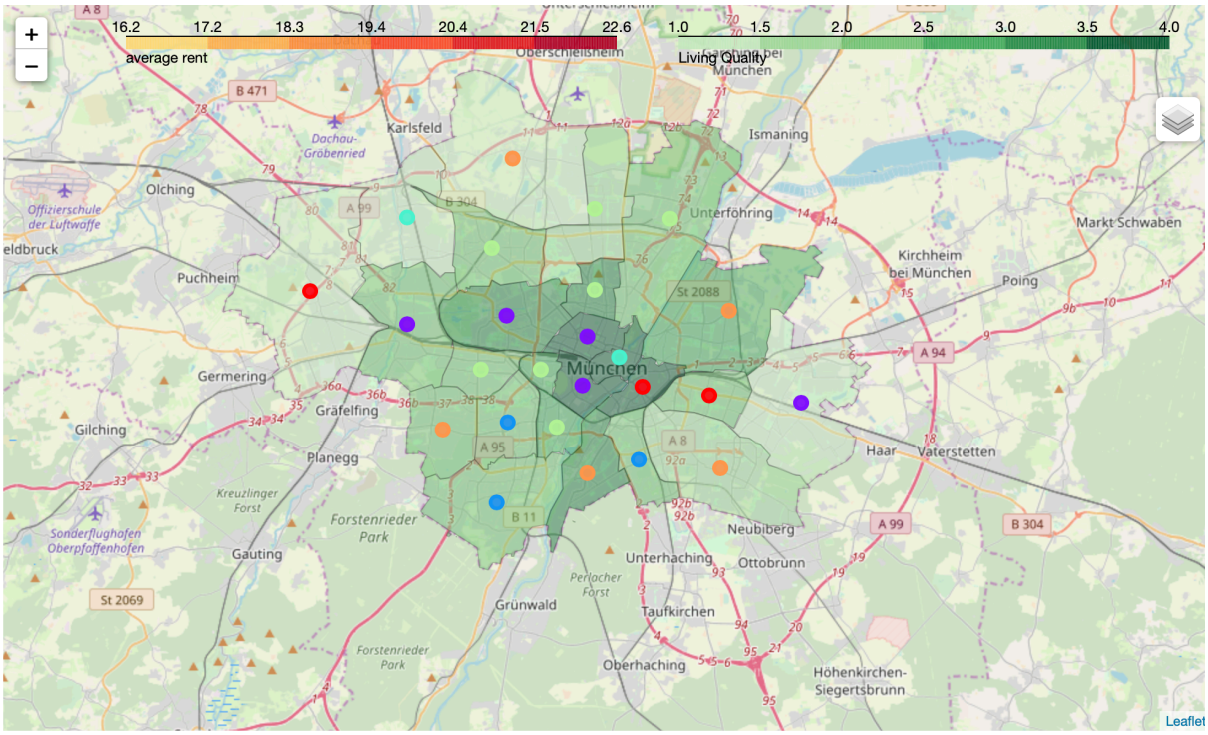
```
def readclean_csv(path, characteristic='gesamt'):
    dataframe=pd.read_csv(path) #read the csv into a pandas dataframe
    dataframe=dataframe[dataframe[,Jahr']==dataframe['Jahr'].max()]#select the data for the
most recent year
    dataframe=dataframe.loc[dataframe['Räumliche Gliederung’]!='Stadt München'] #only get the
data for the boroughs
    #get the borough numbers from the string and then convert it to an integer
    dataframe['Räumliche Gliederung']=dataframe['Räumliche Gliederung'].map(lambda x: str(x)
[:2])
    dataframe['Räumliche Gliederung']=pd.to_numeric(dataframe['Räumliche Gliederung'])
    dataframe=dataframe[dataframe['Ausprägung']==characteristic] # filter for the desired
characteristic
    dataframe.drop(dataframe.columns[drop_columns], axis=1, inplace=True)# drop columns that
are not important
    dataframe.reset_index(drop=True, inplace=True)# the index of all dataframes is supposed
to start with 0
    # create a dataframe with the consolidated (usefull) data off all the processed csv files
    df_allcsv[dataframe.iat[0,0]]=dataframe['Indikatorwert'] # add a new column named after
the characterеistic/indicator of the dataframe and fill the column with the indicator value

return dataframe
```

With the new data (including the clustering labels from the first attempt as one characteristic) I again used k-means clustering, this time only on the borough level, and then used folium to visualize the results and compare the clusters to the data on rent and living quality (I used choropleth maps to visualize rent and quality of living, both as an individual layer).

Results

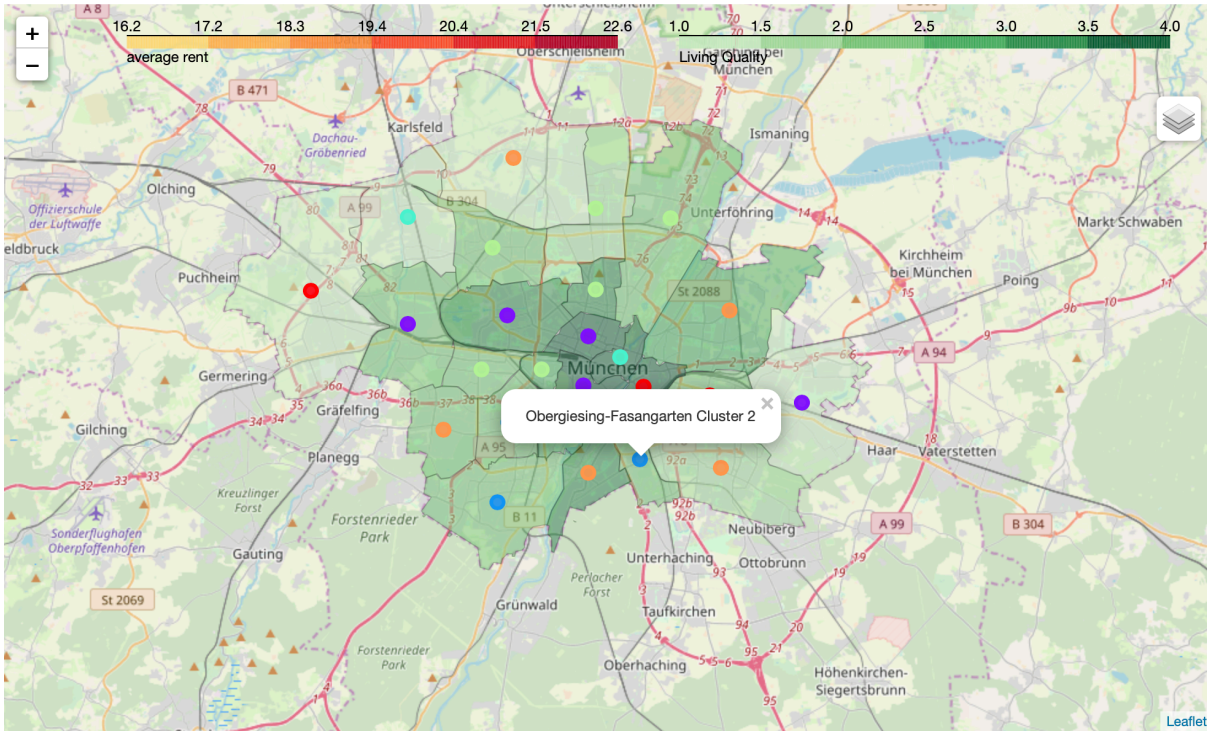
Below you can find two maps. One with the clusters and the rent data and one with the clusters and the quality of living data.





Discussion and Conclusion

From comparing the clusters to the living quality and rent data I think that borough number 2 Obergiesing-Fasangarten might be a good recommendation for anyone that move to Munich and looks for an, right now, undervalued neighbourhood.



Since this is one of my first tries at this subject I am quite sure that sometimes I there would have been a faster/better way to achieve the same results as I did. Please feel welcome to post any suggestions you have on how to improve a future project.