Investigation on Population Migration within Cologne

Jupyter Notebook in Python Language for pursuing the Analysis

All data is gathered from https://www.offenedaten-koeln.de (https://www.offenedaten-koeln.de)

For best experience, please use the Firefox Browser

Notebook Preparations

First, we need to import the necessary libraries

In [1]:
```
# Data tables will be handled by the pandas library
import pandas as pd


# The source data will be given in (geo-)json format.
# We need the json library to quickly parse the raw data into pytho
n dictionaries
import json

# Import libraries for visalization
import folium                                   # Map visualizations
import matplotlib.cm as cm                      # Colormaps
import matplotlib.colors as colors
import matplotlib.pyplot as plt                 # Plotting
import seaborn as sns

# Import libraries for html requests
import requests                                 # Will be used to request
data from Foursquare
import geocoder                                 # Will be used to locate
cologne WGS84 coordinates
from geopy.geocoders import Nominatim           # Will be used to locate
cologne WGS84 coordinates


# Data handling and ML algorithms
import numpy as np
from sklearn.cluster import KMeans              # KMeans Algorithm
from sklearn import tree                        # Decision Tree Algorithm
import sklearn.metrics                          # Evaluation
from sklearn import preprocessing               # Data Preparation
from sklearn.model_selection import train_test_split
```

(1) Get Venue Data for Cologne's Districts

First, we want to perform a similar analysis like we did in the previous capstone project assignments for NY and Toronto. Therefore, we need a data set containing Cologne's districts with center points around which we can search for venues by using Foursquare API calls.

Data Preprocessing

In [2]:
```
# Open GeoJson File of Cologne
with open('bezirke.json', encoding='utf8') as f:
    data = json.load(f)
```

In [3]:
```
# What attributes are available:
print(data['features'][0]['attributes'].keys())
```

```
dict_keys(['OBJECTID', 'NUMMER', 'NAME', 'NR_STADTBEZIRK', 'STADTB
EZIRK', 'FLAECHE', 'LINK'])
```

We see that the file is given in `geojson` format. In the attribute's section of each item, we see several keys. Here, we will use `NAME` which gives us the name of the district.
Let's look at the districts given in the file:

In [4]:
```
# Look at all the neighborhoods
print([x['attributes']['NAME'] for x in data['features']])
```

```
['Godorf', 'Lövenich', 'Weiden', 'Junkersdorf', 'Widdersdorf', 'Vo
gelsang', 'Weidenpesch', 'Mauenheim', 'Marienburg', 'Bayenthal', '
Weiß', 'Hahnwald', 'Meschenich', 'Immendorf', 'Buchforst', 'Höhenb
erg', 'Vingst', 'Grengel', 'Elsdorf', 'Wahnheide', 'Wahn', 'Libur'
, 'Lind', 'Ensen', 'Eil', 'Raderthal', 'Raderberg', 'Rondorf', 'Ho
lweide', 'Buchheim', 'Merheim', 'Neubrück', 'Ostheim', 'Westhoven'
, 'Bocklem./Mengenich', 'Dünnwald', 'Höhenhaus', 'Poll', 'Humboldt
/Gremb.', 'Deutz', 'Bickendorf', 'Sürth', 'Rodenkirchen', 'Urbach'
, 'Niehl', 'Flittard', 'Dellbrück', 'Rath/Heumar', 'Brück', 'Altst
adt-Nord', 'Neustadt-Süd', 'Müngersdorf', 'Roggendorf/Thenh.', 'Zü
ndorf', 'Langel', 'Zollstock', 'Porz', 'Altstadt-Süd', 'Neustadt-N
ord', 'Kalk', 'Braunsfeld', 'Ossendorf', 'Neuehrenfeld', 'Esch/Auw
eiler', 'Pesch', 'Volkhoven/Weiler', 'Heimersdorf', 'Chorweiler',
'Seeberg', 'Lindweiler', 'Worringen', 'Riehl', 'Nippes', 'Lindenth
al', 'Sülz', 'Klettenberg', 'Bilderstöckchen', 'Longerich', 'Gremb
erghoven', 'Ehrenfeld', 'Finkenberg', 'Merkenich', 'Fühlingen', 'B
lumenberg', 'Stammheim', 'Mülheim']
```

Next, we want to retrieve the center points of each district. Unfortunately, they are not given in the file. What we have, however, is the polygon edges of each district. For instance, let's look at the first 5 points of of district `Nippes`:

In [5]:
```
[x['geometry']['rings'] for x in data['features'] if x['attributes'
]['NAME'] == 'Nippes'][0][0][0:5]
```

Out[5]:
```
[[6.955074605831894, 50.972575128345596],
 [6.9550738975575355, 50.972568391333986],
 [6.955071673136261, 50.97256130939216],
 [6.955068646161914, 50.972555310629446],
 [6.955061343520813, 50.97254638057947]]
```

The following function computes the centroid of a polygon given its edge points:

In [6]:
```
# Define a function to find a centroid from a polygon
def centroid(vertexes):
     _x_list = [vertex [0] for vertex in vertexes]
     _y_list = [vertex [1] for vertex in vertexes]
     _len = len(vertexes)
     _x = sum(_x_list) / _len
     _y = sum(_y_list) / _len
     return(_x, _y)
```

Let's use that function to find the centroid for `Nippes`:

In [7]:
```
# For example: center of district 'Nippes'
district = 'Nippes'
centroid([x['geometry']['rings'] for x in data['features'] if x['at
tributes']['NAME'] ==district][0][0])
```

Out[7]:
```
(6.9570853589869674, 50.96677225268491)
```

Let's generalize that procedure to find the centroid for a given district name:

In [8]:
```
def get_centroid_for_district(name):
    return centroid([x['geometry']['rings'] for x in data['features
'] if x['attributes']['NAME'] == name][0][0])
```

Now, we have all ingredients to create a pandas data frame from the json data. We want to extract each district's name, center latitude, center longitude and the list of polygon edge points.

In [9]:
```
# Create a new data frame
df_cologne = pd.DataFrame(columns=['District', 'Latitude', 'Longitude', 'Polygon'])

# Extract the information from the file
df_cologne['District'] = [x['attributes']['NAME'] for x in data['features']]
df_cologne['Latitude'] = df_cologne['District'].apply(get_centroid_for_district).apply(lambda i: i[1])
df_cologne['Longitude'] = df_cologne['District'].apply(get_centroid_for_district).apply(lambda i: i[0])
df_cologne['Polygon'] = df_cologne['District'].apply(lambda d: [x['geometry']['rings'] for x in data['features'] if x['attributes']['NAME'] == d][0][0])
df_cologne.head()
```

Out[9]:

	District	Latitude	Longitude	Polygon
0	Godorf	50.852610	6.982218	[[6.994359341598065, 50.85835989408827], [6.99...
1	Lövenich	50.948533	6.823829	[[6.835101297207516, 50.957260493877406], [6.8...
2	Weiden	50.934514	6.824246	[[6.849501526802686, 50.94220412958467], [6.84...
3	Junkersdorf	50.923891	6.859688	[[6.854198181651118, 50.94052346246431], [6.85...
4	Widdersdorf	50.967660	6.841605	[[6.851763526592514, 50.97718504868292], [6.85...

Creating a Cologne Map

Now that we have the districts' positions, let's create a request to Nominatim to get Cologne's center point in order to center the map around it:

In [10]:
```
address = 'Cologne, DE'

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

```
The geograpical coordinate of Cologne are 50.938361, 6.959974.
```

Next, we create a map centered around Cologne with all the districts' center points from the data frame created above.

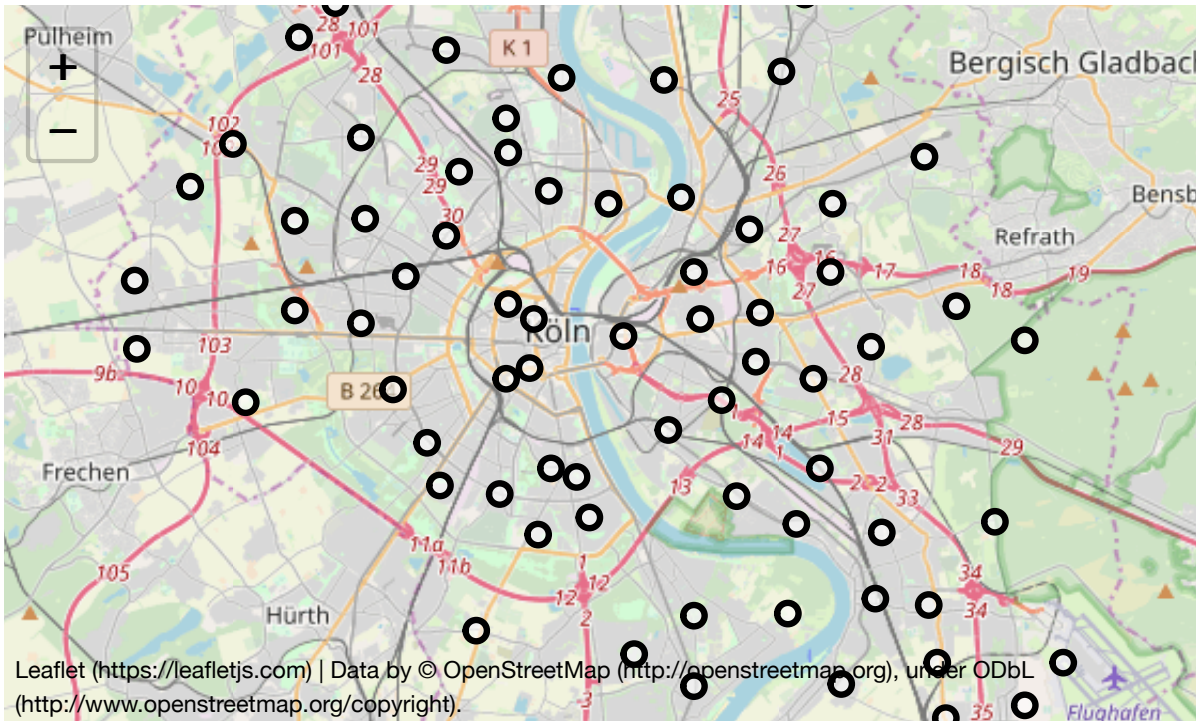
In [11]:
```
# create map and display it
cologne_map = folium.Map(location=[latitude, longitude], zoom_start
=12)

# instantiate a feature group for the incidents in the dataframe
districts = folium.map.FeatureGroup()

# loop through the 100 crimes and add each to the incidents feature
group
for lat, lng, label in zip(df_cologne.Latitude, df_cologne.Longitud
e, df_cologne.District):
    districts.add_child(
        folium.CircleMarker(
            [lat, lng],
            radius=5, # define how big you want the circle markers
to be
            color='black',
            fill=True,
            fill_color='white',
            fill_opacity=0.6,
            popup=label
        )
    )

# add incidents to map
cologne_map.add_child(districts)
```

Out[11]:



Explore the Districts with Foursquare API

The map above looks good. Now, we want to get venue data for each district. Therefore, similar to the analyses done before, we create requests for each district's location to Foursquare using our credentials.

In [12]:
```
CLIENT_ID = 'K0OBBE3XME1ORY4POM5AWPU1BI01AGX01IIDL5EL4D005UA4' # yo
ur Foursquare ID
CLIENT_SECRET = 'APU0I3ENTYA0Z2NHBT1T3RREEH2GBPRF5AHSFYCZVB4TCU1O'
# your Foursquare Secret
VERSION = '20180605' # Foursquare API version
```

We recycle the function from the assignment to get venues in a certain radius around a location:

In [13]:
```
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={}&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_lis
t for item in venue_list])
    nearby_venues.columns = ['Neighborhood',
                  'Neighborhood Latitude',
                  'Neighborhood Longitude',
                  'Venue',
                  'Venue Latitude',
                  'Venue Longitude',
                  'Venue Category']

    return(nearby_venues)
```

Let's perform the request with a limit of 100 venues and a radius of 500:

In [14]:
```
LIMIT = 100
cologne_venues = getNearbyVenues(names=df_cologne['District'],
                                 latitudes=df_cologne['Latitude']
,
                                 longitudes=df_cologne['Longitude
'],
                                 radius=500)
```

```
Godorf
Lövenich
Weiden
Junkersdorf
Widdersdorf
Vogelsang
Weidenpesch
Mauenheim
Marienburg
Bayenthal
Weiß
Hahnwald
Meschenich
Immendorf
Buchforst
Höhenberg
Vingst
Grengel
Elsdorf
Wahnheide
Wahn
Libur
Lind
Ensen
Eil
Raderthal
Raderberg
Rondorf
Holweide
Buchheim
Merheim
Neubrück
Ostheim
Westhoven
Bocklem./Mengenich
Dünnwald
Höhenhaus
Poll
Humboldt/Gremb.
Deutz
Bickendorf
Sürth
Rodenkirchen
Urbach
Niehl
Flittard
Dellbrück
Rath/Heumar
Brück
```

```
Altstadt-Nord
Neustadt-Süd
Müngersdorf
Roggendorf/Thenh.
Zündorf
Langel
Zollstock
Porz
Altstadt-Süd
Neustadt-Nord
Kalk
Braunsfeld
Ossendorf
Neuehrenfeld
Esch/Auweiler
Pesch
Volkhoven/Weiler
Heimersdorf
Chorweiler
Seeberg
Lindweiler
Worringen
Riehl
Nippes
Lindenthal
Sülz
Klettenberg
Bilderstöckchen
Longerich
Gremberghoven
Ehrenfeld
Finkenberg
Merkenich
Fühlingen
Blumenberg
Stammheim
Mülheim
```

The venues are saved in the following data frame:

In [15]:
```
cologne_venues.head()
```

Out[15]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Ver Categ
0	Godorf	50.852610	6.982218	Godorf Hafen	50.848477	6.982683	Harb Mar
1	Godorf	50.852610	6.982218	Bude Pausenraum	50.852996	6.988761	Breakf S
2	Godorf	50.852610	6.982218	Godorfer Imbiss	50.852567	6.975184	Sna Pla
3	Lövenich	50.948533	6.823829	Bäckerei Kraus - Lövenicher Café	50.946507	6.829737	Bak
4	Lövenich	50.948533	6.823829	Alte Schmiede	50.946739	6.829642	Gastrop

As a next step, we convert the table in a one hot encoded table using pandas' `get_dummies` method. This makes it possible to apply machine learning algorithms on categorical data.

In [16]:
```
# Clustering to see similar districts in cologne
# one hot encoding
cologne_onehot = pd.get_dummies(cologne_venues[['Venue Category']],
prefix="", prefix_sep="")

# add neighborhood column back to dataframe
cologne_onehot['Neighborhood'] = cologne_venues['Neighborhood']

# move neighborhood column to the first column
fixed_columns = [cologne_onehot.columns[-1]] + list(cologne_onehot.
columns[:-1])
cologne_onehot = cologne_onehot[fixed_columns]

cologne_onehot.head()
```

Out[16]:

	Neighborhood	African Restaurant	Airport Service	Argentinian Restaurant	Art Gallery	Art Museum	Arts & Crafts Store	Asian Restaurant	A &
0	Godorf	0	0	0	0	0	0	0	
1	Godorf	0	0	0	0	0	0	0	
2	Godorf	0	0	0	0	0	0	0	
3	Lövenich	0	0	0	0	0	0	0	
4	Lövenich	0	0	0	0	0	0	0	

5 rows × 192 columns

We group the data by district and use the mean method to get a measure for the ratio of venues of a certain category in that district:

In [17]:
```
cologne_grouped = cologne_onehot.groupby('Neighborhood').mean().reset_index()
cologne_grouped.head()
```

Out[17]:

	Neighborhood	African Restaurant	Airport Service	Argentinian Restaurant	Art Gallery	Art Museum	Arts & Crafts Store	Asian Restaurant
0	Altstadt-Nord	0.0	0.01	0.0	0.0	0.020000	0.01	0.010000
1	Altstadt-Süd	0.0	0.00	0.0	0.0	0.026316	0.00	0.026316
2	Bayenthal	0.0	0.00	0.0	0.0	0.000000	0.00	0.000000
3	Bickendorf	0.0	0.00	0.0	0.0	0.000000	0.00	0.000000
4	Bilderstöckchen	0.0	0.00	0.0	0.0	0.000000	0.00	0.000000

5 rows × 192 columns

Let's now - as an additional information - look at the top 3 venues in each district:

In [18]:
```
num_top_venues = 3

for hood in cologne_grouped['Neighborhood']:
    print("{:-^30}".format(hood))
    temp = cologne_grouped[cologne_grouped['Neighborhood'] == hood].T.reset_index()
    temp.columns = ['venue','freq']
    temp = temp.iloc[1:]
    temp['freq'] = temp['freq'].astype(float)
    temp = temp.round({'freq': 2})
    print(temp.sort_values('freq', ascending=False).reset_index(drop=True).head(num_top_venues))
    print('\n')
```

```
--------Altstadt-Nord---------
                venue  freq
0  Italian Restaurant  0.07
1               Café  0.06
2             Bakery  0.05


---------Altstadt-Süd---------
              venue  freq
0             Hotel  0.16
1  Sushi Restaurant  0.08
2              Café  0.08


----------Bayenthal-----------
```

```
         venue  freq
0  Supermarket  0.16
1       Bakery  0.08
2     Bus Stop  0.08


----------Bickendorf----------
         venue  freq
0          Pub  0.17
1  Supermarket  0.17
2         Bank  0.08


-------Bilderstöckchen--------
       venue  freq
0   Bus Stop  0.50
1  Drugstore  0.25
2       Café  0.25


------Bocklem./Mengenich------
              venue  freq
0  Greek Restaurant  0.25
1      Intersection  0.25
2      Soccer Field  0.25


----------Braunsfeld----------
               venue  freq
0             Bakery  0.18
1          Drugstore  0.12
2  German Restaurant  0.12


------------Brück-------------
                venue  freq
0              Forest   0.5
1              Bakery   0.5
2  African Restaurant   0.0


----------Buchforst-----------
          venue  freq
0         Hotel  0.33
1  Tram Station  0.17
2   Supermarket  0.17


-----------Buchheim-----------
              venue  freq
0  Greek Restaurant  0.25
1       Supermarket  0.25
2     Big Box Store  0.25


----------Chorweiler----------
               venue  freq
```

```
0  Electronics Store   0.2
1         Restaurant   0.2
2        Supermarket   0.2


----------Dellbrück-----------
                venue  freq
0  Italian Restaurant   0.2
1           Drugstore   0.1
2              Bakery   0.1


------------Deutz-------------
             venue  freq
0  Ice Cream Shop  0.05
1          Bakery  0.05
2          Bistro  0.05


-----------Dünnwald-----------
           venue  freq
0     Restaurant  0.33
1  Shopping Mall  0.33
2           Park  0.33


----------Ehrenfeld-----------
         venue  freq
0         Café  0.09
1          Bar  0.09
2  Supermarket  0.06


-------------Eil--------------
                venue  freq
0  Athletics & Sports   1.0
1  African Restaurant   0.0
2        Perfume Shop   0.0


-----------Elsdorf------------
                         venue  freq
0  Construction & Landscaping   1.0
1       Portuguese Restaurant   0.0
2                Optical Shop   0.0


------------Ensen-------------
         venue  freq
0  Gas Station  0.25
1        River  0.25
2       Bakery  0.25


----------Finkenberg----------
               venue  freq
0               Café   0.1
```

```
1  Food & Drink Shop   0.1
2                Gym   0.1


-----------Flittard-----------
          venue  freq
0    Food Truck  0.25
1      Bus Stop  0.25
2  Soccer Field  0.25


------------Godorf------------
              venue  freq
0      Snack Place  0.33
1   Breakfast Spot  0.33
2  Harbor / Marina  0.33


--------Gremberghoven---------
                venue  freq
0  Light Rail Station   1.0
1  African Restaurant   0.0
2        Perfume Shop   0.0


-----------Grengel------------
               venue  freq
0   Airport Service   0.5
1             Plane   0.5
2  Pedestrian Plaza   0.0


-----------Hahnwald-----------
                venue  freq
0           Racetrack   1.0
1  African Restaurant   0.0
2        Perfume Shop   0.0


---------Heimersdorf----------
                venue  freq
0          Restaurant   1.0
1  African Restaurant   0.0
2        Perfume Shop   0.0


-----------Holweide-----------
          venue  freq
0  Tram Station   0.5
1      Bus Stop   0.5
2  Perfume Shop   0.0


-------Humboldt/Gremb.--------
              venue  freq
0        Pet Store  0.25
1  Organic Grocery  0.25
```

```
2    Big Box Store  0.25


----------Höhenberg-----------
            venue  freq
0            Pool  0.14
1  Discount Store  0.14
2     Supermarket  0.14


----------Höhenhaus-----------
               venue  freq
0       Tram Station   0.5
1  Outdoor Sculpture   0.5
2          Nightclub   0.0


---------Junkersdorf----------
                     venue  freq
0              Supermarket  0.17
1  Furniture / Home Store  0.17
2                  Bakery  0.08


-------------Kalk-------------
                venue  freq
0              Bakery  0.14
1    Doner Restaurant  0.09
2  Turkish Restaurant  0.05


---------Klettenberg----------
                  venue  freq
0        Train Station   0.2
1         Dessert Shop   0.2
2  Rental Car Location   0.2


------------Langel------------
            venue  freq
0             Bar   0.2
1  Ice Cream Shop   0.2
2          Market   0.2


------------Libur-------------
                venue  freq
0         Golf Course   1.0
1  Israeli Restaurant   0.0
2              Office   0.0


-------------Lind-------------
              venue  freq
0  Business Service  0.33
1          Bus Stop  0.33
2           Theater  0.33
```

```
----------Lindenthal----------
               venue  freq
0  German Restaurant  0.17
1         Restaurant  0.17
2               Café  0.08


----------Lindweiler----------
                venue  freq
0         Flower Shop   1.0
1  African Restaurant   0.0
2        Perfume Shop   0.0


----------Longerich----------
          venue  freq
0  Tram Station   0.4
1   Supermarket   0.2
2      Hospital   0.2


-----------Lövenich-----------
       venue  freq
0        Bar  0.25
1   Bus Stop  0.25
2  Gastropub  0.25


----------Marienburg----------
           venue  freq
0  Grocery Store   0.2
1      Wine Shop   0.2
2       Bus Stop   0.2


----------Mauenheim-----------
               venue  freq
0        Supermarket  0.33
1  German Restaurant  0.17
2       Soccer Field  0.17


-----------Merheim------------
                venue  freq
0   German Restaurant   0.5
1                Park   0.5
2  African Restaurant   0.0


----------Merkenich-----------
                   venue  freq
0                  Café   1.0
1     African Restaurant   0.0
2  Portuguese Restaurant   0.0
```

```
----------Meschenich----------
                venue  freq
0             Brewery   0.5
1             Stables   0.5
2  African Restaurant   0.0


-----------Mülheim-----------
         venue  freq
0         Café  0.15
1  Supermarket  0.15
2    BBQ Joint  0.08


---------Müngersdorf----------
                venue  freq
0               Café  0.25
1  German Restaurant  0.25
2        Pizza Place  0.25


-----------Neubrück-----------
                venue  freq
0           Drugstore  0.25
1  Convenience Store  0.25
2              Plaza  0.25


---------Neuehrenfeld---------
                venue  freq
0  Italian Restaurant   0.1
1         Supermarket   0.1
2          Restaurant   0.1


--------Neustadt-Nord---------
              venue  freq
0             Hotel  0.10
1  Sushi Restaurant  0.07
2               Bar  0.05


---------Neustadt-Süd---------
                venue  freq
0  Italian Restaurant  0.09
1                 Bar  0.09
2             Bakery  0.05


------------Niehl------------
                venue  freq
0  German Restaurant   0.2
1       Intersection   0.2
2     Ice Cream Shop   0.2
```

```
------------Nippes------------
                venue  freq
0                Café  0.15
1  Italian Restaurant  0.09
2                 Bar  0.09


----------Ossendorf-----------
              venue  freq
0         Racetrack  0.11
1  Indoor Play Area  0.11
2     Historic Site  0.11


-----------Ostheim------------
              venue  freq
0       Supermarket  0.75
1      Tram Station  0.25
2  Pedestrian Plaza  0.00


------------Pesch-------------
                venue  freq
0           Steakhouse  0.14
1  Italian Restaurant  0.14
2         Supermarket  0.14


-------------Poll-------------
            venue  freq
0    Tram Station  0.18
1       Drugstore  0.09
2  Ice Cream Shop  0.09


-------------Porz-------------
                venue  freq
0  Turkish Restaurant  0.12
1       Train Station  0.06
2           BBQ Joint  0.06


----------Raderberg-----------
                  venue  freq
0           Supermarket  0.31
1  Gym / Fitness Center  0.12
2             Drugstore  0.06


----------Raderthal-----------
              venue  freq
0  Greek Restaurant  0.25
1        Playground  0.25
2              Park  0.25


------------Riehl-------------
```

```
         venue  freq
0  Zoo Exhibit  0.62
1       Bakery  0.04
2   Playground  0.04


---------Rodenkirchen---------
          venue  freq
0          Pool   1.0
1  Perfume Shop   0.0
2  Optical Shop   0.0


-----------Rondorf------------
                venue  freq
0    Insurance Office   1.0
1  African Restaurant   0.0
2        Perfume Shop   0.0


-----------Seeberg------------
                venue  freq
0  Turkish Restaurant  0.50
1      Discount Store  0.25
2  Light Rail Station  0.25


----------Stammheim-----------
            venue  freq
0     Supermarket  0.25
1  Ice Cream Shop  0.25
2        Pharmacy  0.25


-------------Sülz-------------
                  venue  freq
0                Bakery  0.14
1  Vietnamese Restaurant  0.14
2                   Pub  0.07


------------Sürth-------------
               venue  freq
0       Tram Station   0.2
1        Supermarket   0.2
2  German Restaurant   0.2


------------Urbach------------
               venue  freq
0          Drugstore  0.11
1        Pizza Place  0.11
2  Electronics Store  0.11


------------Vingst------------
           venue  freq
```

```
0    Supermarket  0.31
1         Bakery  0.23
2  Metro Station  0.08


----------Vogelsang-----------
                venue  freq
0              Lawyer   0.5
1              Bakery   0.5
2  African Restaurant   0.0


-------------Wahn-------------
         venue  freq
0     Platform  0.27
1  Supermarket  0.18
2        Hotel  0.09


----------Wahnheide-----------
                venue  freq
0  Italian Restaurant  0.33
1               Hotel  0.33
2    Greek Restaurant  0.17


------------Weiden------------
               venue  freq
0  German Restaurant   0.5
1      Historic Site   0.5
2             Office   0.0


---------Weidenpesch----------
               venue  freq
0   Greek Restaurant  0.25
1  Korean Restaurant  0.25
2     Ice Cream Shop  0.25


----------Westhoven-----------
                  venue  freq
0      Business Service  0.25
1  Gym / Fitness Center  0.25
2           Gas Station  0.25


---------Widdersdorf----------
          venue  freq
0    Food Truck  0.33
1   Supermarket  0.17
2  Soccer Field  0.17


----------Worringen-----------
                venue  freq
0         Supermarket  0.25
```

```
1             Taverna  0.25
2  Photography Studio  0.25


----------Zollstock----------
                         venue  freq
0            German Restaurant  0.29
1               Ice Cream Shop  0.29
2  Eastern European Restaurant  0.14


----------Zündorf----------
                 venue  freq
0          Supermarket  0.25
1  Italian Restaurant  0.12
2         Soccer Field  0.12
```

The following function generalizes this behavior and gives us the top *x* venues in a certain district. This will then be used to get for each district the top *x* venues in a data table.

In [19]:
```
def return_most_common_venues(row, num_top_venues):
    row_categories = row.iloc[1:]
    row_categories_sorted = row_categories.sort_values(ascending=Fa
lse)

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

In [20]:
```
num_top_venues = 10

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, indic
ators[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'] = cologne_grouped['Neig
hborhood']

for ind in np.arange(cologne_grouped.shape[0]):
    neighborhoods_venues_sorted.iloc[ind, 1:] = return_most_common_
venues(cologne_grouped.iloc[ind, :], num_top_venues)

neighborhoods_venues_sorted[neighborhoods_venues_sorted.Neighborhoo
d == "Nippes"]
```

Out[20]:

	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 Com V
51	Nippes	Café	Italian Restaurant	Bar	German Restaurant	Supermarket	Bakery	

Now we can see, that for instance in Nippes, there are many places for meals and drinks.

Clustering of Venues

In the next step, we try to find similarities between districts according to their most common venues. For this, we use k-means clustering.

In [21]:
```
# set number of clusters
kclusters = 10

cologne_grouped_clustering = cologne_grouped.drop('Neighborhood', 1
)
for i in cologne_grouped_clustering.columns:
    cologne_grouped_clustering[i] = preprocessing.minmax_scale(colo
gne_grouped_clustering[i].astype('float64'))

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

The kmeans algorithm has no assigned a cluster id to each district. Let's merge this data with the original data frame:

In [22]:
```
# add clustering labels
neighborhoods_venues_sorted.insert(0, 'Cluster Labels', kmeans.labe
ls_)

cologne_merged = df_cologne

# merge toronto_grouped with toronto_data to add latitude/longitude
for each neighborhood
cologne_merged = cologne_merged.join(neighborhoods_venues_sorted.se
t_index('Neighborhood'), on='District')

cologne_merged
```

Out[22]:

	District	Latitude	Longitude	Polygon	Cluster Labels	1st Most Common Venue	2nd Mo Comm Ven
0	Godorf	50.852610	6.982218	[[6.994359341598065, 50.85835989408827], [6.99...	3.0	Snack Place	Breakfa Sp
1	Lövenich	50.948533	6.823829	[[6.835101297207516, 50.957260493877406], [6.8...	3.0	Bus Stop	B
2	Weiden	50.934514	6.824246	[[6.849501526802686, 50.94220412958467], [6.84...	6.0	German Restaurant	Historic Si
3	Junkersdorf	50.923891	6.859688	[[6.854198181651118, 50.94052346246431], [6.85...	3.0	Furniture / Home Store	Supermark
4	Widdersdorf	50.967660	6.841605	[[6.851763526592514, 50.97718504868292], [6.85...	1.0	Food Truck	Soccer Fie
...	...	...	...	...	...	...	
81	Merkenich	51.033502	6.940967	[[6.903330233923989, 51.06204852036002], [6.91...	3.0	Café	Zoo Exhil
82	Fühlingen	51.033309	6.912747	[[6.914875435436402, 51.03564778930263], [6.91...	NaN	NaN	Na
83	Blumenberg	51.040677	6.873809	[[6.888760590383288, 51.046465060907174], [6.8...	NaN	NaN	Na
84	Stammheim	50.989277	6.993882	[[7.010640423873277, 50.99586620162058], [7.01...	6.0	Park	Supermark
85	Mülheim	50.965368	6.999677	[[7.009698154969715, 50.98762355280711], [7.01...	6.0	Café	Supermark

86 rows × 15 columns

For some districts there was no data received apparently from Foursquare. Let's just remove those districts from the analysis. Also, after removing the NaNs we can convert the cluster label column to integer type.

In [23]:
```
cologne_merged.dropna(inplace=True)
cologne_merged['Cluster Labels'] = cologne_merged['Cluster Labels']
.astype(int)
```

Let's now visualize the result on a map:

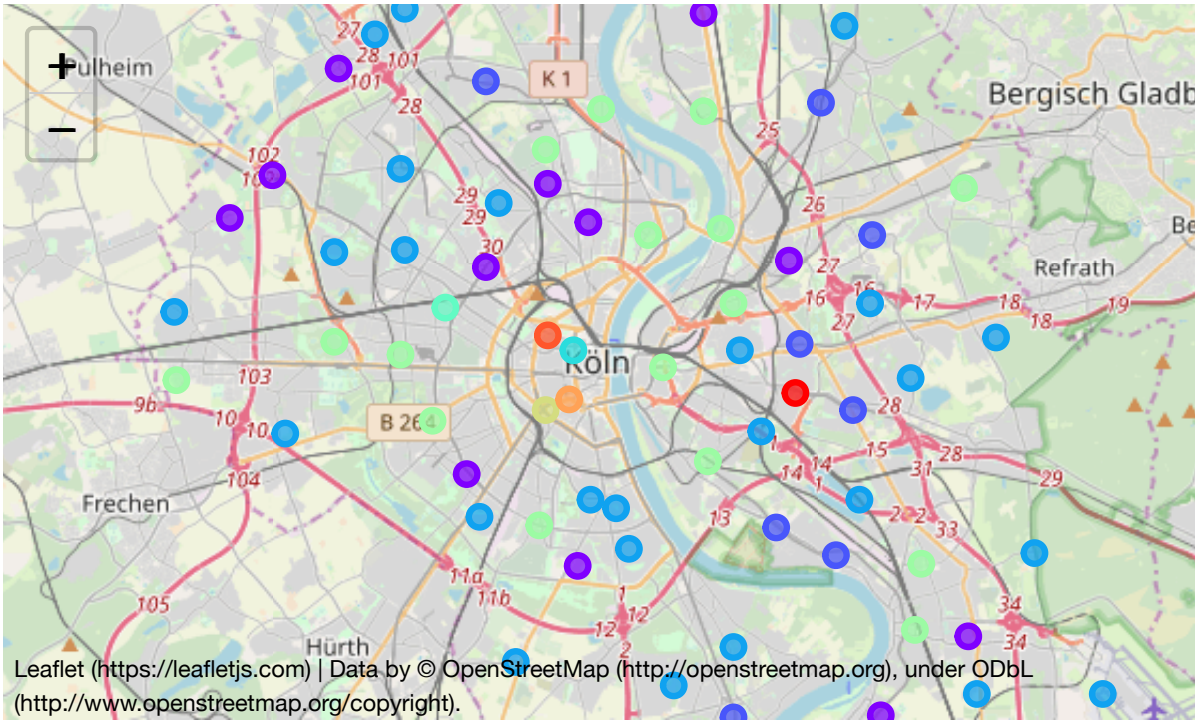
In [24]:
```
# create map
map_clusters = folium.Map(location=[latitude, longitude], zoom_start=11)

# set color scheme for the clusters
x = np.arange(kclusters)
ys = [i + x + (i*x)**2 for i in range(kclusters)]
colors_array = cm.rainbow(np.linspace(0, 1, len(ys)))
rainbow = [colors.rgb2hex(i) for i in colors_array]

# add markers to the map
markers_colors = []
for lat, lon, poi, cluster in zip(cologne_merged['Latitude'], cologne_merged['Longitude'], cologne_merged['District'], cologne_merged['Cluster Labels']):
    label = folium.Popup(str(poi) + ' Cluster ' + str(cluster), parse_html=True)
    folium.CircleMarker(
        [lat, lon],
        radius=5,
        popup=label,
        color=rainbow[cluster-1],
        fill=True,
        fill_color=rainbow[cluster-1],
        fill_opacity=0.7).add_to(map_clusters)

map_clusters
```

Out[24]:



We can see the different clusters of the Cologne venue environment. One interpretation of the result could be that in the city center the venue distribution is very heterogeneous. The clusters are not similar to any other cluster. Each district in the city center forms its own cluster and thus does not resemble any other district in the Cologne area. If we go to the outer rings, we see more similarities between districts. For instance, the districts close to but right of the Rhein river seem to be quite similar (cluster 6). They also show similarties to clusters on the left side of the rivers, in districts where industry governed. If we look at cluster 3, we again see a lot of food and drink venues but now in form of restaurants and also shops. This could be an area with many households. Cluster 1 shows many activity locations.

In [25]:
```
cologne_merged[cologne_merged['Cluster Labels'] == 3].head()
```

Out[25]:

	District	Latitude	Longitude	Polygon	Cluster Labels	1st Most Common Venue	2nd Most Common Venue
0	Godorf	50.852610	6.982218	[[6.994359341598065, 50.85835989408827], [6.99...	3	Snack Place	Breakfast Spot
1	Lövenich	50.948533	6.823829	[[6.835101297207516, 50.957260493877406], [6.8...	3	Bus Stop	Bar
3	Junkersdorf	50.923891	6.859688	[[6.854198181651118, 50.94052346246431], [6.85...	3	Furniture / Home Store	Supermarket
5	Vogelsang	50.960676	6.875571	[[6.8724234275435485, 50.97622549898243], [6.8...	3	Bakery	Lawyer
8	Marienburg	50.900442	6.969971	[[6.971316407575258, 50.90477967339803], [6.97...	3	Bus Stop	Wine Shop

In [26]:
```
cologne_merged[cologne_merged['Cluster Labels'] == 1].head()
```

Out[26]:

	District	Latitude	Longitude	Polygon	Cluster Labels	1st Most Common Venue
4	Widdersdorf	50.967660	6.841605	[[6.851763526592514, 50.97718504868292], [6.85...	1	Food Truck
7	Mauenheim	50.974533	6.944026	[[6.937528917218961, 50.977380452716204], [6.9...	1	Supermarket
25	Raderthal	50.896836	6.953965	[[6.949481603841121, 50.90524475670013], [6.94...	1	Bakery
29	Buchheim	50.958775	7.021337	[[7.033473269521677, 50.96039724052178], [7.03...	1	Greek Restaurant
34	Bocklem./Mengenich	50.976213	6.855554	[[6.866097320256192, 50.99182970235463], [6.86...	1	Intersection

This concludes the section about Cologne's venue environment analysis. There are some hints about whether or not people might want to migrate to other districts but for sure further analyses have to be performed.

(2) Get Area Usage Data

Next we look at the usage of space within the different districts. For this, we have the following data set:

In [27]:
```
# Open GeoJson File of Cologne
with open('flaeche.json', encoding='utf8') as f:
    data = json.load(f)
```

Again, we have a geojson file which has attributes about different space usage categories of the districts. We want to extract only a few in here. These are:

- FN_BEBAUT_AP : Ratio of built-up area
- FN_PARK_AP : Ratio of park area
- FN_VERKEHR_AP : Ratio of area assigned to traffic
- FN_WASSER_AP : Area ratio of water

In [28]:
```
df_cologne_area = pd.DataFrame(columns=['District','built-up', 'park', 'traffic', 'water'])
for district in data['features']:
    df_cologne_area = df_cologne_area.append(
        {
            'District': district['properties']['NAME'],
            'built-up': district['properties']['FN_BEBAUT_AP'],
            'park': district['properties']['FN_PARK_AP'],
            'traffic': district['properties']['FN_VERKEHR_AP'],
            'water': district['properties']['FN_WASSER_AP']
        }, ignore_index=True
    )
df_cologne_area.head()
```

Out[28]:

	District	built-up	park	traffic	water
0	Mülheim	52.1	10	24.6	10
1	Braunsfeld	68.5	1.2	26.8	0
2	Ossendorf	41.4	24.5	15.1	0
3	Porz	61.2	6.1	19.9	9
4	Altstadt-Süd	54.7	4.2	30.9	10.2

Cluster analysis

After we have a data frame containing the ratios of different area usage categories, we can again try to find clusters but this time considering area usages. As the process is similar as above for the venues, I won't further explain the process.

In [29]:
```
# set number of clusters
kclusters = 6

# Normalize the data by dividing by 100 (as they are percent values)
c_area_grouped_clustering = df_cologne_area.drop('District', 1) / 100

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

In [30]:
```
# add clustering labels
df_cologne_area_with_lables = df_cologne_area.copy()
df_cologne_area_with_lables.insert(len(df_cologne_area_with_lables.
columns), 'Area Cluster Labels', kmeans.labels_)

df_cologne_area_with_lables.head()

df_cologne_merged_area = df_cologne_area_with_lables.merge(df_colog
ne[['District', 'Latitude', 'Longitude']], on='District', how='oute
r')
df_cologne_merged_area.head()
```

Out[30]:

	District	built-up	park	traffic	water	Area Cluster Labels	Latitude	Longitude
0	Mülheim	52.1	10	24.6	10	2	50.965368	6.999677
1	Braunsfeld	68.5	1.2	26.8	0	5	50.940041	6.896742
2	Ossendorf	41.4	24.5	15.1	0	4	50.977418	6.896325
3	Porz	61.2	6.1	19.9	9	5	50.884047	7.062007
4	Altstadt-Süd	54.7	4.2	30.9	10.2	2	50.930917	6.951047

Let's also check if we have any NaN in the frame:

In [31]:
```
df_cologne_merged_area[df_cologne_merged_area.isna().any(axis=1)]
```

Out[31]:

	District	built-up	park	traffic	water	Area Cluster Labels	Latitude	Longitude

As there are no NaNs, we can directly plot the data in a map:

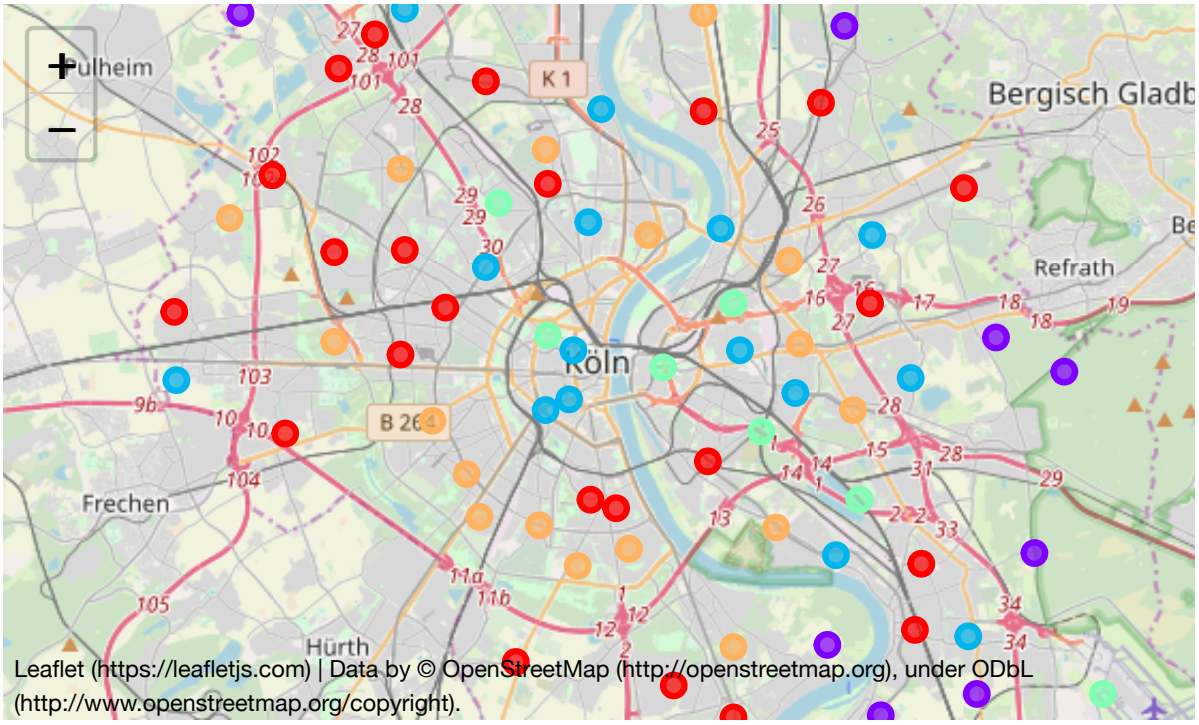
In [66]:
```
# create map
map_clusters_area = folium.Map(location=[latitude, longitude], zoom
_start=11)

# set color scheme for the clusters
x = np.arange(kclusters)
ys = [i + x + (i*x)**2 for i in range(kclusters)]
colors_array = cm.rainbow(np.linspace(0, 1, len(ys)))
rainbow = [colors.rgb2hex(i) for i in colors_array]

# add markers to the map
markers_colors = []
for lat, lon, poi, cluster in zip(df_cologne_merged_area['Latitude'
], df_cologne_merged_area['Longitude'], df_cologne_merged_area['Dis
trict'], df_cologne_merged_area['Area Cluster Labels']):
    label = folium.Popup(str(poi) + ' Cluster ' + str(cluster), par
se_html=True)
    folium.CircleMarker(
        [lat, lon],
        radius=5,
        popup=label,
        color=rainbow[cluster-1],
        fill=True,
        fill_color=rainbow[cluster-1],
        fill_opacity=0.7).add_to(map_clusters_area)

map_clusters_area
```

Out[66]:



As expected we see the distribution of clusters matching the open street map's color code. Areas which are close to rivers belong to one cluster category and then the clusters are ring-shaped with their common center being the city center. This makes again sense, since built-up areas and traffic decreases radially. Let's try to use a Choropleth map to make this more apparent:

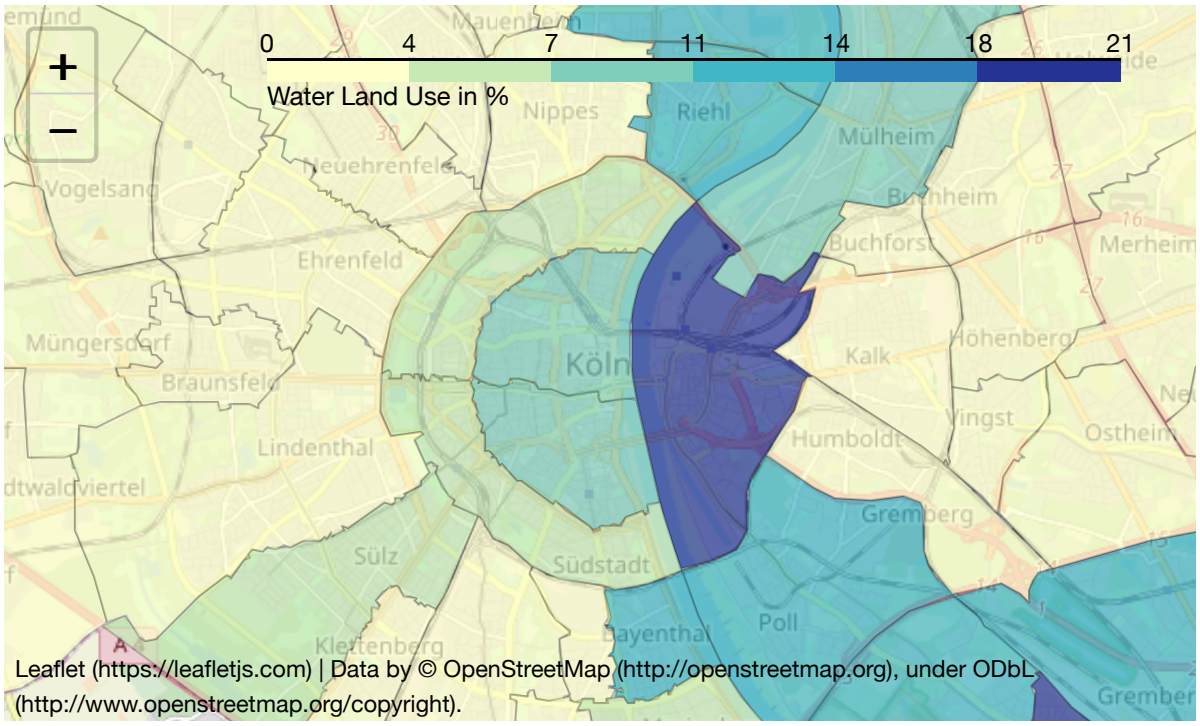
In [33]:
```
# Open GeoJson File of Cologne without encoding
with open('flaeche.json') as f:
    data = json.load(f)
df_cologne_area_encNone = pd.DataFrame(columns=['District','built-up', 'park', 'traffic', 'water'])
for district in data['features']:
    df_cologne_area_encNone = df_cologne_area_encNone.append(
        {
            'District': district['properties']['NAME'],
            'built-up': district['properties']['FN_BEBAUT_AP'],
            'park': district['properties']['FN_PARK_AP'],
            'traffic': district['properties']['FN_VERKEHR_AP'],
            'water': district['properties']['FN_WASSER_AP']
        }, ignore_index=True
    )
```

In [67]:
```
from folium import plugins
cc_map = folium.Map(location=[latitude, longitude],zoom_start=12)

folium.Choropleth(
    geo_data='flaeche.json',
    name='Water',
    data=df_cologne_area_encNone,
    columns=['District', 'water'],
    key_on='feature.properties.NAME',
    fill_color='YlGnBu',
    fill_opacity=0.7,
    line_opacity=0.2,
    legend_name='Water Land Use in %'
).add_to(cc_map)

cc_map
```

Out[67]:



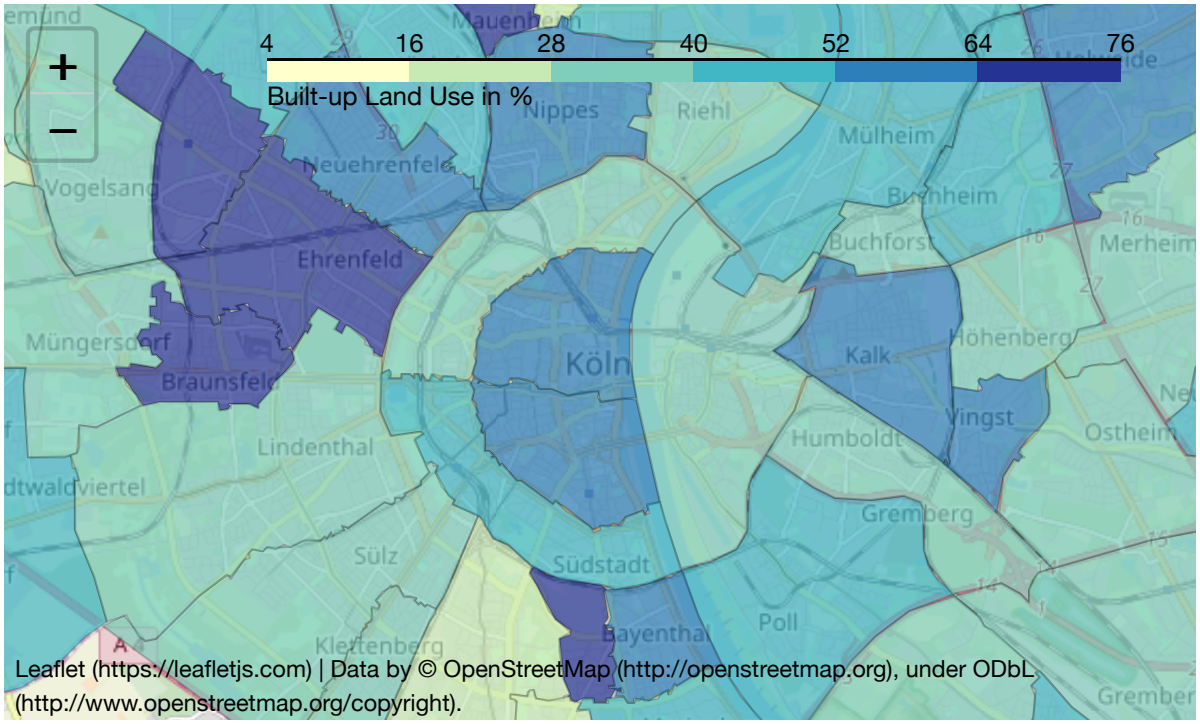
In this Choropleth map we see the water density in the certain districts. This semi-explains the distribution of clusters. Let's also look at the built-up density:

In [68]:
```
from folium import plugins
cc_map = folium.Map(location=[latitude, longitude],zoom_start=12)

folium.Choropleth(
    geo_data='flaeche.json',
    name='Built-up',
    data=df_cologne_area_encNone,
    columns=['District', 'built-up'],
    key_on='feature.properties.NAME',
    fill_color='YlGnBu',
    fill_opacity=0.7,
    line_opacity=0.2,
    legend_name='Built-up Land Use in %'
).add_to(cc_map)

cc_map
```

Out[68]:



Here, we see how built-up areas are more dense towards the city center, but we also find some dense areas towards the south-east near the airport. This could - for instance - explain, why districts in that area fall into the same cluster label class as districts in the city center.

(3) Households

In the following, we use the very same analysis as for the area usage, but now for the distribution of households. The data has the same format, i.e. the same processes are used to extract the following columns:

- District: Name of district
- density: Density of households per ha
- ratio-xP: Ratio of people living in households of x persons
- ratio-5P+: Ratio of people living in households of 5 persons or above

In [36]:
```
# Open GeoJson File of Cologne
with open('haushalte.json', encoding='utf8') as f:
    data = json.load(f)
```

In [37]:
```
df_cologne_household = pd.DataFrame(columns=['District','density',
'ratio-1P', 'ratio-2P', 'ratio-3P', 'ratio-4P', 'ratio-5P+'])
for district in data['features']:
    df_cologne_household = df_cologne_household.append(
        {
            'District': district['properties']['NAME'],
            'density': district['properties']['HHD_AP'],
            'ratio-1P': district['properties']['HH_HG_1PERS_AP'],
            'ratio-2P': district['properties']['HH_HG_2PERS_AP'],
            'ratio-3P': district['properties']['HH_HG_3PERS_AP'],
            'ratio-4P': district['properties']['HH_HG_4PERS_AP'],
            'ratio-5P+': district['properties']['HH_HG_AB5PERS_AP']
,
        }, ignore_index=True
    )
df_cologne_household.head()
```

Out[37]:

	District	density	ratio-1P	ratio-2P	ratio-3P	ratio-4P	ratio-5P+
0	Stammheim	10.4	40.7	32	14.2	9	4.2
1	Mülheim	31.8	54.3	24.4	10.3	7.2	3.8
2	Braunsfeld	40.6	56.0	26.5	9.5	6.1	1.9
3	Ossendorf	6.6	38.1	27.9	15.6	12.7	5.8
4	Porz	22.7	44.0	29.3	12.6	9.5	4.6

Again, using clustering, we retrieve a map showing the cluster distribution by household:

In [38]:
```
# set number of clusters
kclusters = 5

c_hh_grouped_clustering = df_cologne_household.drop('District', 1)
/ 100

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

In [39]:
```
# add clustering labels
df_cologne_hh_with_lables = df_cologne_household.copy()
df_cologne_hh_with_lables.insert(len(df_cologne_hh_with_lables.colu
mns), 'HH Cluster Labels', kmeans.labels_)

df_cologne_hh_with_lables.head()

df_cologne_merged_hh = df_cologne_hh_with_lables.merge(df_cologne[[
'District', 'Latitude', 'Longitude']], on='District', how='outer')
df_cologne_merged_hh.head()
```

Out[39]:

	District	density	ratio-1P	ratio-2P	ratio-3P	ratio-4P	ratio-5P+	HH Cluster Labels	Latitude	Longitude
0	Stammheim	10.4	40.7	32	14.2	9	4.2	1	50.989277	6.993882
1	Mülheim	31.8	54.3	24.4	10.3	7.2	3.8	3	50.965368	6.999677
2	Braunsfeld	40.6	56.0	26.5	9.5	6.1	1.9	0	50.940041	6.896742
3	Ossendorf	6.6	38.1	27.9	15.6	12.7	5.8	1	50.977418	6.896325
4	Porz	22.7	44.0	29.3	12.6	9.5	4.6	3	50.884047	7.062007

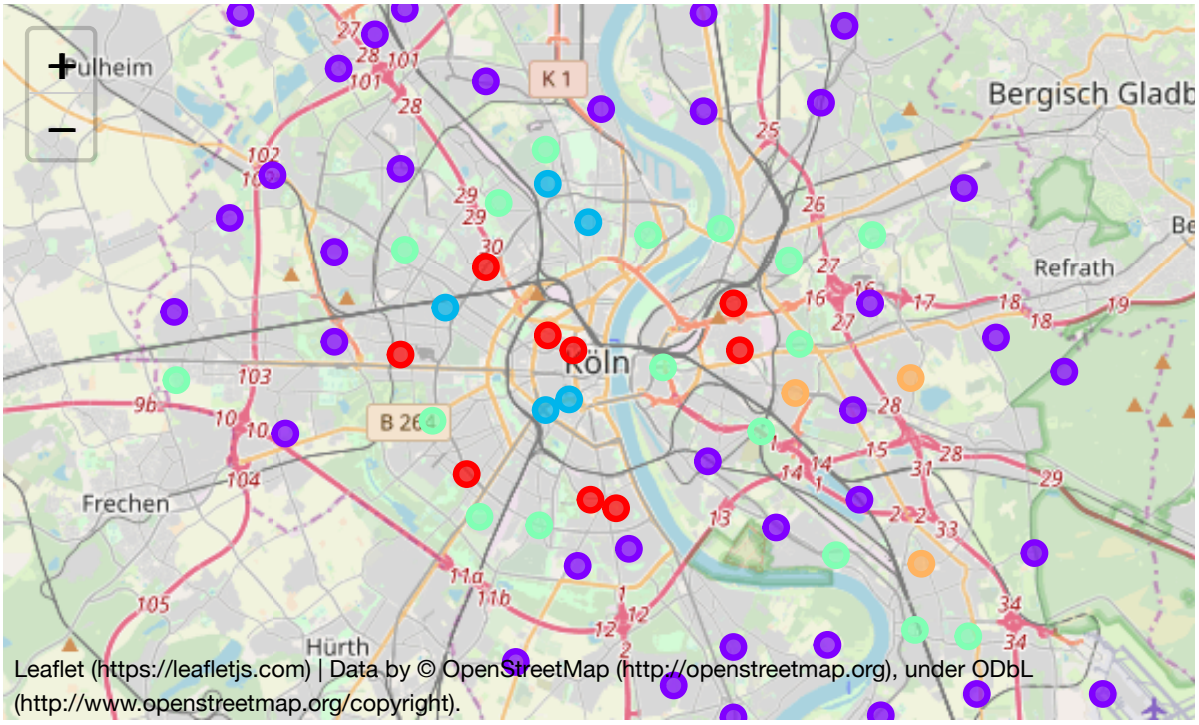
In [40]:
```
# create map
map_clusters_hh = folium.Map(location=[latitude, longitude], zoom_s
tart=11)

# set color scheme for the clusters
x = np.arange(kclusters)
ys = [i + x + (i*x)**2 for i in range(kclusters)]
colors_array = cm.rainbow(np.linspace(0, 1, len(ys)))
rainbow = [colors.rgb2hex(i) for i in colors_array]

# add markers to the map
markers_colors = []
for lat, lon, poi, cluster in zip(df_cologne_merged_hh['Latitude'],
df_cologne_merged_hh['Longitude'], df_cologne_merged_hh['District']
, df_cologne_merged_hh['HH Cluster Labels']):
    label = folium.Popup(str(poi) + ' Cluster ' + str(cluster), par
se_html=True)
    folium.CircleMarker(
        [lat, lon],
        radius=5,
        popup=label,
        color=rainbow[cluster-1],
        fill=True,
        fill_color=rainbow[cluster-1],
        fill_opacity=0.7).add_to(map_clusters_hh)

map_clusters_hh
```

Out[40]:



Here, it is very obvious, that there is one cluster which contains the outer areas of Cologne. Let's look at it's shape:

In [41]:
```
df_cologne_merged_hh[df_cologne_merged_hh['HH Cluster Labels'] == 1
].iloc[:,2:7].mean()
```

Out[41]:
```
ratio-1P     38.726531
ratio-2P     31.510204
ratio-3P     14.404082
ratio-4P     10.777551
ratio-5P+     4.581633
dtype: float64
```

We see, that the household distribution is rather equal. This speaks for larger families living in these areas. Compared to households near the city center, we see the difference:

In [42]:
```
df_cologne_merged_hh[df_cologne_merged_hh['HH Cluster Labels'] == 2
].iloc[:,2:7].mean()
```

Out[42]:
```
ratio-1P     61.86
ratio-2P     22.50
ratio-3P      8.32
ratio-4P      5.44
ratio-5P+     1.88
dtype: float64
```

Lastly, let's visualize the density of households on a Choropleth map:

In [43]:
```
# Open GeoJson File of Cologne without encoding
with open('haushalte.json') as f:
    data = json.load(f)
df_cologne_hh_encNone = pd.DataFrame(columns=['District','built-up'
, 'park', 'traffic', 'water'])
for district in data['features']:
    df_cologne_hh_encNone = df_cologne_hh_encNone.append(
        {
            'District': district['properties']['NAME'],
            'density': district['properties']['HHD_AP'],
            'ratio-1P': district['properties']['HH_HG_1PERS_AP'],
            'ratio-2P': district['properties']['HH_HG_2PERS_AP'],
            'ratio-3P': district['properties']['HH_HG_3PERS_AP'],
            'ratio-4P': district['properties']['HH_HG_4PERS_AP'],
            'ratio-5P+': district['properties']['HH_HG_AB5PERS_AP']
,
        }, ignore_index=True
    )
```

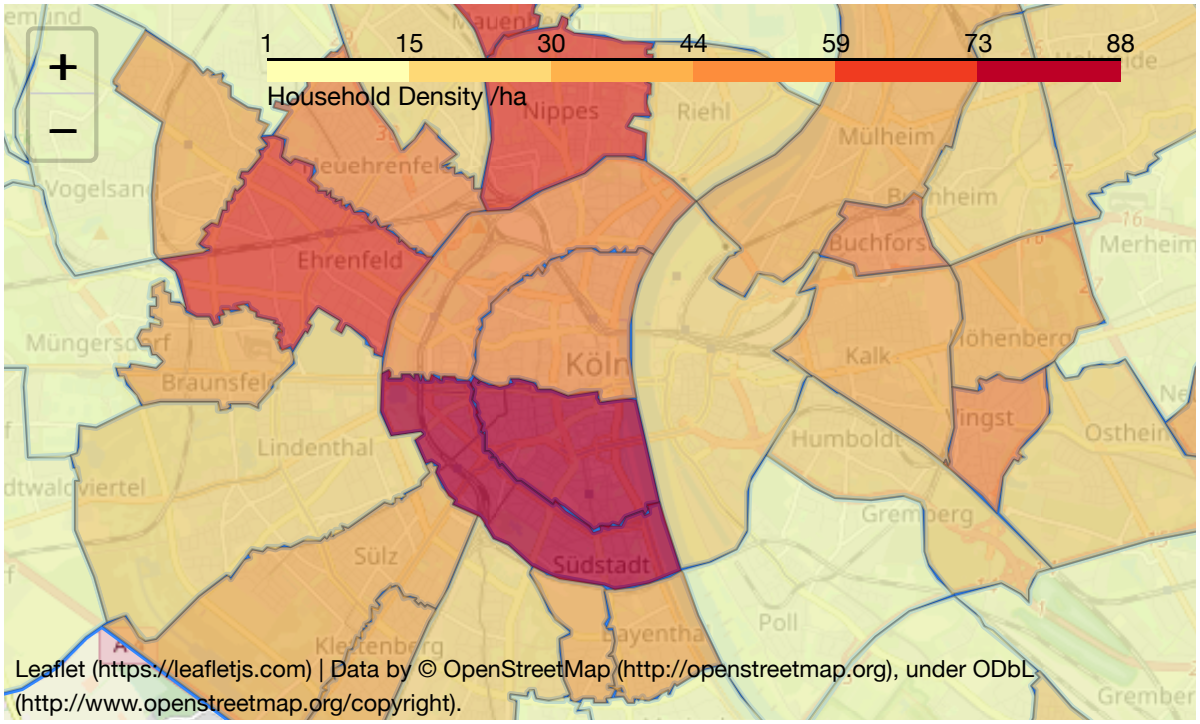
In [44]:
```
from folium import plugins
cc_map = folium.Map(location=[latitude, longitude],zoom_start=12)

folium.GeoJson('haushalte.json').add_to(cc_map)

folium.Choropleth(
    geo_data='haushalte.json',
    name='Household Density',
    data=df_cologne_hh_encNone,
    columns=['District', 'density'],
    key_on='feature.properties.NAME',
    fill_color='YlOrRd',
    fill_opacity=0.7,
    line_opacity=0.2,
    legend_name='Household Density /ha'
).add_to(cc_map)

cc_map
```

Out[44]:



This result matches the expectation as districts which show higher amounts of single person households will also show a higher density of households.

(4) Age Distribution

Lastly, we look at the age distribution in the districts. We have the data in `.csv` format and can easily scrape them.

In [45]:
```
df_cologne_age = pd.read_csv('altersgruppen.csv', sep=';')
df_cologne_age.head()
```

Out[45]:

	Nr.	Stadtteil	Einwohner insgesamt	0-2	3-5	6-14	15-17	18-20	21-34	35-59	60-64	65-74	75
0	101	Altstadt-Süd	27571	517	423	961	364	662	9222	9693	1236	2240	106
1	102	Neustadt-Süd	38262	1018	767	1654	507	1021	13821	13914	1487	2247	89
2	103	Altstadt-Nord	17896	338	226	526	177	449	6050	6545	731	1310	59
3	104	Neustadt-Nord	28206	768	573	1183	364	538	8210	11427	1306	2196	85
4	105	Deutz	15153	377	305	703	248	438	4408	5530	628	1222	61

Do some cosmetics:

In [46]:
```
df_cologne_age.drop(['Nr.'], axis=1, inplace=True)
df_cologne_age.rename(columns={'Einwohner insgesamt': 'Total', 'Sta
dtteil': 'District', '80 und älter': '80+'}, inplace=True)
```

In [47]:
```
# Normalize values
df_cologne_age_relative = df_cologne_age[df_cologne_age.columns[2:]
].apply(lambda c: c/df_cologne_age.Total)
df_cologne_age_relative['District'] = df_cologne_age['District']

# Move District-column to beginning of table
fixed_columns = list(df_cologne_age_relative.columns[-1:]) + list(d
f_cologne_age_relative.columns[:-1])
df_cologne_age_relative = df_cologne_age_relative[fixed_columns]

df_cologne_age_relative.head()
```

Out[47]:

	District	0-2	3-5	6-14	15-17	18-20	21-34	35-59	60-6
0	Altstadt-Süd	0.018752	0.015342	0.034855	0.013202	0.024011	0.334482	0.351565	0.0448
1	Neustadt-Süd	0.026606	0.020046	0.043228	0.013251	0.026684	0.361220	0.363651	0.0388
2	Altstadt-Nord	0.018887	0.012629	0.029392	0.009890	0.025089	0.338064	0.365724	0.0408
3	Neustadt-Nord	0.027228	0.020315	0.041941	0.012905	0.019074	0.291073	0.405127	0.0463
4	Deutz	0.024880	0.020128	0.046393	0.016366	0.028905	0.290899	0.364944	0.0414

And perform again the cluster analysis:

In [48]:
```
# set number of clusters
kclusters = 5

c_age_grouped_clustering = df_cologne_age_relative.drop('District',
1)

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

In [49]:
```
# add clustering labels
df_cologne_age_with_lables = df_cologne_age_relative.copy()
df_cologne_age_with_lables.insert(len(df_cologne_age_with_lables.co
lumns), 'Age Cluster Labels', kmeans.labels_)

df_cologne_age_with_lables.head()

df_cologne_merged_age = df_cologne_age_with_lables.merge(df_cologne
[['District', 'Latitude', 'Longitude']], on='District')
```

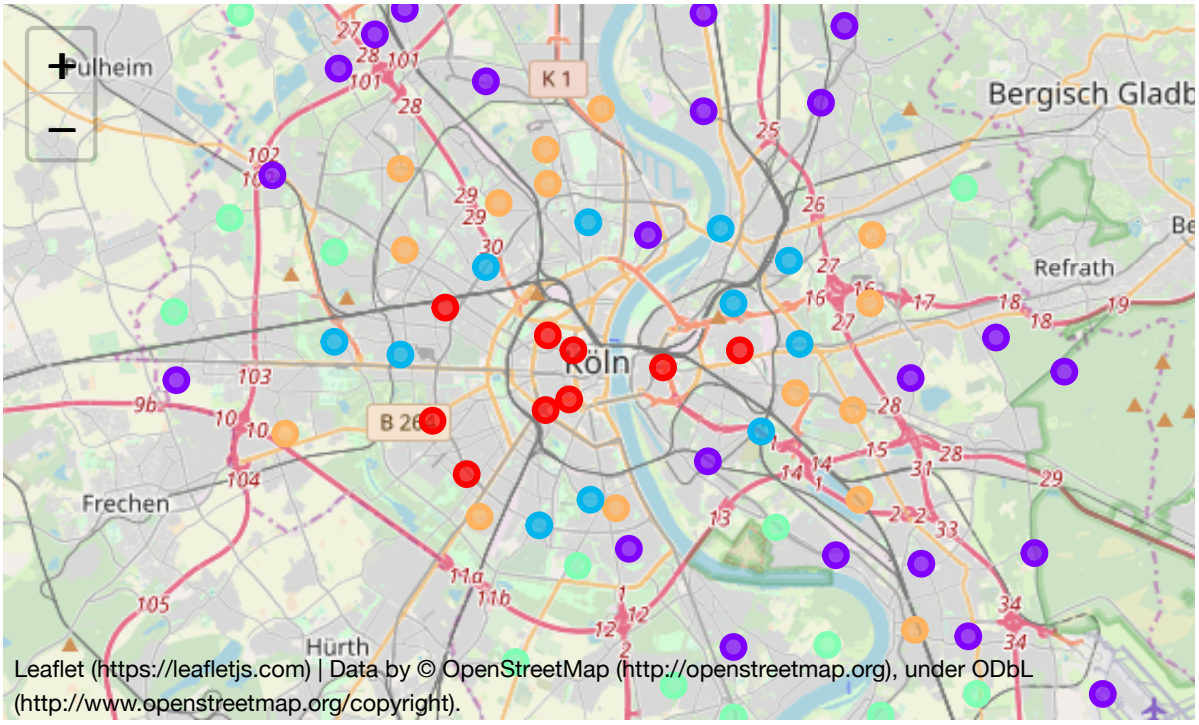
In [50]:
```
# create map
map_clusters_age = folium.Map(location=[latitude, longitude], zoom_
start=11)

# set color scheme for the clusters
x = np.arange(kclusters)
ys = [i + x + (i*x)**2 for i in range(kclusters)]
colors_array = cm.rainbow(np.linspace(0, 1, len(ys)))
rainbow = [colors.rgb2hex(i) for i in colors_array]

# add markers to the map
markers_colors = []
for lat, lon, poi, cluster in zip(df_cologne_merged_age['Latitude']
, df_cologne_merged_age['Longitude'], df_cologne_merged_age['Distri
ct'], df_cologne_merged_age['Age Cluster Labels']):
    label = folium.Popup(str(poi) + ' Cluster ' + str(cluster), par
se_html=True)
    folium.CircleMarker(
        [lat, lon],
        radius=5,
        popup=label,
        color=rainbow[cluster-1],
        fill=True,
        fill_color=rainbow[cluster-1],
        fill_opacity=0.7).add_to(map_clusters_age)

map_clusters_age
```

Out[50]:



We see again radial clustering. But here, we can see a clearer distinction than before. Let's make this distinction visible at one example: Note how all the districts in the center of Cologne (horizontal center) are in the same cluster class (cluster 0). Please look at the age distribution there:

In [51]:
```
df_cologne_merged_age[df_cologne_merged_age['Age Cluster Labels'] =
= 0].iloc[:,1:12].mean()*100
```

Out[51]:
```
0-2        2.636131
 3-5       2.030705
 6-14      4.782664
15-17      1.557625
18-20      2.658425
21-34     31.367353
35-59     36.322382
60-64      4.182191
65-74      7.392915
75-79      3.271608
80+        3.798002
dtype: float64
```

In comparison let's look at the age distributions from two outer cluster families:

In [52]:
```
df_cologne_merged_age[df_cologne_merged_age['Age Cluster Labels'] =
= 1].iloc[:,1:12].mean()*100
```

Out[52]:
```
0-2        2.755135
 3-5       2.790196
 6-14      8.350272
15-17      3.048606
18-20      3.060187
21-34     16.454788
35-59     34.952961
60-64      5.952040
65-74     10.995371
75-79      5.543783
80+        6.096660
dtype: float64
```

In [53]:
```
df_cologne_merged_age[df_cologne_merged_age['Age Cluster Labels'] =
= 4].iloc[:,1:12].mean()*100
```

Out[53]:
```
0-2        3.208497
 3-5       3.121043
 6-14      9.346335
15-17      3.198538
18-20      3.399559
21-34     19.583120
35-59     37.059361
60-64      5.087842
65-74      8.474782
75-79      3.713653
80+        3.807268
dtype: float64
```

We see that in the center live people in working age. This may be because, it's rather expensive to live there but at the same time it's a very busy place. People who get older, or who want to raise children move more to the outer districts. Which is visible from the higher amount of people in age 6-14 (children at school) and 65-74 (retired).

(5) Putting everything together

Finally we want to look at the migration between districts. In the last sections we have prepared the following data tables:

In [54]:
```
df_cologne_merged_age.head(0)
```

Out[54]:

District	0-2	3-5	6-14	15-17	18-20	21-34	35-59	60-64	65-74	75-79	80+	Age Cluster Labels	Latitude	Longitude

In [55]:
```
df_cologne_merged_area.head(0)
```

Out[55]:

District	built-up	park	traffic	water	Area Cluster Labels	Latitude	Longitude

In [56]:
```
df_cologne_merged_hh.head(0)
```

Out[56]:

District	density	ratio-1P	ratio-2P	ratio-3P	ratio-4P	ratio-5P+	HH Cluster Labels	Latitude	Longitude

In [57]:
```
cologne_merged.head(0)
```

Out[57]:

District	Latitude	Longitude	Polygon	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue

Let's now merge the cluster analyses' results into one frame:

In [58]:
```
df_cologne_result = cologne_merged[['District', 'Latitude', 'Longit
ude', 'Cluster Labels']].merge(df_cologne_merged_age[['District','A
ge Cluster Labels']], on='District')
df_cologne_result = df_cologne_result.merge(df_cologne_merged_area[
['District', 'Area Cluster Labels']], on='District')
df_cologne_result = df_cologne_result.merge(df_cologne_merged_hh[['
District', 'HH Cluster Labels']], on='District')
df_cologne_result.rename(columns={'Cluster Labels': 'Venue Cluster
Labels'}, inplace=True)
df_cologne_result.head()
```

Out[58]:

	District	Latitude	Longitude	Venue Cluster Labels	Age Cluster Labels	Area Cluster Labels	HH Cluster Labels
0	Godorf	50.852610	6.982218	3	4	5	1
1	Lövenich	50.948533	6.823829	3	3	0	1
2	Weiden	50.934514	6.824246	6	1	2	3
3	Junkersdorf	50.923891	6.859688	3	4	0	1
4	Widdersdorf	50.967660	6.841605	1	3	4	1

Now, we add a data file which contains data about the net migration within a district, i.e. if it's > 0, we have a growing population. Let's encode that fact in a new variable: `Churn` :

In [59]:
```
# Open GeoJson File of Cologne without encoding
with open('einwohner.json') as f:
    data = json.load(f)
y = pd.DataFrame(columns=['District','Churn'])
for district in data['features']:
    y = y.append({
        'District': district['properties']['NAME'],
        'Churn': district['properties']['SALDO_BINNEN_AA'] > 0
    }, ignore_index=True)
df_cologne_result = df_cologne_result.merge(y, on='District')
df_cologne_result['Churn'] = df_cologne_result['Churn'].map({True:
1, False: 0})
df_cologne_result.head()
```

Out[59]:

	District	Latitude	Longitude	Venue Cluster Labels	Age Cluster Labels	Area Cluster Labels	HH Cluster Labels	Churn
0	Godorf	50.852610	6.982218	3	4	5	1	1
1	Weiden	50.934514	6.824246	6	1	2	3	1
2	Junkersdorf	50.923891	6.859688	3	4	0	1	1
3	Widdersdorf	50.967660	6.841605	1	3	4	1	1
4	Vogelsang	50.960676	6.875571	3	3	0	1	0

This is our data set. We split it into a training and test set and model a decision tree out of it.

In [60]:
```
X = df_cologne_result.iloc[:,3:-1]
Y = df_cologne_result['Churn']

X_train, X_test, y_train, y_test = train_test_split( X, Y, test_siz
e=0.4, random_state=4)

clf = tree.DecisionTreeClassifier(criterion='entropy', max_depth=3
)
clf = clf.fit(X_train, y_train)
```

Unfortunately, we have a rather low f1 and jaccard score. Probably, we'd need more data to get better results. Therefore, it would be good to make the analysis more granular, i.e. have more datapoints per district.

In [61]:
```
print('Decision Tree F1-Score: {:.2f}'.format(sklearn.metrics.f1_sc
ore(y_test, clf.predict(X_test), average='weighted')))
print('Decision Tree Jaccard-Score: {:.2f}'.format(sklearn.metrics.
jaccard_similarity_score(y_test, clf.predict(X_test))))
```

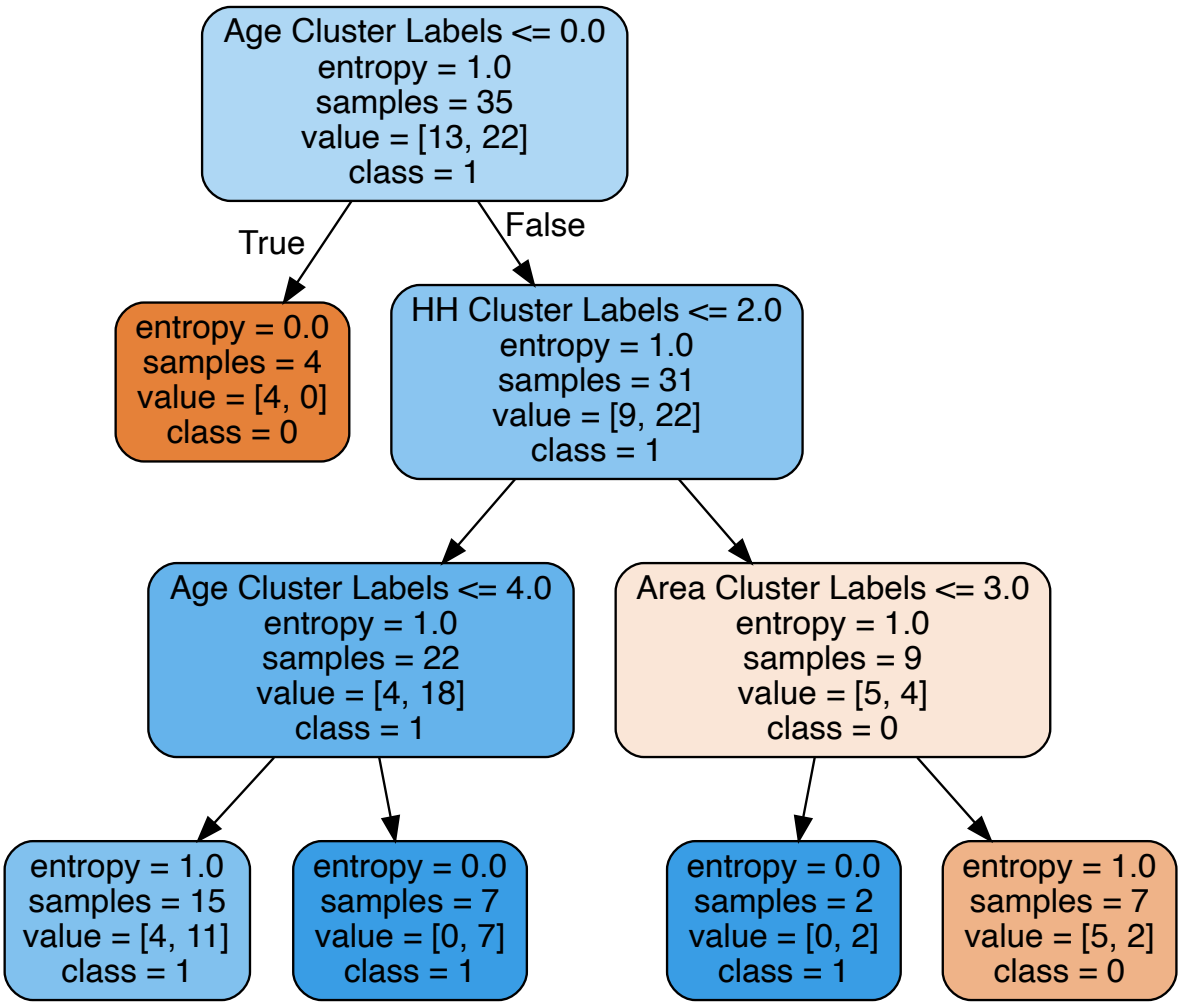
```
Decision Tree F1-Score: 0.49
Decision Tree Jaccard-Score: 0.54

C:\Users\sebastian\Anaconda3\envs\coursera\lib\site-packages\sklea
rn\metrics\classification.py:635: DeprecationWarning: jaccard_simi
larity_score has been deprecated and replaced with jaccard_score.
It will be removed in version 0.23. This implementation has surpri
sing behavior for binary and multiclass classification tasks.
  'and multiclass classification tasks.', DeprecationWarning)
```

Nevertheless, let's look at the resulting tree:

In [62]:
```
import graphviz
dot_data = tree.export_graphviz(
    clf,
    feature_names=X.columns.values,
    class_names=list(map(str,range(max(Y)+1))),
    filled=True, rounded=True, precision=0,
    out_file=None)
graph = graphviz.Source(dot_data, format='svg')
graph.render(filename='figs/decision')
graph
```

Out[62]:



By looking at the top features of the decision tree, we see which categories seem to have the biggest impact on whether or not a district is growing or shrinking.

In [63]:
```
df_topfeatures = pd.DataFrame({'Feature': X.columns.values, 'Importance':clf.feature_importances_}).sort_values(by='Importance', ascending=False).head().reset_index()
df_topfeatures
```

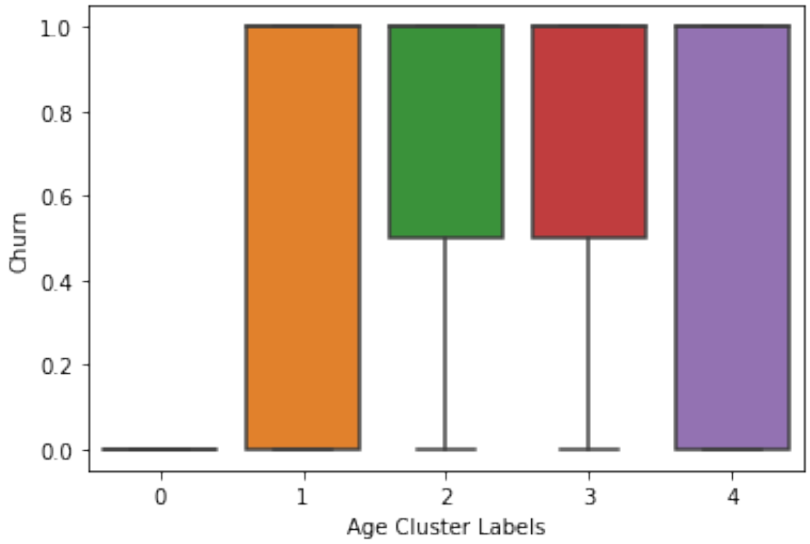
Out[63]:

	index	Feature	Importance
0	1	Age Cluster Labels	0.602413
1	3	HH Cluster Labels	0.202085
2	2	Area Cluster Labels	0.195502
3	0	Venue Cluster Labels	0.000000

This is, that primarily, the age distribution govers, if a district is growing or shrinking. Let's therefore look at the age cluster labels that mean a high/low churn:

In [64]:
```
sns.boxplot(data=df_cologne_result, x='Age Cluster Labels', y='Churn')
```

Out[64]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1faad13d550>
```



We see that for instance districts with age cluster 0 labels have a shrinking population. We remember, these where the ones in the city center, where most probably, costs of living are very high and where we had a high ratio of younger people living in 1Person households. As soon as the found a family, they will move to outer district, i.e. this makes sense. If we look for instance at age cluster 3:

In [65]:
```
df_cologne_merged_age[df_cologne_merged_age['Age Cluster Labels'] ==
= 3].iloc[:,1:12].mean()*100
```

Out[65]:
```
0-2        2.624339
 3-5       2.921833
 6-14      9.284933
15-17      3.396935
18-20      2.971762
21-34     13.878731
35-59     39.165455
60-64      6.060903
65-74     10.416385
75-79      4.601962
80+        4.676762
dtype: float64
```

We see, that in those districts, older people are living that have already a family or they are retired. In both cases they probably have a lower interest in leaving their place, i.e. they stay while people from other districts enter.

In [ ]: