Capstone Project – The battle of the neighbourhoods

Where to live in Rotterdam, the greatest city of Holland

René Kluijtmans

Capstone Project – Applied Data Science

IBM Data Science Professional Certificate

Introduction

Relatively few know about the beauty and greatness of Rotterdam. Amsterdam is the capital of Holland, famous under tourists. Foreigners that go to Holland most likely want to visit Amsterdam. As a second best, The Hague is known for being the political centre of Holland. However, relatively few know Rotterdam. A city with hard working people that made sure Rotterdam was rebuild after being bombed in World War 2 and WOW, they did an amazing job. Rotterdam is a very modern city, with its hard working citizens still living in it. Rotterdam is multicultural, has all the facilities you need (and more) and don't forget about the beautiful sightseeing spots all over the city.

Most likely, you now found out that I actually adore Rotterdam. Currently I live in a village nearby Rotterdam, but I'm looking to move and live in the city of Rotterdam.

Business Problem

I always lived close to Rotterdam and know the city by heart. However, when I think about moving there, I actually don't know what the best neighbourhood would be. As I really want to have the full experience, I need all the top venues nearby. However, what would then be the best place to buy or rent a house or apartment?

As I can't choose the best neighbourhood by heart, I would love to use my new data science skills to determine the best neighbourhood for me!

Data Description

For this project, geographical data of the neighbourhoods in Rotterdam is required. As there are no clear postal codes available for the neighbourhoods, I will go with the name of the neighbourhoods as starting point for my research.

To extract the neighbourhood names of Rotterdam, I will scrape the following webpage: https://en.wikipedia.org/wiki/Districts_and_neighbourhoods_of_Rotterdam

This page has information on the neighbourhoods of Rotterdam. To make sure we only use relevant data, we will check if there are neighbourhoods to be excluded from this project.

First of all, I notice "Kop van Zuid; number and map are dated, since 3 March 2010 part of Feijenoord" - this means we have to drop that one.

Secondly, on the webpage there is a map next to each neighbourhood to see where in Rotterdam this neighbourhood is located. As there are also villages and boroughs that are outside of the city of Rotterdam, but fall under the township of Rotterdam, we have to drop those as well. I don't want to live in a village part of the township of Rotterdam, but really in Rotterdam itself. This means that the following neighbourhoods will be dropped from the data as well:

- Pernis
- Hoogvliet-Noord
- Hoogvliet-Zuid
- Rozenburg
- Maasvlakte
- Europoort
- Botlek
- Vondelingenplaat
- Rijnpoort
- Dorp
- Strand en Duin
- Noordzeeweg

The same goes for the 2 'Bedrijventerrein' areas - which means factory area, because that is of course not the area I want to buy a house!

As I only derive the neighbourhood names from this webpage, I have to use geopy and Nominatim to get the latitude and longitude for these neighbourhoods as well. When having a neighbourhood table complete with name, latitude and longitude, I can explore venues for each neighbourhood with Foursquare API.

Foursquare is a location data provider with information about all manner of venues and events within an area of interest. Such information includes venue names, locations, ratings etc. With the list of neighbourhoods Rotterdam, I will call the Foursquare API to gather information about venues in each neighbourhood. The radius that I will use is set to be 500 meters.

After connecting to Foursquare API, I should have a table consisting of the following information:

- Neighbourhood
- Neighbourhood Latitude
- Neighbourhood Longitude
- Venue Name
- Venue Latitude
- Venue Longitude
- Venue Category

Based on all the information for the neighbourhoods in Rotterdam, I can research what would be the best neighbourhood to buy or rent a house. I will cluster the neighbourhoods together based on similar venue categories with K-Means clustering method. With that information, it should be time to start looking for houses!

Methodology

As I'll perform my data research in Python, I started with installing and importing the relevant libraries...

```
!pip install folium
import pandas as pd
import requests
import numpy as np
import matplotlib.pyplot as plt
import folium
from sklearn.cluster import KMeans

print('All has been installed and imported')
```

I scraped the relevant data from the Wikipedia page with neighbourhoods in Rotterdam:

```
url = "https://en.wikipedia.org/wiki/Districts_and_neighbourhoods_of_Rotterdam"
rdam_url = requests.get(url)
rdam = pd.read_html(rdam_url.text)
rdam = rdam[1]
rdam
```

I cleaned the table, so that I only had a table with the names of the neighbourhoods. As already discussed in the data description, there were some neighbourhoods that needed to be excluded from this project. I used the following codes to do so:

```
rdam_nh = rdam[['Neighbourhood name']]
rdam_nh = rdam_nh.rename(columns={'Neighbourhood name' : 'Neighbourhood'})
pd.set_option('display.max_rows', None)
rdam_nh
```

```
rdam_nh = rdam_nh.drop([rdam_nh.index[4],rdam_nh.index[54],rdam_nh.index[72],rdam_nh.index[73],rdam_nh.index[74],rdam_
nh.index[75],rdam_nh.index[76]])
rdam_nh.set_index('Neighbourhood')
rdam_nh.reset_index()
```

```
rdam_nh = rdam_nh.drop([rdam_nh.index[75],rdam_nh.index[76],rdam_nh.index[77],rdam_nh.index[78],rdam_nh.index[79],rdam
_nh.index[81],rdam_nh.index[82],rdam_nh.index[83]])
```

Now my table with neighbourhoods is complete. I now had to add the latitude and longitude of each neighbourhood. To do so, I installed geopy and Nominatim and performed the following code:

```
from geopy.geocoders import Nominatim
neighbourhoods = rdam_nh['Neighbourhood']
geolocator = Nominatim(user_agent="Your_Name")
latitude=[]
longitude=[]
address=[]

for neighbourhood in neighbourhoods:
    location = geolocator.geocode(neighbourhood)
    latitude.append(location.latitude)
    longitude.append(location.longitude)
    address.append(location.address)
```

```
rdam_nh['Latitude'] = latitude
rdam_nh['Longitude'] = longitude
rdam_nh['Address'] = address

rdam_nh
```

It showed a table like this:

	Neighbourhood	Latitude	Longitude	Address
0	Stadsdriehoek	51.921768	4.486689	Stadsdriehoek, Centrum, Rotterdam, Zuid-Hollan...
1	Oude Westen	51.918045	4.466252	Oude Westen, Centrum, Rotterdam, Zuid-Holland,...
2	Cool	32.800129	-98.001153	Cool, Parker County, Texas, United States
3	C.S. kwartier	51.923476	4.471072	Rotterdam Centraal, CS-Kwartier, Centrum, Rott...
5	Nieuwe Werk	51.909355	4.477428	Nieuwe Werk, Centrum, Rotterdam, Zuid-Holland,...
6	Dijkzigt	51.912436	4.471307	Dijkzigt, Centrum, Rotterdam, Zuid-Holland, Ne...
7	Delfshaven	51.909995	4.445700	Delfshaven, Schiedamseweg, Bospolder, Delfshav...
8	Bospolder	51.908980	4.442782	Bospolder, Delfshaven, Rotterdam, Zuid-Holland...
9	Tussendijken	51.913092	4.441788	Tussendijken, Delfshaven, Rotterdam, Zuid-Holl...
10	Spangen	49.084583	6.358119	Pange, Metz, Moselle, Grand Est, France métrop...

Unfortunately, the geopy loop did not find all the right neighbourhoods. Sometimes it took a neighbourhood from another city and even another country! To clear these failures, I had to look for the right data by adding Rotterdam to the formula and corrected the false data:

```
failures = ['Cool','Spangen','Schieveen','Rubroek','Noordereiland','Zuiderpark']
failure_lat = []
failure_long = []
failure_address = []

for failure in failures:
    failure_data = geolocator.geocode('{}, Rotterdam'.format(failure))
    failure_lat.append(failure_data.latitude)
    failure_long.append(failure_data.longitude)
    failure_address.append(failure_data.address)

df_failure = pd.DataFrame()
df_failure['Latitude'] = failure_lat
df_failure['Longitude'] = failure_long
df_failure['Address'] = failure_address

df_failure
```

Which led to the following table with correct values:

	Latitude	Longitude	Address
0	51.917184	4.477964	Cool, Centrum, Rotterdam, Zuid-Holland, Nederland
1	51.917315	4.435696	Spangen, Delfshaven, Rotterdam, Zuid-Holland, ...
2	51.963773	4.427631	Schieveense polder, Rotterdam, Zuid-Holland, N...
3	51.928047	4.492325	Rubroek, Kralingen-Crooswijk, Rotterdam, Zuid-...
4	51.913256	4.494534	Noordereiland, Feijenoord, Rotterdam, Zuid-Hol...
5	51.881252	4.478361	Zuiderpark, Rotterdam, Zuid-Holland, Nederland

To include the right values in my table I used iloc:

```
rdam_nh.Latitude.iloc[2] = '51.917184'
rdam_nh.Longitude.iloc[2] = '4.477964'
rdam_nh.Latitude.iloc[9] = '51.917315'
rdam_nh.Longitude.iloc[9] = '4.435696'
rdam_nh.Latitude.iloc[17] = '51.963773'
rdam_nh.Longitude.iloc[17] = '4.427631'
rdam_nh.Latitude.iloc[33] = '51.928047'
rdam_nh.Longitude.iloc[33] = '4.492325'
rdam_nh.Latitude.iloc[48] = '51.913256'
rdam_nh.Longitude.iloc[48] = '4.494534'
rdam_nh.Latitude.iloc[68] = '51.881252'
rdam_nh.Longitude.iloc[68] = '4.478361'

rdam_nh
```

Finally, all set to go! Now let's get the coordinates from Rotterdam and create a map with folium (I dropped the address in the meantime):

```
rotterdam = geolocator.geocode('Rotterdam, Nederland')
print(rotterdam.latitude)
print(rotterdam.longitude)
```

```
51.9228934
4.4631786
```

Let's go for it!

```
# Creating the map of London
map_Rdam = folium.Map(location=[rotterdam.latitude, rotterdam.longitude], zoom_start=11)
map_Rdam

# adding markers to map
for latitude, longitude, neighbourhood in zip(rdam_nh['Latitude'], rdam_nh['Longitude'], rdam_nh['Neighbourhood']):
    label = '{}'.format(neighbourhood)
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [latitude, longitude],
        radius=5,
        popup=label,
        color='red',
        fill=True
        ).add_to(map_Rdam)

map_Rdam
```

The map will be shown in the result section. For the analysis on venues I used Foursquare API. First I created values for the relevant information, then I defined a function to get all the venues per neighbourhood, with a radius of 500.

```
CLIENT_ID = 'TRI02FZMYRQNNRMRO3IJ11UKUFPEA2LL1FGRCVWFXLOKY4BF'
CLIENT_SECRET = 'UAZGC4MYMQRW1NC43REGPIRPDISDZ20GHOY5GXPOJIZUEB0B'
VERSION = '20210423'
```

```
def getNearbyVenues(names, latitudes, longitudes, radius=500):

    venues_list=[]
    for name, lat, lng in zip(names, latitudes, longitudes):
        print(name)

        url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={},{}&radius={}'.fo
rmat(
            CLIENT_ID,
            CLIENT_SECRET,
            VERSION,
            lat,
            lng,
            radius
            )

        results = requests.get(url).json()["response"]['groups'][0]['items']

        venues_list.append([(
            name,
            lat,
            lng,
            v['venue']['name'],
            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 = ['Neighbourhood',
                  'Neighbourhood Latitude',
                  'Neighbourhood Longitude',
                  'Venue',
                  'Venue Category']

    return(nearby_venues)
```

```
Rdam_venues = getNearbyVenues(rdam_nh['Neighbourhood'], rdam_nh['Latitude'], rdam_nh['Longitude'])
```

I grouped the results on Venue Category and applied One Hot Encoding to prepare the data for a function to get the most common venues.

```
Rdam_venues.groupby('Venue Category').max()
```

```
Rdam_venues_OHE = pd.get_dummies(Rdam_venues[['Venue Category']], prefix="", prefix_sep="")
Rdam_venues_OHE
```

Now that One Hot Encoding has been applied, I could apply the function for the most common venues. I set the max num of top venues to 10, as there were many venue categories. Then I created a table with the top 10 venues per neighbourhood (table will be shown in results).

```
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 = 10

indicators = ['st', 'nd', 'rd']

# create columns according to number of top venues
columns = ['Neighbourhood']
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))
```

```
neighbourhoods_venues_sorted_rdam = pd.DataFrame(columns=columns)
neighbourhoods_venues_sorted_rdam['Neighbourhood'] = Rdam_grouped['Neighbourhood']

for ind in np.arange(Rdam_grouped.shape[0]):
    neighbourhoods_venues_sorted_rdam.iloc[ind, 1:] = return_most_common_venues(Rdam_grouped.iloc[ind, :], num_top_ven
ues)

neighbourhoods_venues_sorted_rdam.head()
```

With the tables with the most common venues per neighbourhood, I was able to apply K Means Clustering. I decided to go with 5 clusters. When each neighbourhood is assigned to a specific cluster, I add this label to the table.

```
neighbourhoods_venues_sorted_rdam.insert(0, 'Cluster Labels', kmeans_rdam.labels_ +1)
```

```
rdam_data = rdam_nh

rdam_data = rdam_data.join(neighbourhoods_venues_sorted_rdam.set_index('Neighbourhood'), on='Neighbourhood')

rdam_data.head()
```

```
k_num_clusters = 5

Rdam_grouped_clustering = Rdam_grouped.drop('Neighbourhood', 1)

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

Before I show the clusters on the map, I make sure to drop the NaN values before plotting. To plot the data I used cm and colors from matplotlib. Then I wrote the code to plot all the clusters (map shown in results):

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

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

```
map_clusters_rdam = folium.Map(location=[rotterdam.latitude, rotterdam.longitude], zoom_start=11)

# set color scheme for the clusters
x = np.arange(k_num_clusters)
ys = [i + x + (i*x)**2 for i in range(k_num_clusters)]
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(rdam_data['Latitude'], rdam_data['Longitude'], rdam_data['Neighbourhood'], rdam_data
['Cluster Labels']):
    label = folium.Popup('Cluster ' + str(int(cluster) +1) + '\n' + str(poi) , parse_html=True)
    folium.CircleMarker(
        [lat, lon],
        radius=5,
        popup=label,
        color=rainbow[int(cluster-1)],
        fill=True,
        fill_color=rainbow[int(cluster-1)]
        ).add_to(map_clusters_rdam)

map_clusters_rdam
```

Finally, I examined all the different clusters to check what specifies them and what cluster would be the best to look for a new house. The following code was applied for each of the 5 clusters:

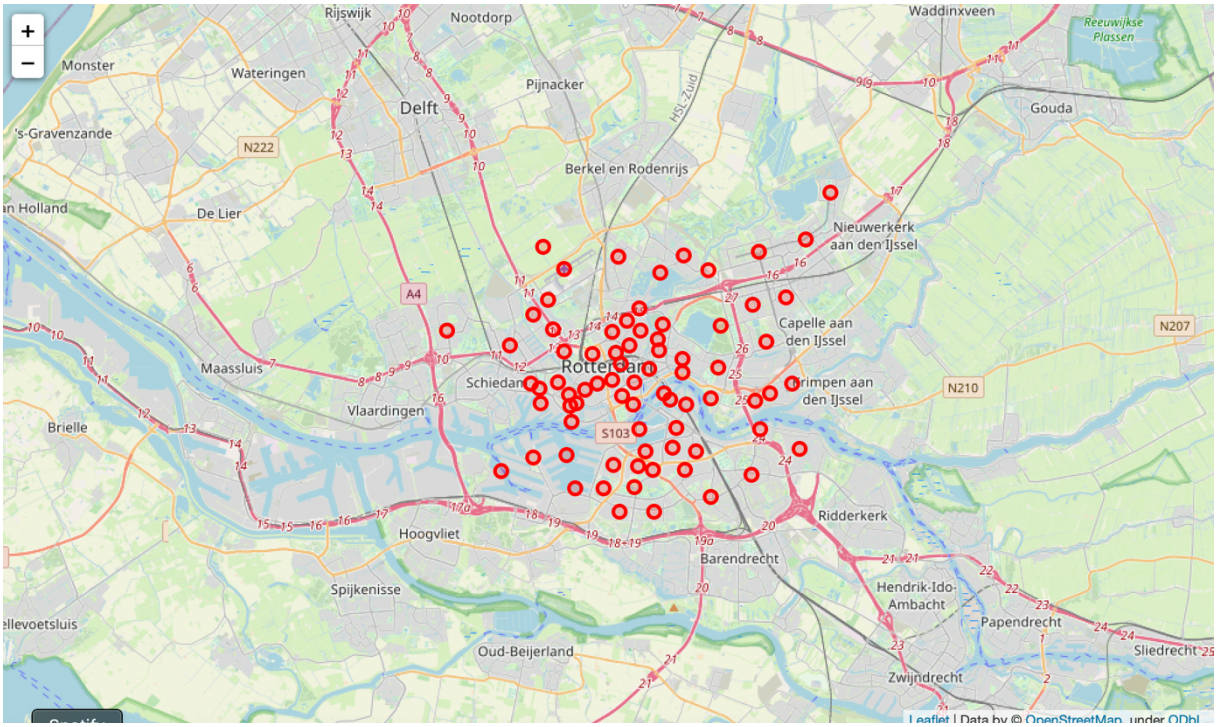
```
rdam_data.loc[rdam_data['Cluster Labels'] == 1, rdam_data.columns[[0] + list(range(4, rdam_data.shape[1]))]]
```

Results

The starting table of this project with all the neighbourhoods with latitude and longitude was as follows:

	Neighbourhood	Latitude	Longitude
0	Stadsdriehoek	51.921768	4.486689
1	Oude Westen	51.918045	4.466252
2	Cool	51.917184	4.477964
3	C.S. kwartier	51.923476	4.471072
5	Nieuwe Werk	51.909355	4.477428
6	Dijkzigt	51.912436	4.471307
7	Delfshaven	51.909995	4.445700
8	Bospolder	51.908980	4.442782
9	Tussendijken	51.913092	4.441788
10	Spangen	51.917315	4.435696
11	Nieuwe Westen	51.914776	4.451135
12	Middelland	51.916655	4.457914
13	Oud-Mathenesse	51.916548	4.420908
14	Witte Dorp	51.914994	4.425384

Luckily I managed to clear the false values and get a complete table with all relevant neighbourhoods and their latitude and longitude. On the map, the neighbourhood were plotted as follows:



After plotting and using Foursquare API, I managed to create a table with all the venues and category of the venues per neighbourhood.

	Neighbourhood	Neighbourhood Latitude	Neighbourhood Longitude	Venue	Venue Category
0	Stadsdriehoek	51.921768	4.486689	Bokaal	Bar
1	Stadsdriehoek	51.921768	4.486689	Markthal	Market
2	Stadsdriehoek	51.921768	4.486689	Little V	Vietnamese Restaurant
3	Stadsdriehoek	51.921768	4.486689	Picknick	Café
4	Stadsdriehoek	51.921768	4.486689	Rotterdamse Centrummarkt	Market

Grouping the data by venue category gave me the following table:

	Neighbourhood	Neighbourhood Latitude	Neighbourhood Longitude	Venue
Venue Category				
Airport	Landzicht	51.945439	4.430210	Vliegclub Rotterdam
Airport Service	Landzicht	51.945439	4.430210	Check-in Transavia
American Restaurant	Rubroek	51.928047	4.492325	Courzand
Aquarium	Blijdorpse polder	51.927815	4.439426	Oceanium
Argentinian Restaurant	Nieuwe Werk	51.954909	4.492864	Gauchos Aan de Maas
Art Gallery	Cool	51.917184	4.477964	TENT
Art Museum	Nieuwe Werk	51.912436	4.477428	Museum Boijmans Van Beuningen
Asian Restaurant	Zuidwijk	51.954909	4.526205	Warung Mirosso
Athletics & Sports	Zuiderpark	51.945439	4.493850	Velox
BBQ Joint	Afrikaanderwijk	51.901375	4.501627	Ortam BBQ
Bagel Shop	Stadsdriehoek	51.934493	4.498008	Bagels & Beans

After One Hot Encoding and grouping by neighbourhood on the mean values, the division of the venue categories per neighbourhood was as shown in the table underneath:

	Neighbourhood	Airport	Airport Service	American Restaurant	Aquarium	Argentinian Restaurant	Art Gallery	Art Museum	Asian Restaurant	Athletics & Sports	...	Vegetarian / Vegan Restaurant	Video Store	Vietnamese Restaurant
0	's-Gravenland	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.000000	0.0	0.0
1	Afrikaanderwijk	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.038462	0.0	...	0.000000	0.0	0.0
2	Agniesebuurt	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.033333	0.0	...	0.033333	0.0	0.0
3	Bergpolder	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.000000	0.0	0.0
4	Beverwaard	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.000000	0.0	0.0

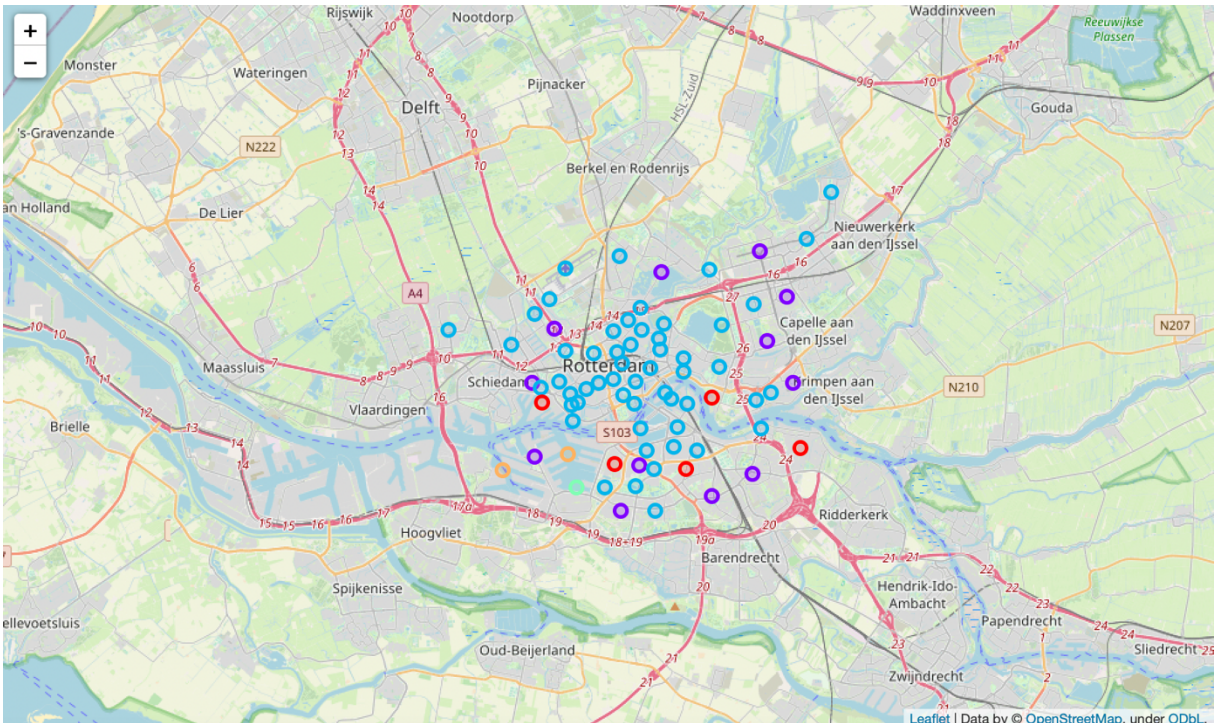
Now I could the determine the most common venues per neighbourhood, table for this is shown below:

	Neighbourhood	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	's-Gravenland	Pharmacy	Playground	Supermarket	Drugstore	Shopping Mall	Falafel Restaurant	Food Court	Food & Drink Shop	Flower Shop	Fish Market
1	Afrikaanderwijk	Supermarket	Bar	Turkish Restaurant	Tram Station	Coffee Shop	Drugstore	Pizza Place	Chinese Restaurant	Middle Eastern Restaurant	Metro Station
2	Agniesebuurt	French Restaurant	Middle Eastern Restaurant	Sandwich Place	Café	Coffee Shop	Pizza Place	Jazz Club	Theater	Karaoke Bar	Supermarket
3	Bergpolder	Supermarket	Tram Station	Grocery Store	Performing Arts Venue	Sporting Goods Shop	Shoe Store	Plaza	Pool	Sports Bar	Middle Eastern Restaurant
4	Beverwaard	Tram Station	Fast Food Restaurant	Zoo Exhibit	Farm	Food Truck	Food Stand	Food Service	Food Court	Food & Drink Shop	Flower Shop

Then I applied K-means and could add the Cluster label as 4th column in the table:

	Neighbourhood	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
0	Stadsdriehoek	51.921768	4.486689	2.0	Italian Restaurant	Plaza	Market	Coffee Shop	Smoke Shop	Gourmet Shop	Szechuan Restaurant	French Restaurant
1	Oude Westen	51.918045	4.466252	2.0	Breakfast Spot	Museum	Bakery	Gastropub	Italian Restaurant	Pool	Beer Bar	Supermarket
2	Cool	51.917184	4.477964	2.0	Bar	Coffee Shop	Café	Sandwich Place	Hostel	Japanese Restaurant	Burger Joint	French Restaurant
3	C.S. kwartier	51.923476	4.471072	2.0	Hotel	Café	Coffee Shop	Clothing Store	Shopping Mall	Mediterranean Restaurant	Bubble Tea Shop	Men's Store
5	Nieuwe Werk	51.909355	4.477428	2.0	Restaurant	French Restaurant	Ice Cream Shop	Bistro	Speakeasy	Gastropub	Sports Bar	Steakhouse

The clusters could now be plotted into the map of Rotterdam, which looked like this:



The final step was to examine the 5 clusters separately, starting with cluster 1:

	Neighbourhood	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
13	Oud-Mathenesse	Bus Stop	Tram Station	Supermarket	Spa	Smoke Shop	Chinese Restaurant	Liquor Store	Fast Food Restaurant	Pizza Place	Farmers Market
16	Kleinpolder	Supermarket	Park	Department Store	Drugstore	Liquor Store	Chinese Restaurant	Bakery	Gym Pool	Gastropub	Bus Stop
30	Hillegersberg-Zuid	Supermarket	Drugstore	Wine Bar	Bar	Department Store	French Restaurant	Candy Store	Cheese Shop	Italian Restaurant	Bakery
51	Lombardijen	Bus Stop	Supermarket	Bar	Bakery	Drugstore	Vegetarian / Vegan Restaurant	Basketball Court	Thai Restaurant	Rock Club	Intersection
52	Groot-IJsselmonde	Supermarket	Department Store	Café	Bar	Discount Store	Bakery	Shopping Mall	Drugstore	Tram Station	Fish Market
55	's-Gravenland	Pharmacy	Playground	Supermarket	Drugstore	Shopping Mall	Falafel Restaurant	Food Court	Food & Drink Shop	Flower Shop	Fish Market
57	Prinsenland	Supermarket	Pharmacy	Gym	Cosmetics Shop	Shopping Mall	Flower Shop	Sporting Goods Shop	Bookstore	Bistro	Park

Cluster 1 seems to have all the necessary venues, but nothing special over there. All the basic venues are there, so no need to go far away for all the basic needs.

Then over the Cluster 2:

	Neighbourhood	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	Stadsdriehoek	Italian Restaurant	Plaza	Market	Coffee Shop	Smoke Shop	Gourmet Shop	Szechuan Restaurant	French Restaurant	Noodle House	Sa Pl
1	Oude Westen	Breakfast Spot	Museum	Bakery	Gastropub	Italian Restaurant	Pool	Beer Bar	Supermarket	Plaza	Pu
2	Cool	Bar	Coffee Shop	Café	Sandwich Place	Hostel	Japanese Restaurant	Burger Joint	French Restaurant	Residential Building (Apartment / Condo)	Bo
3	C.S. kwartier	Hotel	Café	Coffee Shop	Clothing Store	Shopping Mall	Mediterranean Restaurant	Bubble Tea Shop	Men's Store	Lingerie Store	Fr Re
5	Nieuwe Werk	Restaurant	French Restaurant	Ice Cream Shop	Bistro	Speakeasy	Gastropub	Sports Bar	Steakhouse	Beer Garden	Pa
6	Dijkzigt	Bar	French Restaurant	Museum	Bakery	Burger Joint	Art Museum	Hostel	Pizza Place	Coffee Shop	Be
7	Delfshaven	Pub	Restaurant	Historic Site	Supermarket	Park	Drugstore	Coffee Shop	Plaza	Pool Hall	Cl Re
8	Bospolder	Pub	Asian Restaurant	Supermarket	Furniture / Home Store	Drugstore	Pet Store	Coffee Shop	Plaza	Park	Pc

Cluster 2 seems more city-like. It seems like there is a lot to do. In addition, venues like supermarkets and bakeries are included in these neighbourhoods.

Cluster 3 only exists of 1 neighbourhood:

	Neighbourhood	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
81	Waalhaven-Zuid	Boat or Ferry	Sandwich Place	Zoo Exhibit	Farmers Market	Food Truck	Food Stand	Food Service	Food Court	Food & Drink Shop	Flower Shop

This neighbourhood seems to be close to the water, while the most common venue is a boat or ferry. Furthermore it looks like there are a lot of options for food in this neighbourhood.

Cluster 4 exists of 2 neighbourhood and looks quite similar to cluster 3 with a harbor, boat and food nearby:

	Neighbourhood	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
79	Waalhaven	Boat or Ferry	Harbor / Marina	Zoo Exhibit	Farmers Market	Food Truck	Food Stand	Food Service	Food Court	Food & Drink Shop	Flower Shop
80	Eemhaven	Harbor / Marina	Gym	Zoo Exhibit	Food Stand	Food Service	Food Court	Food & Drink Shop	Flower Shop	Fish Market	Fish & Chips Shop

At last, cluster 5, which seem to be close to tram and bus stations. Good to quickly move within the city. In addition, no lack of food in these neighbourhoods as well.

	Neighbourhood	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
40	De Esch	Tram Station	Concert Hall	Supermarket	Field	Restaurant	Shopping Mall	Farm	Food Service	Food Court	Food & Drink Shop
43	Vreewijk	Tram Station	Pizza Place	Kebab Restaurant	Department Store	Farm	Food Stand	Food Service	Food Court	Food & Drink Shop	Flower Shop
53	Beverwaard	Tram Station	Fast Food Restaurant	Zoo Exhibit	Farm	Food Truck	Food Stand	Food Service	Food Court	Food & Drink Shop	Flower Shop
66	Oud-Charlois	Bus Stop	Tram Station	Zoo Exhibit	Farmers Market	Food Truck	Food Stand	Food Service	Food Court	Food & Drink Shop	Flower Shop
78	Nieuw-Mathenesse	Garden	Turkish Restaurant	Tram Station	Fast Food Restaurant	Zoo Exhibit	Farm	Food Stand	Food Service	Food Court	Food & Drink Shop

Discussion

Having examined all clusters and knowing what my personal wish for my new neighbourhood is, I can conclude that cluster 2 is most suitable to look for a new house. Cluster 2 also exists of the most neighbourhoods and looking to the map, it's also the most centralized cluster of all 5. I still have to decide which neighbourhood would be the best, but this model helped me a lot with decreasing the number of choices.

Although the information in this model could be seen as general and not too specific, I really think it can help people to decide where to live in a certain city. For Rotterdam, I would be very happy to apply my model to anyone that is looking to live in the city of Rotterdam. In addition, I would also be very happy to keep continuing improving this model, to give the best advice.

The K-Means Clustering method has again shown that it has really an added value in machine learning. I really enjoyed learning all the machine learning methods and was able to apply 1 in my final project.