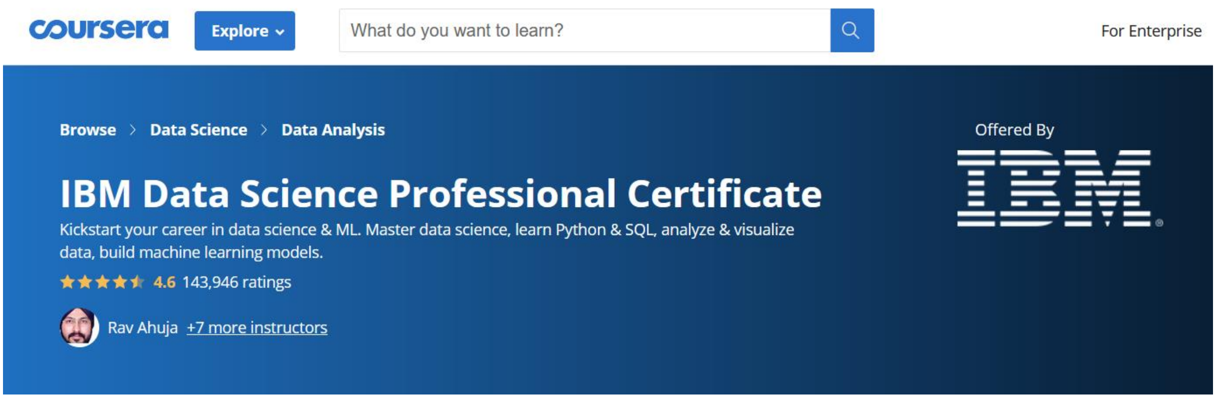

IBM Data Science Professional Certificate

Capstone Project: The Battle of Neighbourhoods

Cologne: Where would I go when I am hungry

The Hunger Games in Cologne

Author: Vladyslav Borysenko

Table of Contents

Table of Contents 2
List of Figures 3
1. Introduction 4
 1.1. Preamble 4
 1.2. Introduction 4
2. Data 5
 2.1. Data acquisition 5
  2.1.1. Cologne Wikipedia 5
  2.1.2. Geocoding 7
  2.1.3. Venues using Foursquare API 7
 2.2. Data cleaning 7
3. Methodology 9
 3.1. Foursquare API 9
 3.2. GeoPy 10
 3.3. Folium 10
 3.4. One-Hot Encoding 11
 3.5. K-means clustering 12
 3.6. Top 10 venues 12
4. Results 13
5. Discussion 15
6. Conclusion 17
7. Acknowledgements 17
8. References 18

List of Figures

Figure 2-1: Wikipedia article with Cologne's districts. Source: (Anon, 2020) 5

Figure 2-2: Cologne districts map. Source: (Anon, 2020) 6

Figure 2-3: Extracting information from Wikipedia. 6

Figure 2-4: Using Geopy to obtain geographical coordinates. 7

Figure 2-5: Venue explorer 7

Figure 2-6: Removing redundant values. 8

Figure 2-7: Missing data check. 8

Figure 3-1: Foursquare API request methods. Source: (Thomas Myer, 2010) 9

Figure 3-2: Latitude and Longitude added to the districts. 10

Figure 3-3: Creating a district map using Folium. 10

Figure 3-4: Resulting Folium map 11

Figure 3-5: One-hot encoding. 11

Figure 3-6: K-means clustering. 12

Figure 3-7: Top 10 venues in each neighbourhood 12

Figure 4-1: Total venues' count 13

Figure 4-2: Venues per district. 13

Figure 4-3: Labeled clusters. 14

Figure 4-4: Final Folium map with five clusters. 14

Figure 5-1: Cluster 1 – German and Greek cuisines + bars. 15

Figure 5-2: Cluster 2 – Italian cuisine + bars. 15

Figure 5-3: Cluster 3 – Europe + Middle East. 16

Figure 5-4: Cluster 4 – Restaurants: Italia + Germany. 16

Figure 5-5: Cluster 5 – Turkish and Mediterranean cuisine + Snacks. 16

1. Introduction

1.1. Preamble

While reading this paper or article, you might have noticed it is all about Cologne. Why? The reason is simple. At the moment of writing the paper, I live in Schweinfurt, a small city in Germany in a province called Bavaria. Schweinfurt is a city with many firms, plants and manufacturing companies such as Schaeffler, ZF, Bosch, SKF, etc. Hence, there are not so many places for recreation and entertainment. Those that are here are good, but their low numbers make it difficult to practice clustering in Schweinfurt. You might say I could have tried to cluster plants, to see which one produces what. I could have, yet the project requirement is to use Foursquare's API, which is not a perfect solution for clustering production plants at all. So, this is how I came up with an idea to focus the project on Cologne.

As my best friend lives and Cologne and I have visited it a few times, we have encountered an issue where we would wander around Cologne and, obviously, get hungry. However, Cologne is a vast city, and even my friend does not know it that well. After having our meals at a few random places, I have decided to use a more structured approach next time I go to Cologne.

Also, while doing the research I have found that this kind of project was already performed for Cologne (Johannes Wagner, 2020), yet I have attempted my own, to get results with the new data.

1.2. Introduction

As it was mentioned, Cologne is a big city and has a lot of venues and places to go to. This fact does not make it easy for tourists to choose where to go and have their meal. And, believe me, tourists do it a lot. For many people, it is the very reason they do tourism in the first place. Many people come to Cologne from different locations and countries wanting to try various or, sometimes, very particular types of cuisines.

But not only that, but locals often want to try something new and travel to the other neighbourhood to visit some places their friends were talking about or simply explore what is there.

Having these concerns in mind, I have basically narrowed it all down to a single question: "If one were to go to that part of Cologne, which kind of food places would they find there?"

Well, now it looks like a perfect case to apply some clustering. Hence, let the Cologne Hunger Games begin!

2. Data

The data was collected from three main sources. It was then cleaned, organised and connected/joined to work in conjunction. In parallel, the resulting data frames were observed and analysed.

2.1.Data acquisition

The first step was to obtain the data on the neighbourhoods of Cologne. Luckily, there were many options available: the public [Wikipedia](), the [official website of the City of Cologne](), and some [other websites]() that specialise on collecting data. Yet, the easiest way was to use Wikipedia for some necessary information about the districts.

2.1.1.Cologne Wikipedia

It was found out that Cologne has 9 (nine) central districts (Figure 2-1).

Districts [edit]

Map	Coat	City district	City parts	Area	Population[1]	Pop. density	District Councils	Town Hall
		District 1 **Köln-Innenstadt**	Altstadt-Nord, Altstadt-Süd, Deutz, Neustadt-Nord, Neustadt-Süd	*16.4 km²*	*127.033*	*7.746/km²*	Bezirksamt Innenstadt Brückenstraße 19, D-50667 Köln	[Insert Image Here]
		District 2 **Köln-Rodenkirchen**	Bayenthal, Godorf, Hahnwald, Immendorf, Marienburg, Meschenich, Raderberg, Raderthal, Rodenkirchen, Rondorf, Sürth, Weiß, Zollstock	*54.6 km²*	*100.936*	*1.850/km²*	Bezirksamt Rodenkirchen Hauptstraße 85, D-50996 Köln	
		District 3 **Köln-Lindenthal**	Braunsfeld, Junkersdorf, Klettenberg, Lindenthal, Lövenich, Müngersdorf, Sülz, Weiden, Widdersdorf	*41.6 km²*	*137.552*	*3.308/km²*	Bezirksamt Lindenthal Aachener Straße 220, 50931 Köln	
		District 4 **Köln-Ehrenfeld**	Bickendorf, Bocklemünd/Mengenich, Ehrenfeld, Neuehrenfeld, Ossendorf, Vogelsang	*23.8 km²*	*103.621*	*4.348/km²*	Bezirksamt Ehrenfeld Venloer Straße 419 – 421, D-50825 Köln	[Insert Image Here]
		District 5 **Köln-Nippes**	Bilderstöckchen, Longerich, Mauenheim, Niehl, Nippes, Riehl, Weidenpesch	*31.8 km²*	*110.092*	*3.462/km²*	Bezirksamt Nippes Neusser Straße 450, D-50733 Köln	
		District 6 **Köln-Chorweiler**	Blumenberg, Chorweiler, Esch/Auweiler, Fühlingen, Heimersdorf, Lindweiler, Merkenich, Pesch, Roggendorf/Thenhoven, Seeberg, Volkhoven/Weiler, Worringen	*67.2 km²*	*80.870*	*1.204/km²*	Bezirksamt Chorweiler Pariser Platz 1, D-50765 Köln	[Insert Image Here]
		District 7 **Köln-Porz**	Eil, Elsdorf, Ensen, Finkenberg, Gremberghoven, Grengel, Langel, Libur, Lind, Poll, Porz, Urbach, Wahn, Wahnheide, Westhoven, Zündorf	*78.8 km²*	*106.520*	*1.352/km²*	Bezirksamt Porz Friedrich-Ebert-Ufer 64–70, D-51143 Köln	
		District 8 **Köln-Kalk**	Brück, Höhenberg, Humboldt/Gremberg, Kalk, Merheim, Neubrück, Ostheim, Rath/Heumar, Vingst	*38.2 km²*	*108.330*	*2.841/km²*	Bezirksamt Kalk Kalker Hauptstraße 247–273, D-51103 Köln	
		District 9 **Köln-Mülheim**	Buchforst, Buchheim, Dellbrück, Dünnwald, Flittard, Höhenhaus, Holweide, Mülheim, Stammheim	*52.2 km²*	*144.374*	*2.764/km²*	Bezirksamt Mülheim Wiener Platz 2a, D-51065 Köln	
		Cologne		***405.15 km²***	***1.019.328***[2]	***2.516/km²***		

Figure 2-1: Wikipedia article with Cologne's districts. Source: (Anon, 2020)

For better understanding Figure 2-2 shows the map that visualises the districts.

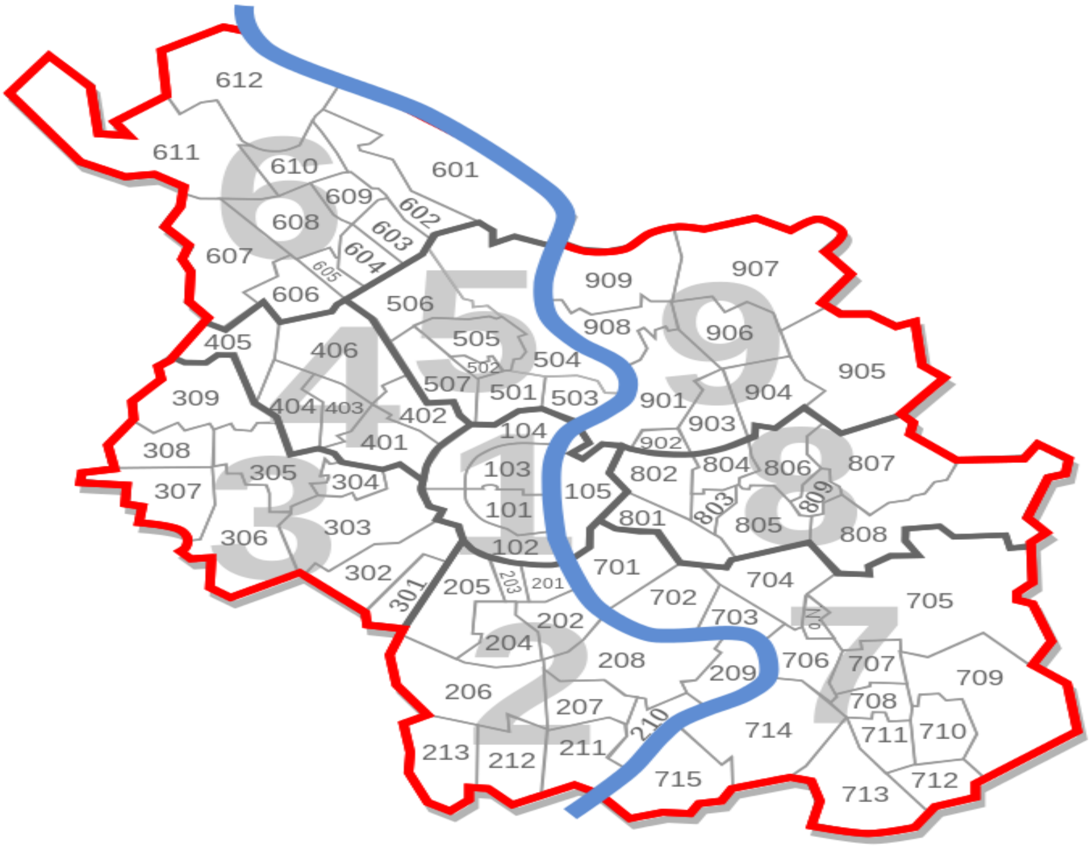

Figure 2-2: Cologne districts map. Source: (Anon, 2020)

With this all in hand, it is easy to extract information from Wikipedia using ‘pandas.read_html’ function (Figure 2-3).

Creating a request with the URL that leads to the Cologne Districts Wiki.

```
[2]: url = "https://en.wikipedia.org/wiki/Districts_of_Cologne"
     res = requests.get(url)
     res
```

```
[2]: <Response [200]>
```

Response 200 means everything is OK. Now, let's use `pandas.read_html`

```
[3]: url_raw = pd.read_html(res.content)
     type(url_raw)
```

```
[3]: list
```

```
[4]: url_raw = pd.read_html(res.content)[1]
     url_raw
```

Figure 2-3: Extracting information from Wikipedia.

2.1.2.Geocoding

The following data obtained was the geographical coordinates of Cologne and its districts. I gave another chance to GeoPy module, and this time it worked perfectly, though producing the working code required some googling and efforts (Figure 2-4).

```
from geopy.geocoders import Nominatim
geolocator = Nominatim(user_agent="Cologne_food_explorer") #getting coordinates for a given address

df['Major_Dist_Coord']= df['City district'].apply(geolocator.geocode).apply(lambda x: (x.latitude, x.longitude)) #creating
df[['Latitude', 'Longitude']] = df['Major_Dist_Coord'].apply(pd.Series) #creating two separate columns with lat and long

df.drop(['Major_Dist_Coord'], axis=1, inplace=True) #dropping the temporary column
df
```

Figure 2-4: Using Geopy to obtain geographical coordinates.

2.1.3.Venues using Foursquare API

Finally, the last piece of data came with the Foursquare API. Like in previous labs, the request was created (actually multiple requests) and the list of venues was composed and put into a data frame (Figure 2-5).

```
def getNearbyVenues(names, latitudes, longitudes, radius=4000, LIMIT=1000):

    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={}'.for
            CLIENT_ID,
            CLIENT_SECRET,
            VERSION,
            lat,
            lng,
            radius,
            LIMIT
            )
```

Full code in the notebook

Figure 2-5: Venue explorer.

2.2.Data cleaning

Some of the data came with NaN values and additional unnecessary information. Therefore, the dataset was cleaned and redundant values removed (Figure 2-6).

This was achieved using pandas “drop” function with the “inplace” argument equal to “True”.

Later on, the final data frame was checked for null values (Figure 2-7) to determine whether further cleaning was necessary, but, luckily, no NaN values were found. In this case, data cleaning did not take much time as the data was structured pretty well and, also, there was not that much data so that missing values would be of great concern. As our research was a product of curiosity, we also had some degree of freedom when working with the data. Yet, no missing data was found.

	Map	Coat	City district	City parts	Area	Population1	Pop. density	District Councils	Town Hall
0	NaN	NaN	District 1 Köln-Innenstadt	Altstadt-Nord, Altstadt-Süd, Deutz, Neustadt-Nord, Neustadt-Süd	16.4 km²	127.033	7.746/km²	Bezirksamt Innenstadt Brückenstraße 19, D-50667 Köln	NaN
1	NaN	NaN	District 2 Köln-Rodenkirchen	Bayenthal, Godorf, Hahnwald, Immendorf, Marienburg, Meschenich, Raderberg, Raderthal, Rodenkirchen, Rondorf, Sürth, Weiß, Zollstock	54.6 km²	100.936	1.850/km²	Bezirksamt Rodenkirchen Hauptstraße 85, D-50996 Köln	NaN
2	NaN	NaN	District 3 Köln-Lindenthal	Braunsfeld, Junkersdorf, Klettenberg, Lindenthal, Lövenich, Müngersdorf, Sülz, Weiden, Widdersdorf	41.6 km²	137.552	3.308/km²	Bezirksamt Lindenthal Aachener Straße 220, 50931 Köln	NaN

```
df = url_raw
df.drop(['Map', 'Coat', 'Town Hall'], axis=1, inplace=True) #drop columns with NaN values
```

```
df.drop([9,10], inplace=True) #drop the two last rows which were redundant
```

```
df
```

	City district	City parts	Area	Population1	Pop. density	District Councils
0	District 1 Köln-Innenstadt	Altstadt-Nord, Altstadt-Süd, Deutz, Neustadt-Nord, Neustadt-Süd	16.4 km²	127.033	7.746/km²	Bezirksamt Innenstadt Brückenstraße 19, D-50667 Köln
1	District 2 Köln-Rodenkirchen	Bayenthal, Godorf, Hahnwald, Immendorf, Marienburg, Meschenich, Raderberg, Raderthal, Rodenkirchen, Rondorf, Sürth, Weiß, Zollstock	54.6 km²	100.936	1.850/km²	Bezirksamt Rodenkirchen Hauptstraße 85, D-50996 Köln
2	District 3 Köln-Lindenthal	Braunsfeld, Junkersdorf, Klettenberg, Lindenthal, Lövenich, Müngersdorf, Sülz, Weiden, Widdersdorf	41.6 km²	137.552	3.308/km²	Bezirksamt Lindenthal Aachener Straße 220, 50931 Köln

Figure 2-6: Removing redundant values.

```
#check for null values
cologne_ffinal[cologne_ffinal['Cluster Labels'].isnull()]
```

Neighbourhood	City parts	Area	Population	Pop. density	District Councils	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

Figure 2-7: Missing data check.

3. Methodology

In order to conduct thorough and meaningful research, the following techniques were employed:

- Foursquare API
- GeoPy
- Folium
- One-Hot Encoding
- K-means clustering
- Top 10 venues

3.1.Foursquare API

Foursquare is a social location service that enables users to explore, rate and comment on the world around them (Thomas Myer, 2010). The service can be used from many portable devices and allows to connect social media accounts to it. More importantly, the Foursquare provides its API to be integrated or used with other apps/tools.

What we needed Foursquare API for was described in paragraph 2.1.3. What is API for? The Foursquare API allows application developers to interact with the Foursquare platform. The API itself is a RESTful set of addresses to which you can send requests, so there's really nothing to download onto your server (Thomas Myer, 2010).

For many advanced features, authentication is needed, but, luckily, in our case, it is not required to obtain a list of venues and their descriptions (Figure 3-1).

Method	Summary	URL	Parameters	HTTP method(s)	Authentication?
Checkins	Returns a list of recent check-ins from friends.	http://api.foursquare.com /v1/checkins	geolat, geolong	GET	Yes
Venues	Returns a list of venues near the area specified.	http://api.foursquare.com /v1/venues	geolat, geolong, l, q	GET	No
Venue detail	Returns venue data for a given venue ID.	http://api.foursquare.com /v1/venue	vid	GET	No
Categories	Returns a hierarchical category list.	http://api.foursquare.com /v1/categories	n/a	GET	No

Figure 3-1: Foursquare API request methods. Source: (Thomas Myer, 2010)

3.2.GeoPy

As mentioned before, I had hard times with GeoPy during the labs, but it is the best way to get geographical coordinates quickly. Therefore, I gave it another go. Fortunately, it worked this time, and I managed to use "Nominatim" function to add latitude and longitude to the data frame containing the information on Cologne's districts.

The way it was done is shown in Figure 2-4 in paragraph 2.1.2.

The resulting data frame looked like this (Figure 3-2).

	City district	City parts	Area	Population	Pop. density	District Councils	Latitude	Longitude
0	Köln-Innenstadt	Altstadt-Nord, Altstadt-Süd, Deutz, Neustadt-Nord, Neustadt-Süd	16.4 km²	127.033	7.746/km²	Bezirksksamt Innenstadt Brückenstraße 19, D-50667 Köln	50.937328	6.959234
1	Köln-Rodenkirchen	Bayenthal, Godorf, Hahnwald, Immendorf, Marienburg, Meschenich, Raderberg, Raderthal, Rodenkirchen, Rondorf, Sürth, Weiß, Zollstock	54.6 km²	100.936	1.850/km²	Bezirksamt Rodenkirchen Hauptstraße 85, D-50996 Köln	50.865622	6.969718
2	Köln-Lindenthal	Braunsfeld, Junkersdorf, Klettenberg, Lindenthal, Lövenich, Müngersdorf, Sülz, Weiden, Widdersdorf	41.6 km²	137.552	3.308/km²	Bezirksamt Lindenthal Aachener Straße 220, 50931 Köln	50.935935	6.871246
3	Köln-Ehrenfeld	Bickendorf, Bocklemünd/Mengenich, Ehrenfeld, Neuehrenfeld, Ossendorf, Vogelsang	23.8 km²	103.621	4.348/km²	Bezirksamt Ehrenfeld Venloer Straße 419 – 421, D-50825 Köln	50.951502	6.916529

Figure 3-2: Latitude and Longitude added to the districts.

3.3.Folium

Folium is a powerful library that uses Python to produce interactive maps using OpenStreetMap technology.

I have used the benefits of Folium and my data frame to plot all nine Cologne districts on a map using the location data I got via GeoPy earlier (Figure 3-3).

```python
import folium

Cologne_map = folium.Map(location=[latitude, longitude], zoom_start=11)

for latitude, longitude, dist in zip(df['Latitude'], df['Longitude'], df['City district']):
    dist = folium.Popup(dist, parse_html=True)
    folium.CircleMarker(
        [latitude, longitude],
        radius=5,
        popup=dist,
        color='green',
        fill=True
        ).add_to(Cologne_map)

Cologne_map
```

Figure 3-3: Creating a district map using Folium.

As a result, the following map was produced by Folium, having all the nine districts neatly plotted according to their centroids (Figure 3-4).

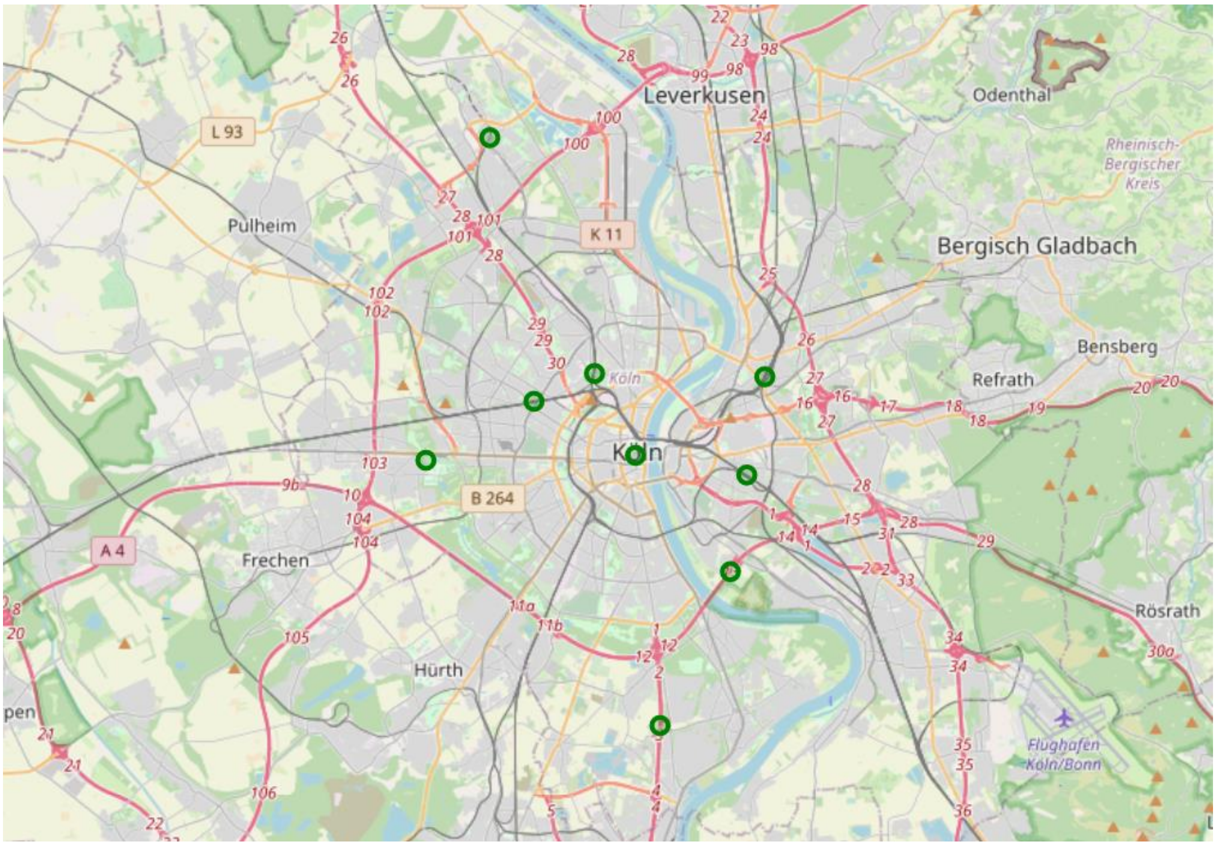

Figure 3-4: Resulting Folium map.

3.4.One-Hot Encoding

One-Hot Encoding is a technique which ensures that machine learning algorithms can process data. Namely, it converts categorical variables into the binary Boolean ones.

In our case, it was all about making it, so when the venue was present in the neighbourhood (value: True), then this venue gets a value of 1 (one) when listed in a given area. Similarly, if the venue is not present in the neighbourhood (value: False), then it gets a numerical value of 0 (zero) for the given city area (Figure 3-5).

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

	Asian Restaurant	Bar	Beer Bar	Cocktail Bar	Doner Restaurant	Eastern European Restaurant	Falafel Restaurant	Fast Food Restaurant	Food & Drink Shop	French Restaurant	German Restaurant	Greek Restaurant	Hookah Bar	Indian Restaurant	Res
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	1	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Figure 3-5: One-hot encoding.

3.5.K-means clustering

It was decided to use the K-means clustering algorithm for the classification. The number of clusters was selected to five. I could have tried some algorithms to try to find an optimal number of clusters, like Silhouette Score, but for me, five clusters sounded reasonable to go for it.

The model was then trained and labelled accordingly (Figure 3-6).

```
from sklearn.cluster import KMeans

k_clusters = 5

#drop the Neighbourhood column to work with numerical values only
cologne_k_clustering = cologne_grouped.drop('Neighbourhood', 1)

KM = KMeans(n_clusters=k_clusters, random_state=0)
```

```
KM.fit(cologne_k_clustering)
KM
```

```
KMeans(n_clusters=5, random_state=0)
```

```
KM.labels_[0:10]
```

```
array([3, 1, 1, 2, 0, 4, 1, 2, 0])
```

Figure 3-6: K-means clustering.

3.6.Top 10 venues

Also, to be able to work with each cluster and adequately classify it, I needed to determine the top 10 venues in each one. For this, I used the technique from the previous labs. With its help and some NumPy magic, the following table was obtained (Figure 3-7).

	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	Köln-Chorweiler	Italian Restaurant	German Restaurant	Restaurant	Sushi Restaurant	Fast Food Restaurant	Hookah Bar	Snack Place	Pizza Place	Scandinavian Restaurant	Schnitzel Restaurant
1	Köln-Ehrenfeld	Italian Restaurant	Bar	Cocktail Bar	Tapas Restaurant	German Restaurant	Restaurant	Sushi Restaurant	Korean Restaurant	Vietnamese Restaurant	Turkish Restaurant
2	Köln-Innenstadt	Italian Restaurant	Cocktail Bar	French Restaurant	Sushi Restaurant	Israeli Restaurant	Snack Place	Restaurant	Pizza Place	Modern European Restaurant	Middle Eastern Restaurant
3	Köln-Kalk	Italian Restaurant	French Restaurant	Middle Eastern Restaurant	Vietnamese Restaurant	Vegetarian / Vegan Restaurant	Turkish Restaurant	German Restaurant	Pizza Place	Asian Restaurant	Spanish Restaurant

Figure 3-7: Top 10 venues in each neighbourhood.

4. Results

Before the final clustering results were produced, some preliminary insights were extracted from the intermediate data frames.

For example, Figure 4-1 shows the overall count of all venues that were parsed through the Foursquare API within the entire Cologne. With this, it was clear which types of food'n'drink places dominate the city.

```
print (Cologne_Food['Venue Category'].value_counts())

Italian Restaurant              37
German Restaurant               28
Bar                             16
Turkish Restaurant              14
Restaurant                      13
Greek Restaurant                11
French Restaurant               11
Sushi Restaurant                 9
Cocktail Bar                     9
Pizza Place                      8
Tapas Restaurant                 7
Vietnamese Restaurant            7
Vegetarian / Vegan Restaurant    6
Middle Eastern Restaurant        6
Snack Place                      5
Spanish Restaurant               4
Mediterranean Restaurant         4
```

Figure 4-1: Total venues' count.

Another filtering step allowed me to look at venues' count per district, basically seeing where one should go if they wanted to eat (Figure 4-2).

```
Cologne_Food[['Neighbourhood','Venue']].groupby('Neighbourhood').count()
```

	Venue
Neighbourhood	
Köln-Chorweiler	8
Köln-Ehrenfeld	37
Köln-Innenstadt	35
Köln-Kalk	26
Köln-Lindenthal	30
Köln-Mülheim	24
Köln-Nippes	27
Köln-Porz	33
Köln-Rodenkirchen	17

Figure 4-2: Venues per district.

Well, I guess, living in Chorweiler is tough :) Yet, it could be that our API calls did not return all the existing venues in there.

Finally, using the Top 10 venues list, I showed in paragraph 3.6, I was able to use it for K-means clustering. Hence, the clusters were labelled and ready to be mapped (Figure 4-3).

	Neighbourhood	City parts	Area	Population	Pop. density	District Councils	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Co
0	Köln-Innenstadt	Altstadt-Nord, Altstadt-Süd, Deutz, Neustadt-Nord, Neustadt-Süd	16.4 km²	127.033	7.746/km²	Bezirksksamt Innenstadt Brückenstraße 19, D-50667 Köln	50.937328	6.959234	1	Italian Restaurant	Cocktail Bar	Rest
1	Köln-Rodenkirchen	Bayenthal, Godorf, Hahnwald, Immendorf, Marienburg, Meschenich, Raderberg, Raderthal, Rodenkirchen, Rondorf, Sürth, Weiß, Zollstock	54.6 km²	100.936	1.850/km²	Bezirksamt Rodenkirchen Hauptstraße 85, D-50996 Köln	50.865622	6.969718	0	German Restaurant	Greek Restaurant	Rest

Figure 4-3: Labeled clusters.

In the end, this map (Figure 4-4) was produced based on the data frame above.

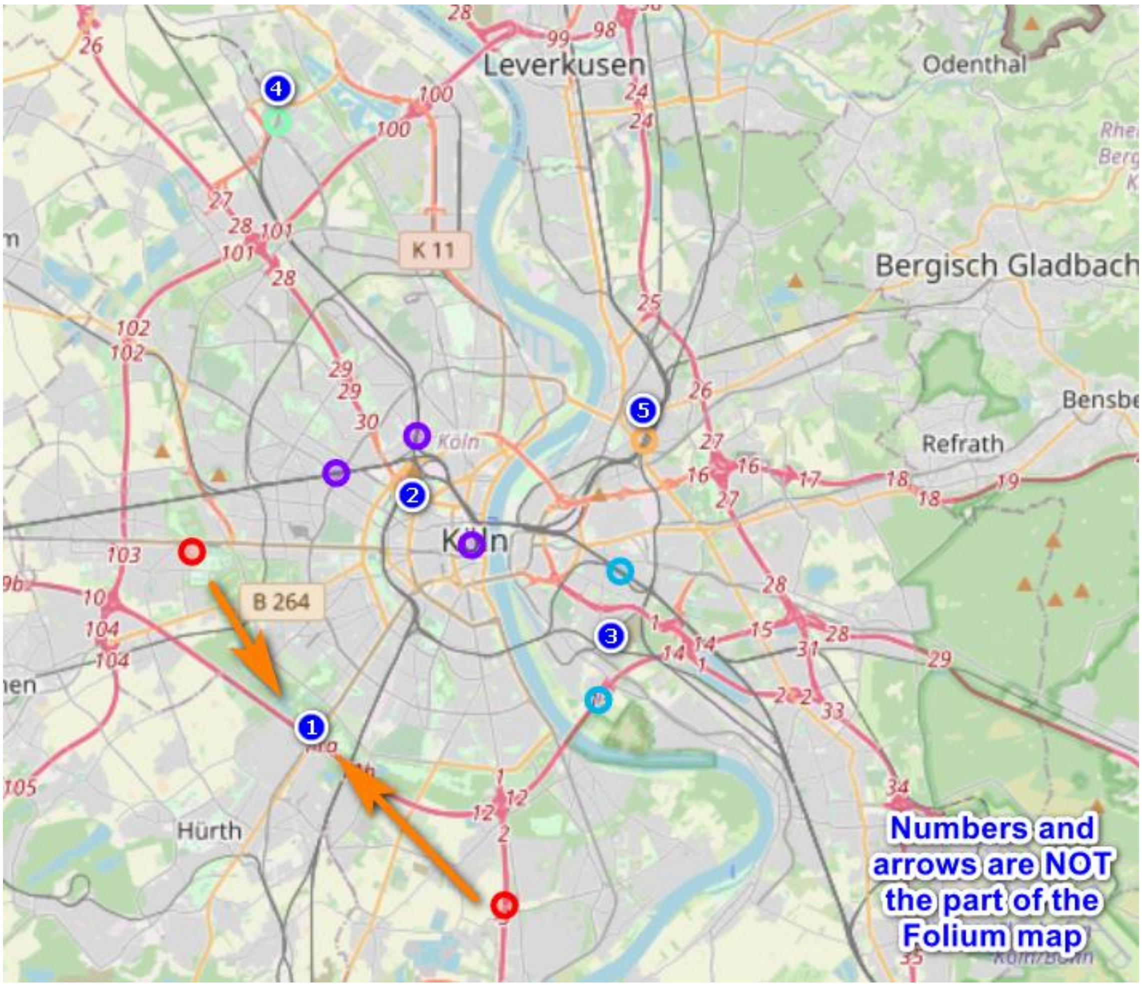

Figure 4-4: Final Folium map with five clusters.

5. Discussion

So, after the algorithm classified the clusters according to the venues each district had, I had a look at each one to try to give my own classification. And here is what I got.

Cluster 1 was dominated by German and Greeks cuisines with some occasional bars. On a closer look, there were enough Italian restaurants too (Figure 5-1).

	City parts	District Councils	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
1	Bayenthal, Godorf, Hahnwald, Immendorf, Marienburg, Meschenich, Raderberg, Raderthal, Rodenkirchen, Rondorf, Sürth, Weiß, Zollstock	Bezirksamt Rodenkirchen Hauptstraße 85, D-50996 Köln	50.865622	6.969718	0	German Restaurant	Greek Restaurant	Italian Restaurant	Fast Food Restaurant	Restaurant	Scandinavian Restaurant
2	Braunsfeld, Junkersdorf, Klettenberg, Lindenthal, Lövenich, Müngersdorf, Sülz, Weiden, Widdersdorf	Bezirksamt Lindenthal Aachener Straße 220, 50931 Köln	50.935935	6.871246	0	German Restaurant	Bar	Greek Restaurant	Restaurant	Sushi Restaurant	Italian Restaurant

Cluster 1

Figure 5-1: Cluster 1 – German and Greek cuisines + bars.

Cluster 2 was dominated by Italian restaurants and various types of bars (Figure 5-2).

	City parts	District Councils	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
0	Altstadt-Nord, Altstadt-Süd, Deutz, Neustadt-Nord, Neustadt-Süd	Bezirksksamt Innenstadt Brückenstraße 19, D-50667 Köln	50.937328	6.959234	1	Italian Restaurant	Cocktail Bar	French Restaurant	Sushi Restaurant	Israeli Restaurant	Snack Place
3	Bickendorf, Bocklemünd/Mengenich, Ehrenfeld, Neuehrenfeld, Ossendorf, Vogelsang	Bezirksamt Ehrenfeld Venloer Straße 419 – 421, D-50825 Köln	50.951502	6.916529	1	Italian Restaurant	Bar	Cocktail Bar	Tapas Restaurant	German Restaurant	Restaurant
4	Bilderstöckchen, Longerich, Mauenheim, Niehl, Nippes, Riehl, Weidenpesch	Bezirksamt NippesNeusser Straße 450,D-50733 Köln	50.958994	6.941777	1	Bar	Italian Restaurant	Restaurant	Cocktail Bar	French Restaurant	German Restaurant

Cluster 2

Figure 5-2: Cluster 2 – Italian cuisine + bars.

Cluster 3 was rather diverse in its cuisines uniting many European types, as well as those of the Middle East (Figure 5-3).

	City parts	District Councils	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
6	Eil, Elsdorf, Ensen, Finkenberg, Gremberghoven, Grengel, Langel, Libur, Lind, Poll, Porz, Urbach, Wahn, Wahnheide, Westhoven, Zündorf	Bezirksamt PorzFriedrich-Ebert-Ufer 64–70, D-51143 Köln	50.906705	6.999129	2	German Restaurant	Pizza Place	Italian Restaurant	Middle Eastern Restaurant	Seafood Restaurant	Greek Restaurant
7	Brück, Höhenberg, Humboldt/Gremberg, Kalk, Merheim, Neubrück, Ostheim, Rath/Heumar, Vingst	Bezirksamt KalkKalker Hauptstraße 247–273,D-51103 Köln	50.931923	7.005806	2	Italian Restaurant	French Restaurant	Middle Eastern Restaurant	Vietnamese Restaurant	Vegetarian / Vegan Restaurant	Turkish Restaurant

Cluster 3

Figure 5-3: Cluster 3 – Europe + Middle East.

Cluster 4 was the restaurant oriented. The most popular restaurants still were of German and Italian cuisine (Figure 5-4).

	City parts	District Councils	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
5	Blumenberg, Chorweiler, Esch/Auweiler, Fühlingen, Heimersdorf, Lindweiler, Merkenich, Pesch, Roggendorf/Thenhoven, Seeberg, Volkhoven/Weiler, Worringen	Bezirksamt Chorweiler Pariser Platz 1, D-50765 Köln	51.021167	6.898034	3	Italian Restaurant	German Restaurant	Restaurant	Sushi Restaurant	Fast Food Restaurant	Hookah Bar

Cluster 4

Figure 5-4: Cluster 4 – Restaurants: Italia + Germany.

Cluster 5 was comprised of venues of Turkish and Mediterranean cuisines. And, of course, one could enjoy some snack while being in Cluster 5.

	City parts	District Councils	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
8	Buchforst, Buchheim, Dellbrück, Dünnwald, Flittard, Höhenhaus, Holweide, Mülheim, Stammheim	Bezirksamt Mülheim Wiener Platz 2a,D-51065 Köln	50.958147	7.013526	4	Turkish Restaurant	Snack Place	Mediterranean Restaurant	Italian Restaurant	German Restaurant	Asian Restaurant

Cluster 5

Figure 5-5: Cluster 5 – Turkish and Mediterranean cuisine + Snacks.

6. Conclusion

While the algorithm did its best, it is a bit underwhelming to see that Italian restaurants dominate Cologne. Yet, the classification sounds reasonable, and, I would say, was performed correctly.
Perhaps, finding the optimal number of clusters could have been better, but as I explored Cologne only, five clusters look good now, mainly because they are similar due to mostly being comprised of Italian and German cuisine venues.
As a personal opinion, I might add that this kind of research was performed in a matter of days (mostly because of me being busy) and could have been done in a matter of hours. This fact is impressive as Data Science is now available to anyone who has a laptop and proper googling skills. Some programming basics are "good-to-have", but could absolutely be gained via googling or going through courses just like this one.
In the end, I would say that the research question was answered, the venues were clustered, and tourists now can use the information to decide to which part of a city to go to enjoy their favourite cuisine.

7. Acknowledgements

This work references the laboratory projects of IBM Data Science Professional certificate course on Coursera. A lot of code was based on the one used in those labs, and other parts were either written by me entirely or with the help of wonderful people from Stack Overflow and other people who had taken this course. In particular, big thanks to Pritthijit Nath and Dr Johannes Wagner (Johannes Wagner, 2020) whose works I have used as a guideline for my project.

8. References

Anon (2020) ‘Districts of Cologne’, *Wikipedia* [Online]. Available at https://en.wikipedia.org/w/index.php?title=Districts_of_Cologne&oldid=969298227 (Accessed 28 December 2020).

Johannes Wagner (2020) *Applied Data Science Capstone Project — The Restaurant Battle of Neighborhoods in Cologne | LinkedIn* [Online]. Available at https://www.linkedin.com/pulse/applied-data-science-capstone-project-restaurant-wagner-mba/?articleId=6670274875946622976 (Accessed 28 December 2020).

Thomas Myer (2010) *Introduction to the Foursquare API* [Online]. Available at http://www.ibm.com/developerworks/library/os-foursquare/index.html (Accessed 29 December 2020).