IBM Applied Data Science Capstone: A Peek into Cologne's Bars and Restaurants Scene

By Anderson Matias

May 18th, 2020

1. Introduction and Business Problem Statement

Cologne, Germany. Founded in 38 BC, has a population of more than 1 million people, making it the fourth largest city in the country. The city is part of the state of North-Rhine Westphalia, forming together with cities like Düsseldorf, Essen, Bonn, and a couple others, the Rhine-Ruhr area, known as the largest metropolitan region in Germany. It is also located in the far west of the country, having the borders of Belgium and Netherlands within a 100km range and being well-centered in Western Europe.

Home of the Kölner Dom, the tallest twin-spired Church in the world, Cologne is known for its openness and multicultural vibe, with a vibrant nightlife and a variety of options for going out, including bars, clubs and restaurants from all kinds of places in the world. It is home for some of the most relevant and biggest events around, besides being part of a circuit of best music concerts and tours one can find in the region.

This mix of history, entertainment and privileged location makes it a particularly good destination for tourists and a nice place to live. In this context, investing in the entertainment business in the city is often considered a good option, but then this brings a couple of questions, such as: what kind of establishment makes more sense to be opened? which locations would be best to start such a business? How competitive will be the chosen market? These are some of the questions we propose to answer with this analysis.

1.1 The Problem Statement: Which Cologne neighborhood is the best to start a bar or restaurant?

In the proposed hypothetical situation, a client is thinking about investing in opening an establishment in the city of Cologne, so he asked to analyze the available data and cross-reference it against some criteria in order to take a decision. Below further details:

- Two types of establishment were defined as initial targets: a bar or restaurant. It must be one of this two categories; therefore, the analysis will prioritize those.
- It is required that the new establishment be located where there is a good amount of people, preferably not so far away from the city center.
- On the other hand, it is important that his establishment can stand out, so a neighborhood with too much competition in the same category should be avoided.
- The place will be rented, and the client is willing to pay up to 12€ per square meter.

1.2 The Target Audience

Besides the obvious interest coming from the client in taking a data-driven decision about where to invest his money on, this study can be helpful to a couple of other stakeholders, such as:

- Other investors with a similar interest in investing in the entertainment and gastronomy business in the city of Cologne.
- Fellow Data Scientists that are looking for ideas or references on how to carry out some tasks that might be applicable to their projects, not just for Cologne, but anywhere in the world.
- Cologne Citizens in general, to get further insights on the city they live in and on which neighborhoods there are more options of venues to go and check out.

2. Data Gathering, Wrangling and Sorting Criteria

2.1 Description

In order to perform the analysis, a combination of geolocation data called from Foursquare and external sources found on the internet will be used. In general, the required data to be collected is as follows:

- Venues position and insights: Foursquare API.
- Details on Cologne Districts: web scraping on the internet.

Data frames will be generated and cross-referenced in search for key information under the given criteria, which are:

- Rent costs: no more than 12€ per square meter.
- Demographic: minimum population density of 3000 people per squared kilometer
- Competition: the chosen district cannot have more than 20 establishments of the same category as the one to be chosen
- Location: as close to the city center as possible

2.2 Considerations on data quality and source choices

When it comes to the data quality and reliability throughout the analysis, the level of confidence on the accuracy of the datasets is assumed fairly high, given the sources used and broad availability of alternative pieces of data on the internet, that corroborate the pieces of information used. Here it is important to say that although the usage of Foursquare API as main source of geolocation data for the venues is imposed as one of the Capstone course requirements, the API is widely known for sourcing data to highly used and trusted applications, such as Apple's Maps and Foursquare City Guide itself.

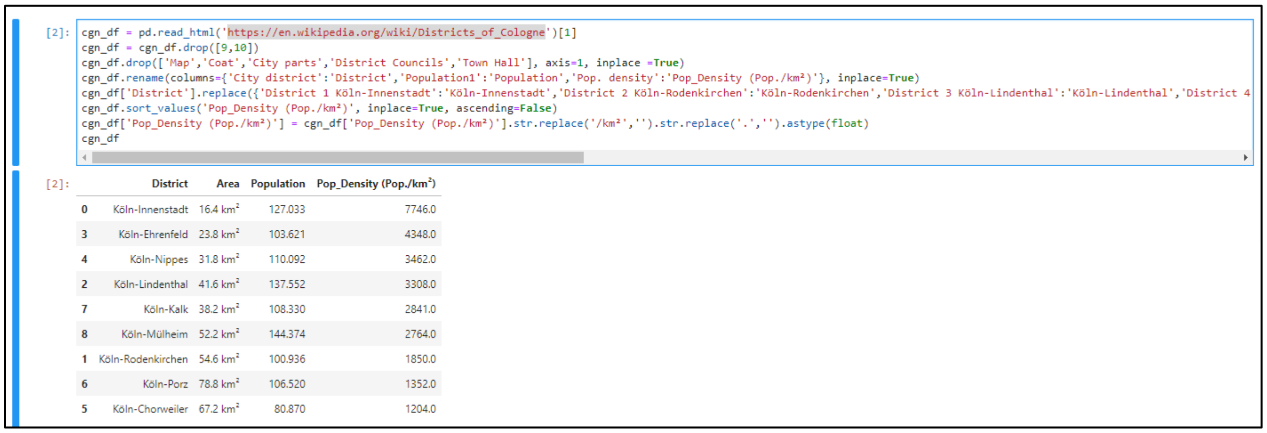
Regarding the data about the city, the accuracy of neighborhoods names, distribution, population density, latitude and longitude was assessed through a combination of cross-checking several sources and using Google Maps to validate that the geolocation was in order. The latter was necessary because Geocoder was used to find the respective coordinates and given it is essentially based on keyword searching out the Data Frame itself, mismatches could in a first moment occur.

One key concern though, was the rent price data available, for a couple reasons. The first one is that there are substantial differences between sources (there is a lot of info available based on real estate listings and websites, which might be biased somehow). Another point was that quite some sources could not be used as there was no distribution per district, making a comparison not feasible. At last, data that was too old (such as 10+ years old reports or so) could also not be considered. Given that, the criteria applied to define where the rent data would come from was based on how reputable the website to be scraped was and from which year would the report be.

3. Methodology and Exploratory Data Analysis

3.1 Web scraping Cologne Districts Data and Cleaning

First, I have collected a list of the districts of Cologne on Wikipedia, sorted, formatted and renamed as per our needs:

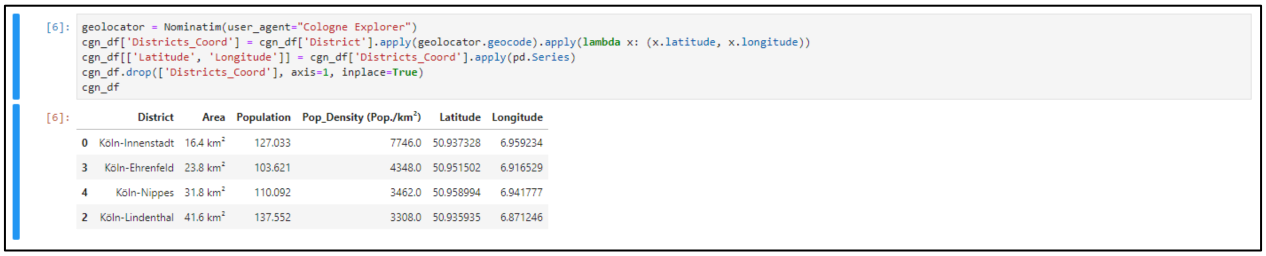


```
[2]: cgn_df = pd.read_html('https://en.wikipedia.org/wiki/Districts_of_Cologne')[1]
     cgn_df = cgn_df.drop([9,10])
     cgn_df.drop(['Map','Coat','City parts','District Councils','Town Hall'], axis=1, inplace =True)
     cgn_df.rename(columns={'City district':'District','Population1':'Population','Pop. density':'Pop_Density (Pop./km²)'}, inplace=True)
     cgn_df['District'].replace({'District 1 Köln-Innenstadt':'Köln-Innenstadt','District 2 Köln-Rodenkirchen':'Köln-Rodenkirchen','District 3 Köln-Lindenthal':'Köln-Lindenthal','District 4
     cgn_df.sort_values('Pop_Density (Pop./km²)', inplace=True, ascending=False)
     cgn_df['Pop_Density (Pop./km²)'] = cgn_df['Pop_Density (Pop./km²)'].str.replace('/km²','').str.replace('.','').astype(float)
     cgn_df
```

[2]:

	District	Area	Population	Pop_Density (Pop./km²)
0	Köln-Innenstadt	16.4 km²	127.033	7746.0
3	Köln-Ehrenfeld	23.8 km²	103.621	4348.0
4	Köln-Nippes	31.8 km²	110.092	3462.0
2	Köln-Lindenthal	41.6 km²	137.552	3308.0
7	Köln-Kalk	38.2 km²	108.330	2841.0
8	Köln-Mülheim	52.2 km²	144.374	2764.0
1	Köln-Rodenkirchen	54.6 km²	100.936	1850.0
6	Köln-Porz	78.8 km²	106.520	1352.0
5	Köln-Chorweiler	67.2 km²	80.870	1204.0

The next step was to find and append the coordinate data for each of the districts. For that, I have used the geolocator library:



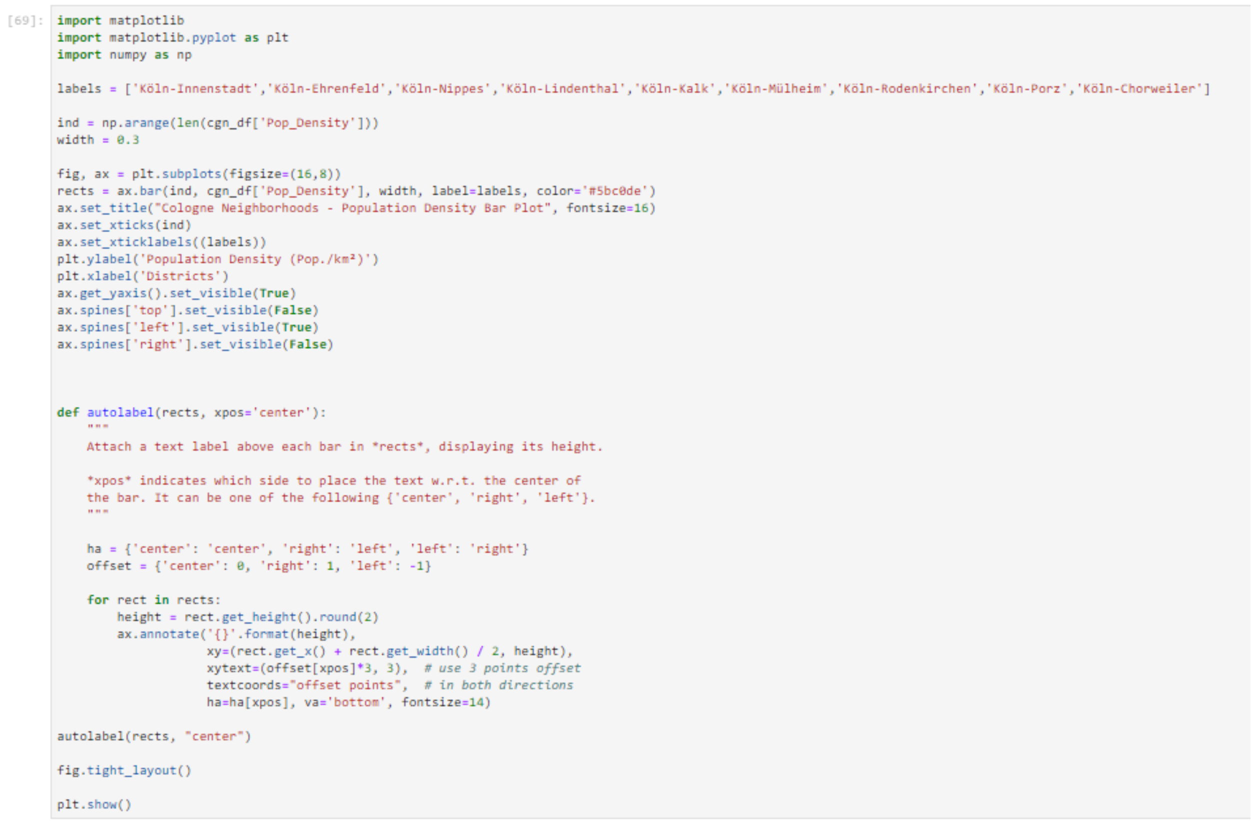
```
[6]: geolocator = Nominatim(user_agent="Cologne Explorer")
     cgn_df['Districts_Coord'] = cgn_df['District'].apply(geolocator.geocode).apply(lambda x: (x.latitude, x.longitude))
     cgn_df[['Latitude', 'Longitude']] = cgn_df['Districts_Coord'].apply(pd.Series)
     cgn_df.drop(['Districts_Coord'], axis=1, inplace=True)
     cgn_df
```

[6]:

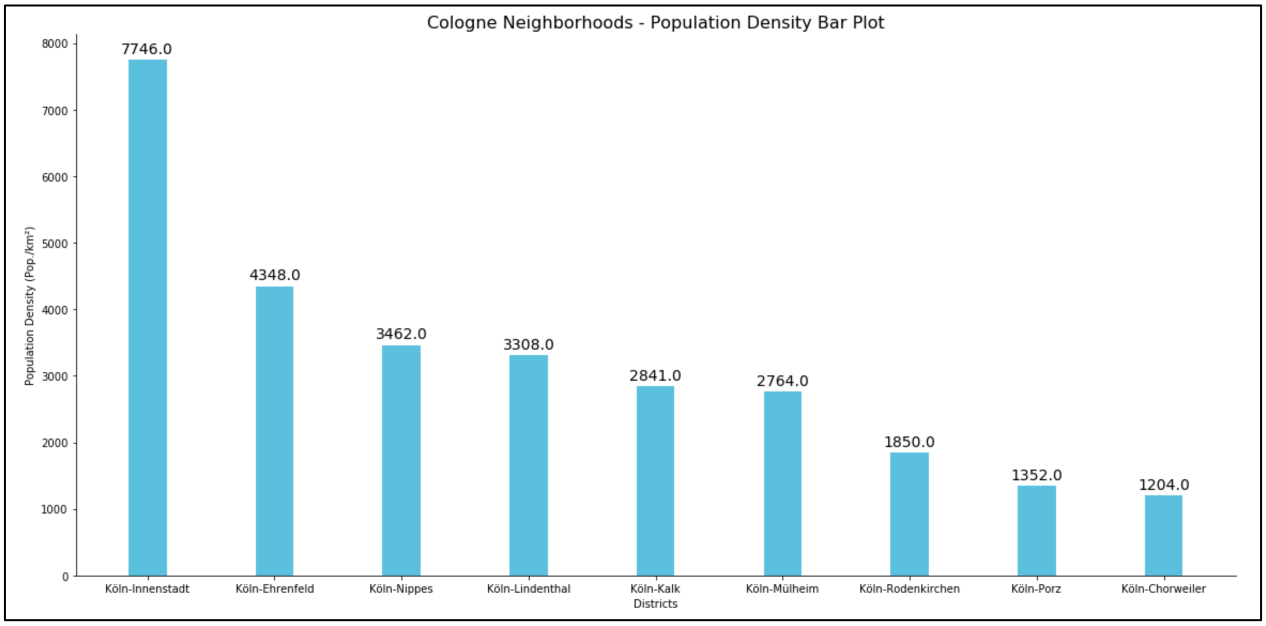
	District	Area	Population	Pop_Density (Pop./km²)	Latitude	Longitude
0	Köln-Innenstadt	16.4 km²	127.033	7746.0	50.937328	6.959234
3	Köln-Ehrenfeld	23.8 km²	103.621	4348.0	50.951502	6.916529
4	Köln-Nippes	31.8 km²	110.092	3462.0	50.958994	6.941777
2	Köln-Lindenthal	41.6 km²	137.552	3308.0	50.935935	6.871246

Please note, that with the population density data already scraped it was possible to rule out a couple of neighborhoods that did not fall into the criteria. As a result, out of 9 neighborhoods that are part of the city of Cologne, remained:

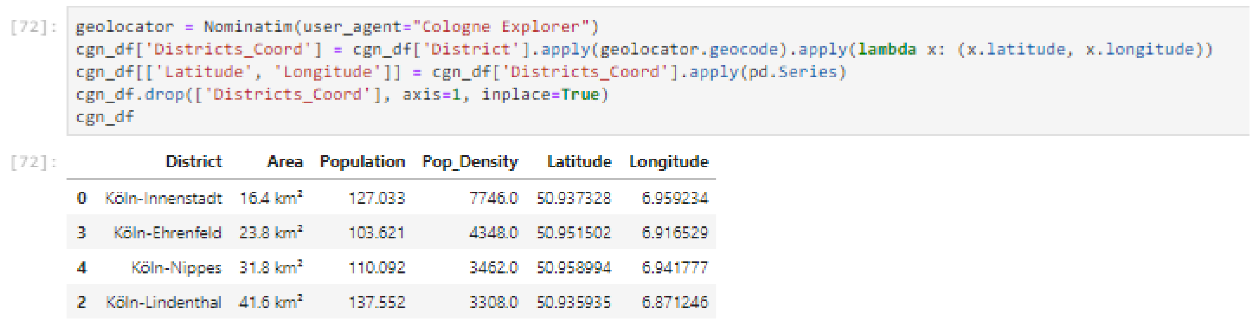
- Köln-Innenstadt (also often referenced as the city center)
- Köln-Ehrenfeld
- Köln-Nippes
- Köln-Lindenthal



```
[69]: import matplotlib
      import matplotlib.pyplot as plt
      import numpy as np

      labels = ['Köln-Innenstadt','Köln-Ehrenfeld','Köln-Nippes','Köln-Lindenthal','Köln-Kalk','Köln-Mülheim','Köln-Rodenkirchen','Köln-Porz','Köln-Chorweiler']

      ind = np.arange(len(cgn_df['Pop_Density']))
      width = 0.3

      fig, ax = plt.subplots(figsize=(16,8))
      rects = ax.bar(ind, cgn_df['Pop_Density'], width, label=labels, color='#5bc0de')
      ax.set_title("Cologne Neighborhoods - Population Density Bar Plot", fontsize=16)
      ax.set_xticks(ind)
      ax.set_xticklabels((labels))
      plt.ylabel('Population Density (Pop./km²)')
      plt.xlabel('Districts')
      ax.get_yaxis().set_visible(True)
      ax.spines['top'].set_visible(False)
      ax.spines['left'].set_visible(True)
      ax.spines['right'].set_visible(False)



      def autolabel(rects, xpos='center'):
          """
          Attach a text label above each bar in *rects*, displaying its height.

          *xpos* indicates which side to place the text w.r.t. the center of
          the bar. It can be one of the following {'center', 'right', 'left'}.
          """

          ha = {'center': 'center', 'right': 'left', 'left': 'right'}
          offset = {'center': 0, 'right': 1, 'left': -1}

          for rect in rects:
              height = rect.get_height().round(2)
              ax.annotate('{}'.format(height),
                          xy=(rect.get_x() + rect.get_width() / 2, height),
                          xytext=(offset[xpos]*3, 3),  # use 3 points offset
                          textcoords="offset points",  # in both directions
                          ha=ha[xpos], va='bottom', fontsize=14)

      autolabel(rects, "center")

      fig.tight_layout()

      plt.show()
```



The next step was to search for the respective coordinates of each district, which I did through Geocoder:

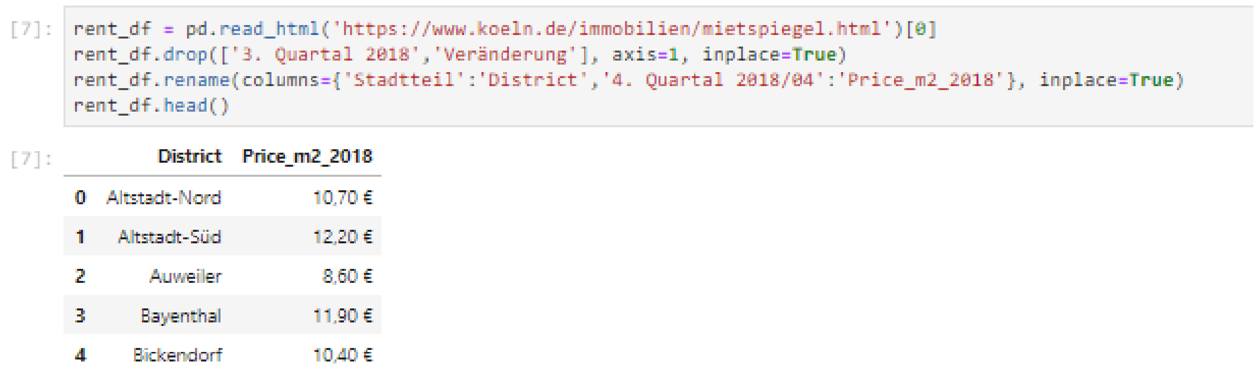


```
[72]: geolocator = Nominatim(user_agent="Cologne Explorer")
      cgn_df['Districts_Coord'] = cgn_df['District'].apply(geolocator.geocode).apply(lambda x: (x.latitude, x.longitude))
      cgn_df[['Latitude', 'Longitude']] = cgn_df['Districts_Coord'].apply(pd.Series)
      cgn_df.drop(['Districts_Coord'], axis=1, inplace=True)
      cgn_df
```

[72]:

	District	Area	Population	Pop_Density	Latitude	Longitude
0	Köln-Innenstadt	16.4 km²	127.033	7746.0	50.937328	6.959234
3	Köln-Ehrenfeld	23.8 km²	103.621	4348.0	50.951502	6.916529
4	Köln-Nippes	31.8 km²	110.092	3462.0	50.958994	6.941777
2	Köln-Lindenthal	41.6 km²	137.552	3308.0	50.935935	6.871246

I then went on looking for rent pricing data, which I have scraped as following:

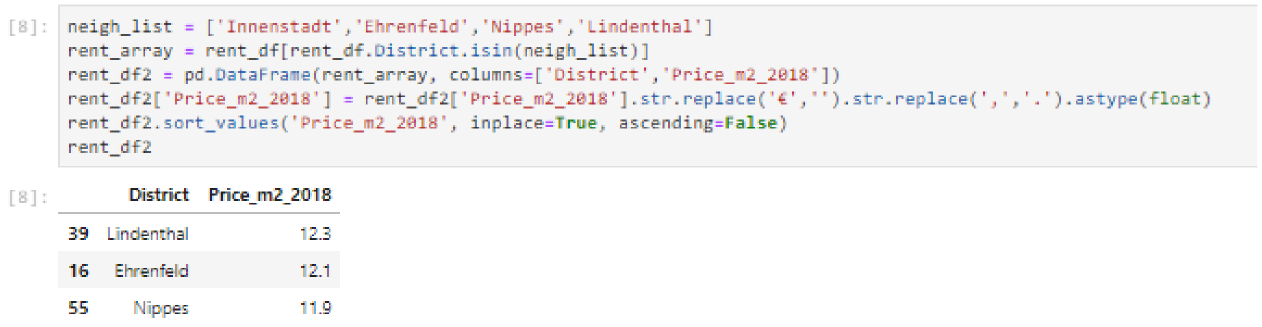


```
[7]: rent_df = pd.read_html('https://www.koeln.de/immobilien/mietspiegel.html')[0]
     rent_df.drop(['3. Quartal 2018','Veränderung'], axis=1, inplace=True)
     rent_df.rename(columns={'Stadtteil':'District','4. Quartal 2018/04':'Price_m2_2018'}, inplace=True)
     rent_df.head()
```

[7]:

	District	Price_m2_2018
0	Altstadt-Nord	10,70 €
1	Altstadt-Süd	12,20 €
2	Auweiler	8,60 €
3	Bayenthal	11,90 €
4	Bickendorf	10,40 €

After that, I have looked up the neighborhoods of interest in the resulting data frame:

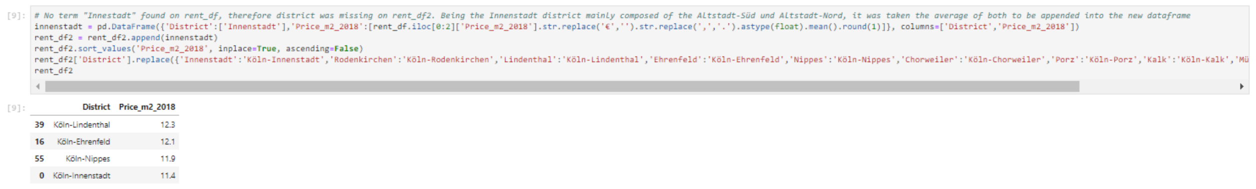


```
[8]: neigh_list = ['Innenstadt','Ehrenfeld','Nippes','Lindenthal']
     rent_array = rent_df[rent_df.District.isin(neigh_list)]
     rent_df2 = pd.DataFrame(rent_array, columns=['District','Price_m2_2018'])
     rent_df2['Price_m2_2018'] = rent_df2['Price_m2_2018'].str.replace('€','').str.replace(',','.').astype(float)
     rent_df2.sort_values('Price_m2_2018', inplace=True, ascending=False)
     rent_df2
```

[8]:

	District	Price_m2_2018
39	Lindenthal	12.3
16	Ehrenfeld	12.1
55	Nippes	11.9

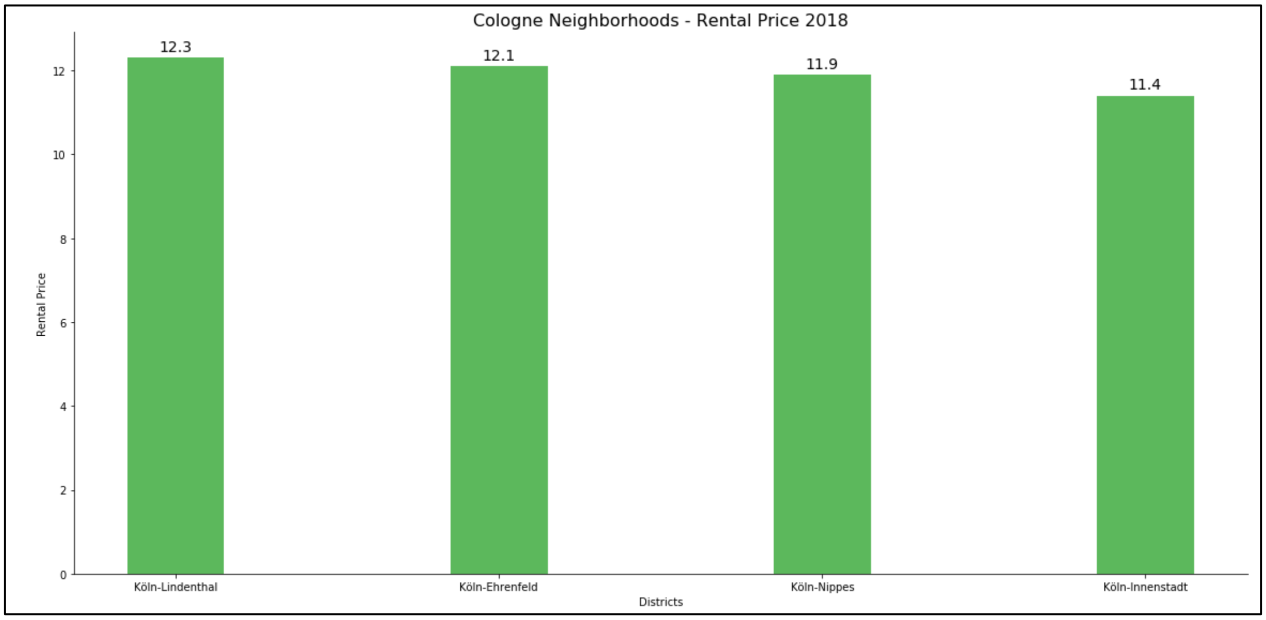
As you can see, Köln-Innenstadt was not directly found through the listing scraped from the website, so I processed the data further taking the average of Altstadt-Nord and -Süd and used it as the data point for Innenstadt:



```
[9]: # No term "Innestadt" found on rent_df, therefore district was missing on rent_df2. Being the Innenstadt district mainly composed of the Altstadt-Süd und Altstadt-Nord, it was taken the average of both to be appended into the new dataframe
     innenstadt = pd.DataFrame({'District':['Innenstadt'],'Price_m2_2018':[rent_df.iloc[0:2]['Price_m2_2018'].str.replace('€','').str.replace(',','.').astype(float).mean().round(1)]}, columns=['District','Price_m2_2018'])
     rent_df2 = rent_df2.append(innenstadt)
     rent_df2.sort_values('Price_m2_2018', inplace=True, ascending=False)
     rent_df2['District'].replace({'Innenstadt':'Köln-Innenstadt','Rodenkirchen':'Köln-Rodenkirchen','Lindenthal':'Köln-Lindenthal','Ehrenfeld':'Köln-Ehrenfeld','Nippes':'Köln-Nippes','Chorweiler':'Köln-Chorweiler','Porz':'Köln-Porz','Kalk':'Köln-Kalk','Mü
     rent_df2
```

[9]:

	District	Price_m2_2018
39	Köln-Lindenthal	12.3
16	Köln-Ehrenfeld	12.1
55	Köln-Nippes	11.9
0	Köln-Innenstadt	11.4



With the data available it was then possible to rule further the districts not falling into the cost criteria, leaving two neighborhoods to be further analyzed:

```
[85]: cgn_df.drop([cgn_df.index[1],cgn_df.index[3]], axis=0, inplace=True)
      cgn_df
```

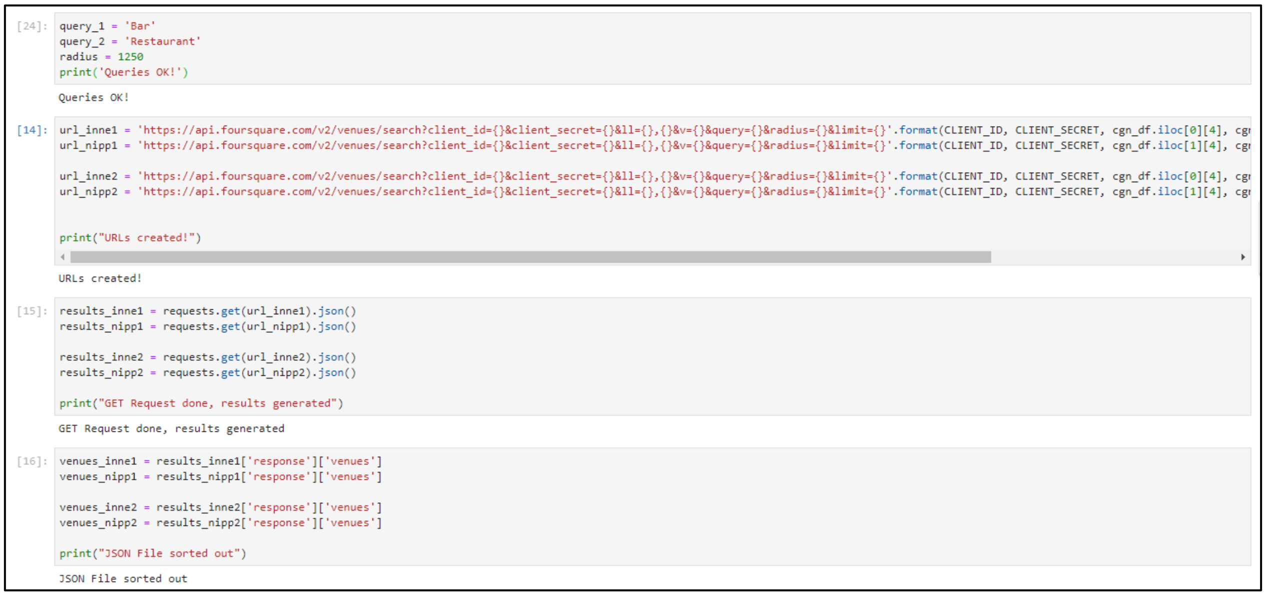
[85]:

	District	Area	Population	Pop_Density	Latitude	Longitude
0	Köln-Innenstadt	16.4 km²	127.033	7746.0	50.937328	6.959234
4	Köln-Nippes	31.8 km²	110.092	3462.0	50.958994	6.941777

3.2 Calling Venues data

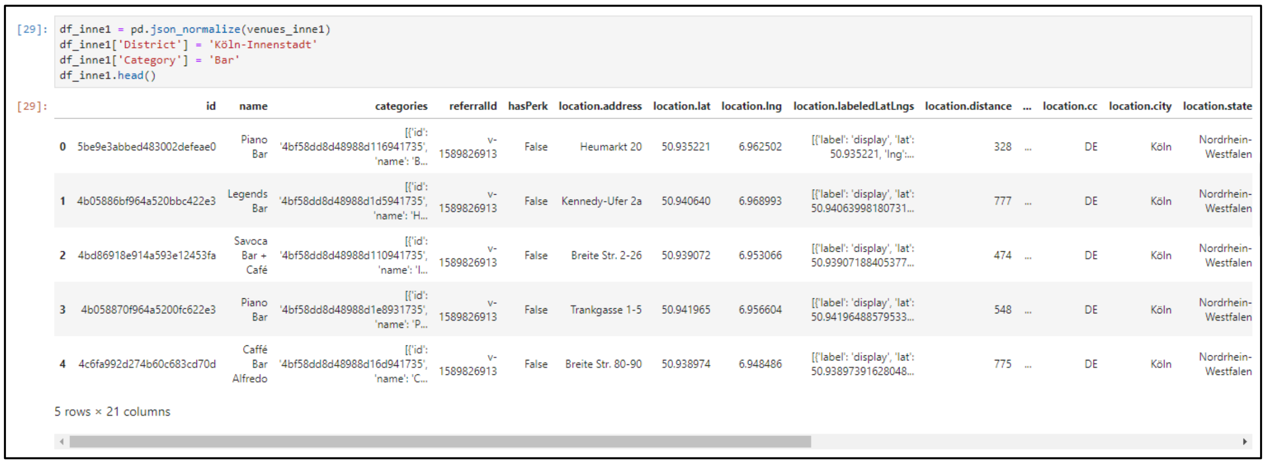
As mentioned before, as a requirement for this Capstone, we needed to call the venues data through Foursquare API. The process is straightforward, and it was made easier by the previous filtering done on the city data.

Here we need to start taking a look into insights on both bars and restaurants, so I've created a couple of variables to deal with each, query_1 and query_2 for the bar and restaurant queries, respectively and 4 for the URLs, 2 per district, one for each type of venue. The radius was established through a couple of iterations done during the data clustering. The method to do it was basically based on previous knowledge of the city, while checking on the map the most senseful way of distributing the data, bigger radius would return inaccuracy on to which neighborhood the venue would belong to, so I've used different figures until the boundaries were respected accordingly:



```
[24]: query_1 = 'Bar'
      query_2 = 'Restaurant'
      radius = 1250
      print('Queries OK!')

      Queries OK!

[14]: url_inne1 = 'https://api.foursquare.com/v2/venues/search?client_id={}&client_secret={}&ll={},{}&v={}&query={}&radius={}&limit={}'.format(CLIENT_ID, CLIENT_SECRET, cgn_df.iloc[0][4], cgr
      url_nipp1 = 'https://api.foursquare.com/v2/venues/search?client_id={}&client_secret={}&ll={},{}&v={}&query={}&radius={}&limit={}'.format(CLIENT_ID, CLIENT_SECRET, cgn_df.iloc[1][4], cgr

      url_inne2 = 'https://api.foursquare.com/v2/venues/search?client_id={}&client_secret={}&ll={},{}&v={}&query={}&radius={}&limit={}'.format(CLIENT_ID, CLIENT_SECRET, cgn_df.iloc[0][4], cgr
      url_nipp2 = 'https://api.foursquare.com/v2/venues/search?client_id={}&client_secret={}&ll={},{}&v={}&query={}&radius={}&limit={}'.format(CLIENT_ID, CLIENT_SECRET, cgn_df.iloc[1][4], cgr


      print("URLs created!")

      URLs created!

[15]: results_inne1 = requests.get(url_inne1).json()
      results_nipp1 = requests.get(url_nipp1).json()

      results_inne2 = requests.get(url_inne2).json()
      results_nipp2 = requests.get(url_nipp2).json()

      print("GET Request done, results generated")

      GET Request done, results generated

[16]: venues_inne1 = results_inne1['response']['venues']
      venues_nipp1 = results_nipp1['response']['venues']

      venues_inne2 = results_inne2['response']['venues']
      venues_nipp2 = results_nipp2['response']['venues']

      print("JSON File sorted out")

      JSON File sorted out
```

With those variables I was now able to create Data frames out of it, below one example of the coding used:



```
[29]: df_inne1 = pd.json_normalize(venues_inne1)
      df_inne1['District'] = 'Köln-Innenstadt'
      df_inne1['Category'] = 'Bar'
      df_inne1.head()
```

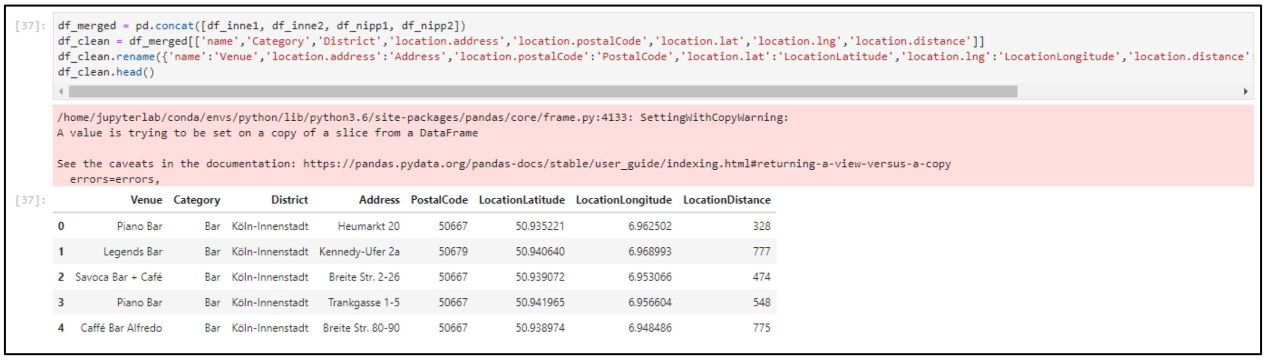
[29]:

	id	name	categories	referralId	hasPerk	location.address	location.lat	location.lng	location.labeledLatLngs	location.distance	...	location.cc	location.city	location.state
0	5be9e3abbed483002defeae0	Piano Bar	[{'id': '4bf58dd8d48988d116941735', 'name': 'B...	v-1589826913	False	Heumarkt 20	50.935221	6.962502	[{'label': 'display', 'lat': 50.935221, 'lng':...	328	...	DE	Köln	Nordrhein-Westfalen
1	4b05886bf964a520bbc422e3	Legends Bar	[{'id': '4bf58dd8d48988d1d5941735', 'name': 'H...	v-1589826913	False	Kennedy-Ufer 2a	50.940640	6.968993	[{'label': 'display', 'lat': 50.94063998180731...	777	...	DE	Köln	Nordrhein-Westfalen
2	4bd86918e914a593e12453fa	Savoca Bar + Café	[{'id': '4bf58dd8d48988d110941735', 'name': 'I...	v-1589826913	False	Breite Str. 2-26	50.939072	6.953066	[{'label': 'display', 'lat': 50.93907188405377...	474	...	DE	Köln	Nordrhein-Westfalen
3	4b058870f964a5200fc622e3	Piano Bar	[{'id': '4bf58dd8d48988d1e8931735', 'name': 'P...	v-1589826913	False	Trankgasse 1-5	50.941965	6.956604	[{'label': 'display', 'lat': 50.94196488579533...	548	...	DE	Köln	Nordrhein-Westfalen
4	4c6fa992d274b60c683cd70d	Caffé Bar Alfredo	[{'id': '4bf58dd8d48988d16d941735', 'name': 'C...	v-1589826913	False	Breite Str. 80-90	50.938974	6.948486	[{'label': 'display', 'lat': 50.93897391628048...	775	...	DE	Köln	Nordrhein-Westfalen

5 rows × 21 columns

3.3 Cleaning the Venues data

After I called the data out of Foursquare, I still needed to clean and prepare the data. The two main tasks were to drop everything that would not be relevant and to blend the different data-frames created, as they were at this point separated.

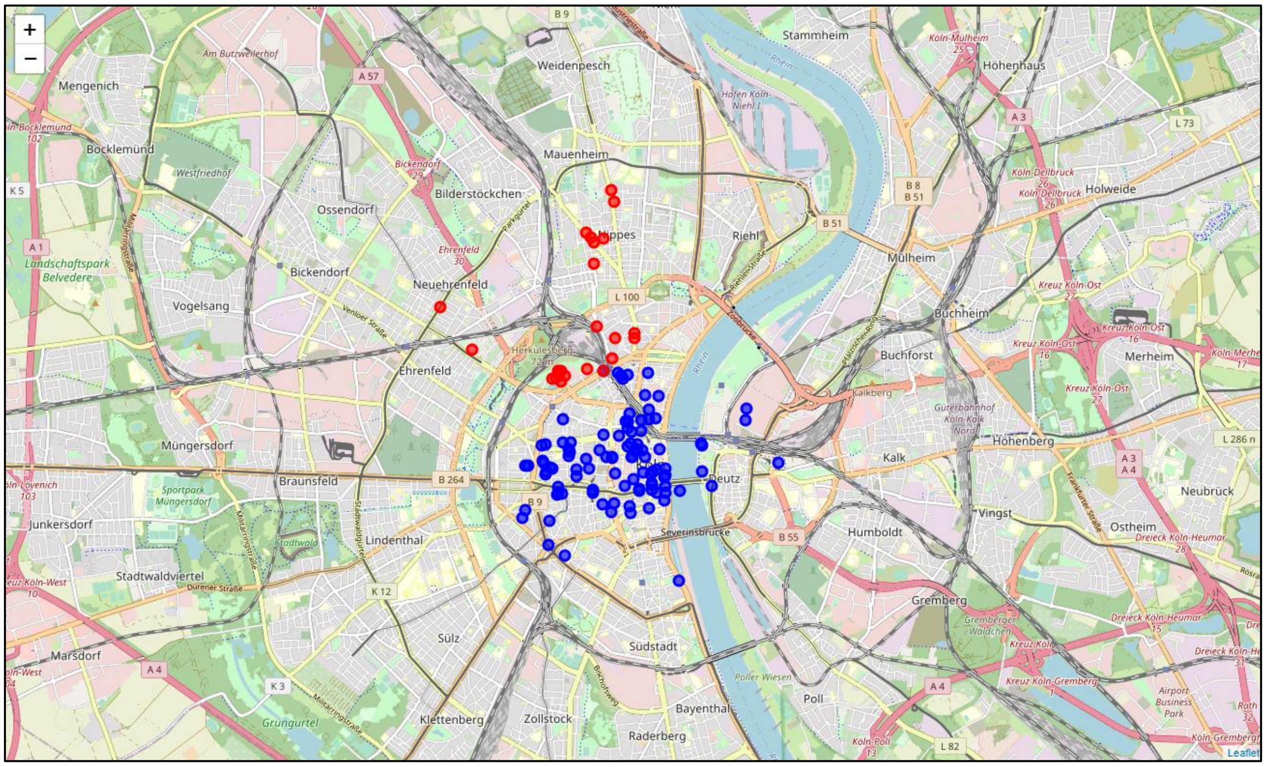


```
[37]: df_merged = pd.concat([df_inne1, df_inne2, df_nipp1, df_nipp2])
      df_clean = df_merged[['name','Category','District','location.address','location.postalCode','location.lat','location.lng','location.distance']]
      df_clean.rename({'name':'Venue','location.address':'Address','location.postalCode':'PostalCode','location.lat':'LocationLatitude','location.lng':'LocationLongitude','location.distance'
      df_clean.head()

      /home/jupyterlab/conda/envs/python/lib/python3.6/site-packages/pandas/core/frame.py:4133: SettingWithCopyWarning:
      A value is trying to be set on a copy of a slice from a DataFrame

      See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
        errors=errors,
```

[37]:

	Venue	Category	District	Address	PostalCode	LocationLatitude	LocationLongitude	LocationDistance
0	Piano Bar	Bar	Köln-Innenstadt	Heumarkt 20	50667	50.935221	6.962502	328
1	Legends Bar	Bar	Köln-Innenstadt	Kennedy-Ufer 2a	50679	50.940640	6.968993	777
2	Savoca Bar + Café	Bar	Köln-Innenstadt	Breite Str. 2-26	50667	50.939072	6.953066	474
3	Piano Bar	Bar	Köln-Innenstadt	Trankgasse 1-5	50667	50.941965	6.956604	548
4	Caffé Bar Alfredo	Bar	Köln-Innenstadt	Breite Str. 80-90	50667	50.938974	6.948486	775

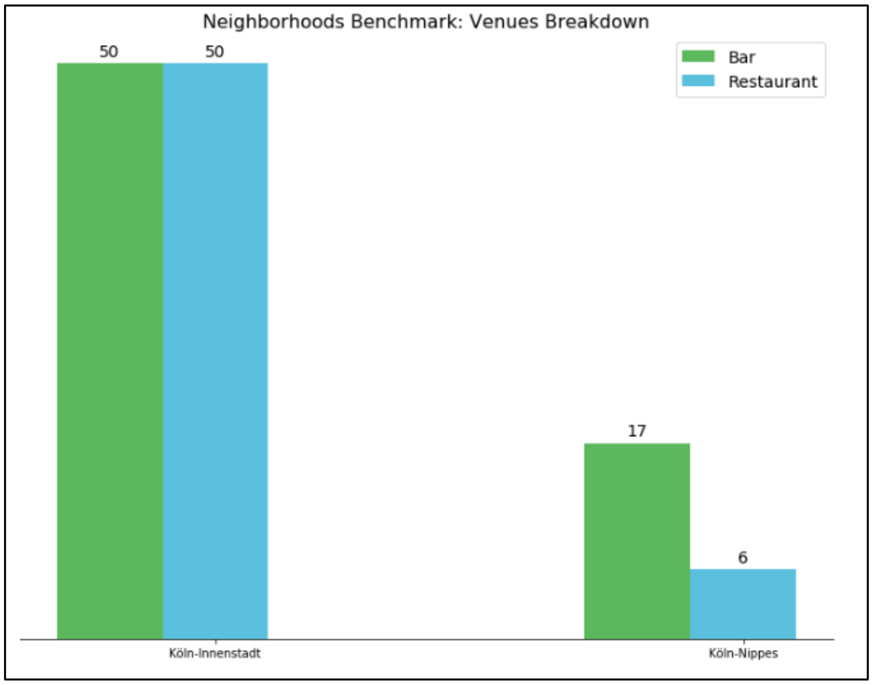
Below the distribution of the venue locations on the city map, being Köln-Nippes represented in red and Köln-Innenstadt in blue:



One thing that calls attention is that there is a clear difference on the amount of venues between the two neighborhoods, which is in fact related to the next topic to be investigated.

3.4 Competition benchmark

One key criterion established in order to choose which kind of venue and neighborhood would be chosen was competition. Let us look at the Neighborhoods number's breakdown:



The chart shows that when it comes to quantity of venues, for both restaurants and bars, Köln-Innenstadt has substantially more competition than in Nippes.

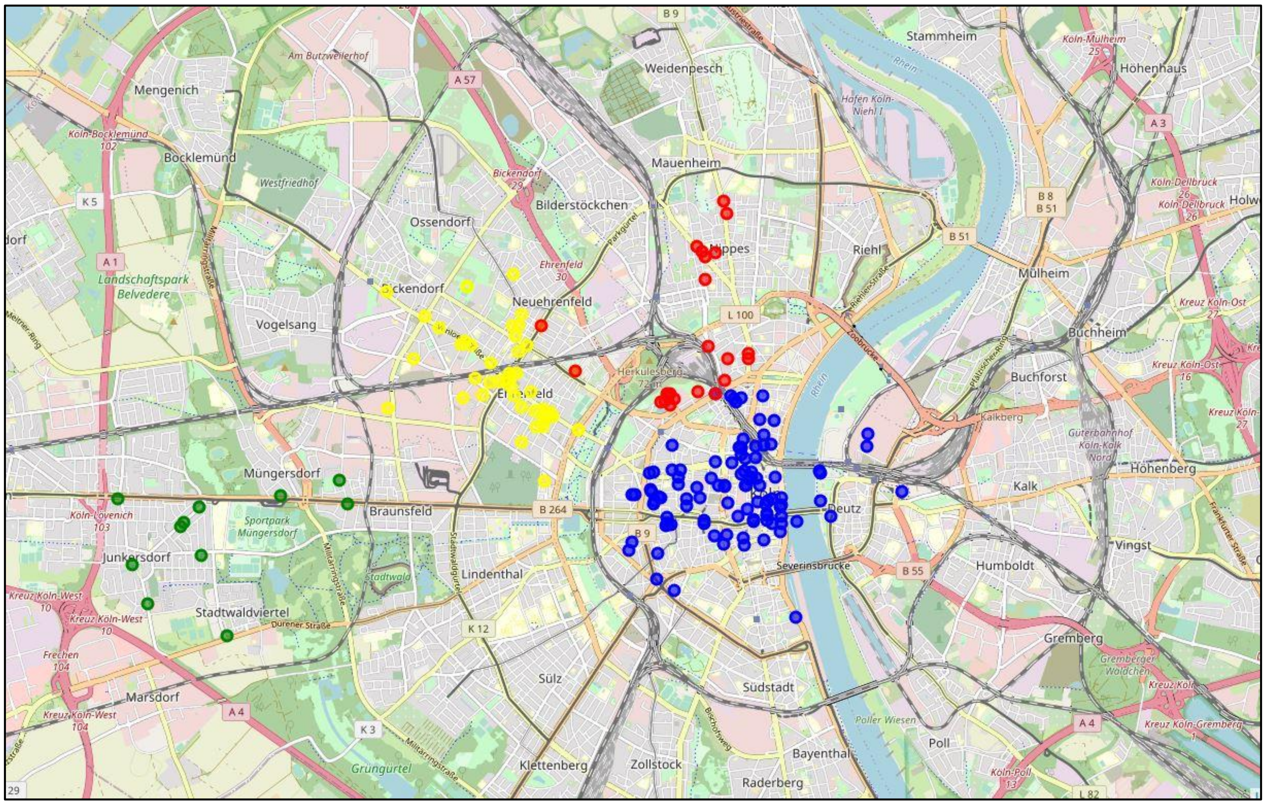
4. Results

According to the competition target defined in the beginning of the analysis, there should be no more than 20 venues falling into the same venue category in the chosen neighborhood, which necessarily means that between the two districts left, Köln-Nippes have more appeal when considering the level of competition.

There is still one decision to be taken: what would make more sense to be opened, a bar or restaurant? When looking at the data, there are 17 bars located in Nippes according to the Foursquare data, and 6 restaurants. In order to give perspective, a couple of considerations are due. First, about bars, the kind of entertainment is more uniform among themselves, unless of course the bar is thematic, is a franchise or is already well known and marketed. Restaurants, on the other hand, might not compete as strongly with each other due to the whole variety of typical food available to be chosen when considering such establishment, especially in such cosmopolitan town as Cologne. This may incur in an exception depending on several different factors that are not covered by this study, such as opening hours, pricing and reviews available on the internet. Under these assumptions, **the recommendation proposed by the study is to open a restaurant in the neighborhood Köln-Nippes.**

5. Discussion

A couple of further considerations need to be mentioned. The first is that, although the data shows a clear path with a well-defined rationale in order to reach the conclusion mentioned above, several factors were not part of the criteria used in the analysis. This means that the business case could be slightly different depending on what kind of other targets were to be established. As an example, one key factor to make the decision to lean towards Nippes instead of Innenstadt was the competition targets. What would happen if we also consider internet reviews benchmark and more detailed rent data for every neighborhood? What about if the competition target was substantially higher than 20? Another aspect is the distance to the city center, which for our use case did not need to be checked deeper, as the neighborhoods that were more distant from downtown were ruled out early in the analysis. But what would happen if the average income were to be considered? Köln-Lindenthal is known for being a higher end neighborhood, which means more expensive (this makes sense when we look at the rent data scraped during the study, this neighborhood was the most expensive). On the other hand, not being that close to the city center could be an asset, depending on a couple of non-tangible aspects that also were not part of this study. Below a visual reference of the venues of 4 of the main neighborhoods distributed in Cologne's map:



By considering more of these non-tangible aspects, not just Lindenthal (in green on the map) but also Ehrenfeld (in yellow) could be more attractive according to certain specific contexts. The bottom line here is that, although the study is completely data-driven, it serves as a good initial reference of what are key important aspects for similar business cases only, so in no way it can propose itself to be an ultimate reference for decision making when considering other kind of criteria. Here it is important to say that, for the purposes of avoiding scope creep and to keep the Capstone Project comprehensiveness as concise as possible, the number of variables and criteria were kept to a minimum.

6. Conclusion

In this analysis, I proposed a discussion, based on objective targets, on which neighborhood of the city of Cologne would have more appeal to open either a bar or a restaurant. After conducting this study, we can infer that there are substantial differences between different parts of the city of Cologne whenever considering starting a business in the entertainment and gastronomy industries. Through several different data analysis techniques and well-defined criteria, we were able to see in a step-by-step approach how to apply Data Science principles in order to take an informed decision. Being myself a citizen of Cologne and living in the city for a couple of years, conducting this study not just meant a lot of learning, but also brought joy and satisfaction not just because I could see several points of convergence in the reached conclusions with the daily routine, but also because of the opportunity to getting to know a bit more of the town I live in. If you are a fellow Data Science student going through the Capstone project and using this material as reference for your own analysis, I hope you find it useful and that you, the same way as me, can see the value in going through the whole process, and can have as much fun as I did by doing this work.