IBM Data Science Capstone Project

(Final Report , by Uygar Hizal)

1) Introduction to the Business Problem:

Hello and welcome to my project report.

In this project I will try to solve an imaginary business problem, to present as many skills as possible from the IBM Data Science Capstone multi Course Program in Coursera.

One Foreign Investor (customer) wants to invest and open a Clothing Store Business in one of the Germany's big cities. He has a concept in his mind, but as being foreigner, he has not much idea about Germany's City structure and therefore needs help.

He owns already a middle range Clothing Store Chain in USA. And this will be the first Store opened in Germany, therefore, it should meet some criteria to present his brand correctly.

After a meeting with him, he defined his business aim and informed me about the criteria's like following, it should be;

1. Opened in one of the big Cities in Germany (Population over 100.000 and more).
2. Within the max. 15 minutes walking distance from the Geographical coordinates of the City Center
3. As far away from other Clothing Stores as possible
4. As close as possible to Italian Restaurants, because his collections are mostly Italian designs and he thinks, the customers visiting the Italian restaurants can be more interested in store windows as walking
5. As close as possible to Hotels, because guests of the in-city hotels are generally tended to buy clothes nearby.
6. After all He stated honestly that, He needs a city that he can pay less possible salaries as he aims to give 20 employee job.
7. He added also, a city with highest possible unemployment rate would be an advantage for him, as finding personal in a short time. Otherwise he could wait longer to complete all employee team.
8. Population of the city also counts as a positive measure too. (City should be as crowded as possible)

In this point it is very important to interpret investor's desires and convert these sentences to the scientific statements. For example, He saying "…max. 15 minutes walking distance." means for a Data Scientist: with an average walking speed of 5 km/h pedestrian, 1250 meters from the Geocoordinates of that city center. And it will be used in Foursquare Api call as Radius measure (R=**1250**). i.e.:

```
def getNearbyVenues(names, latitudes, longitudes, radius=1250):
```

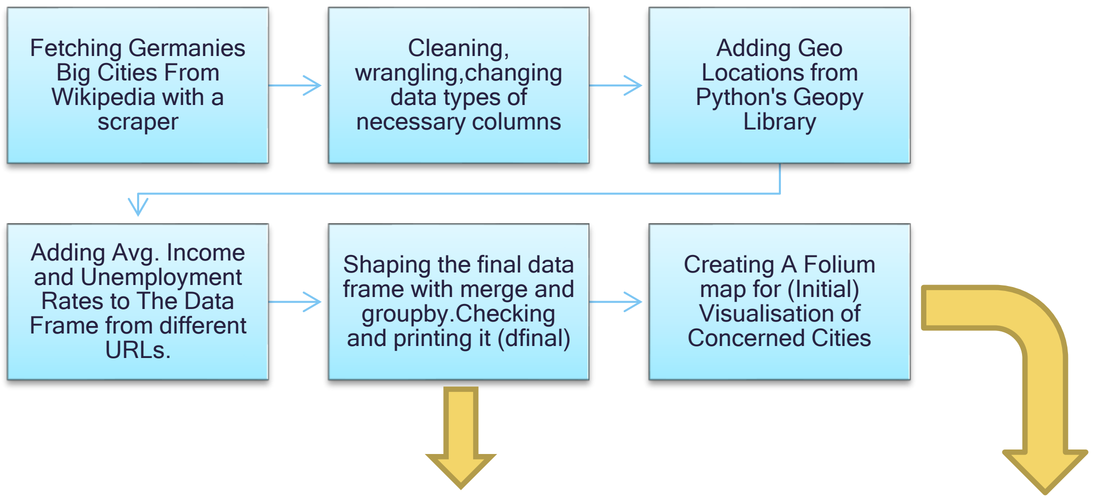
2) Necessary Data and its usage in this case:

As you may see from the business problem part above, I decided to add some more complexity to our standard course problem otherwise it could be solved only with foursquare general venue data.

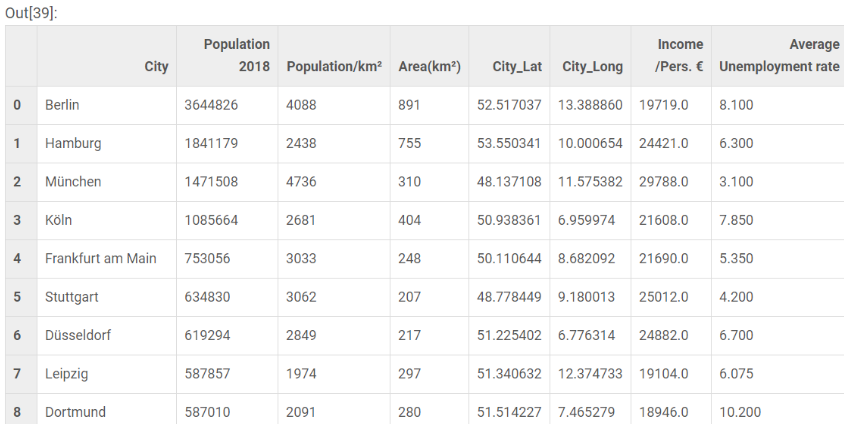
But in this case in addition to general Venues data of all Major German Cities, we will be adding some **Socioeconomic** information like **Population**, **Average Income /person**, **Average Unemployment Rate** and **Area** in km^2 of that city.

DATA :

Part 2.1: Socioeconomic Data



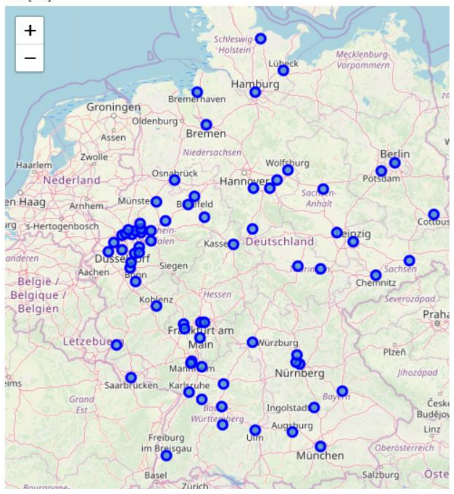
Fetching Germanies Big Cities From Wikipedia with a scraper
Cleaning, wrangling,changing data types of necessary columns
Adding Geo Locations from Python's Geopy Library
Adding Avg. Income and Unemployment Rates to The Data Frame from different URLs.
Shaping the final data frame with merge and groupby.Checking and printing it (dfinal)
Creating A Folium map for (Initial) Visualisation of Concerned Cities



Out[39]:

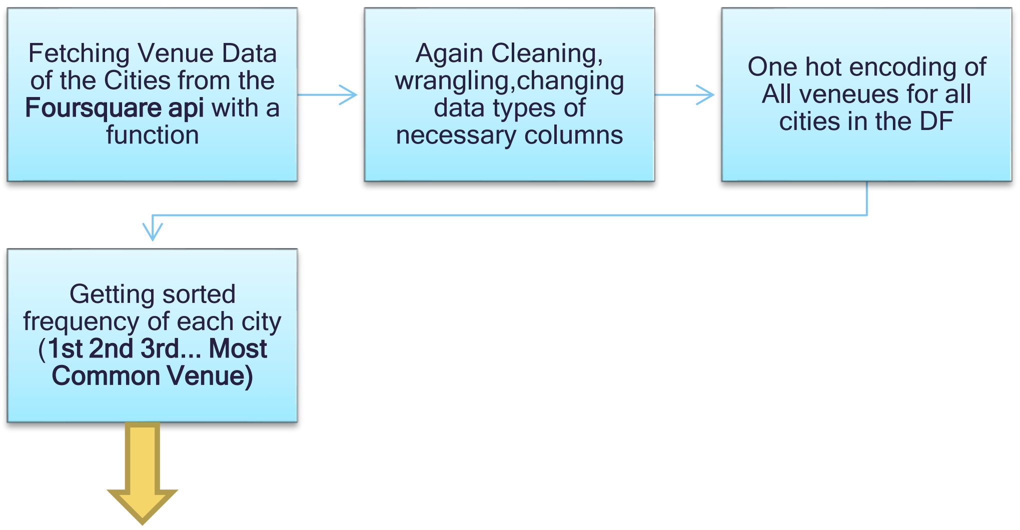
	City	Population 2018	Population/km²	Area(km²)	City_Lat	City_Long	Income /Pers. €	Average Unemployment rate
0	Berlin	3644826	4088	891	52.517037	13.388860	19719.0	8.100
1	Hamburg	1841179	2438	755	53.550341	10.000654	24421.0	6.300
2	München	1471508	4736	310	48.137108	11.575382	29788.0	3.100
3	Köln	1085664	2681	404	50.938361	6.959974	21608.0	7.850
4	Frankfurt am Main	753056	3033	248	50.110644	8.682092	21690.0	5.350
5	Stuttgart	634830	3062	207	48.778449	9.180013	25012.0	4.200
6	Düsseldorf	619294	2849	217	51.225402	6.776314	24882.0	6.700
7	Leipzig	587857	1974	297	51.340632	12.374733	19104.0	6.075
8	Dortmund	587010	2091	280	51.514227	7.465279	18946.0	10.200

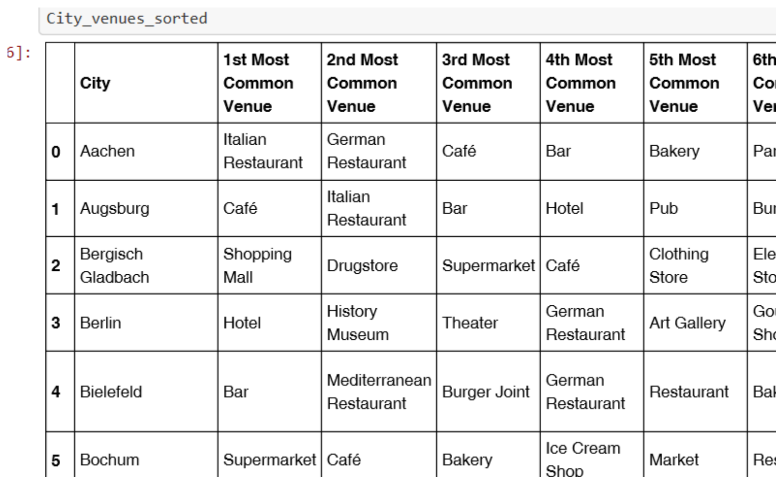
Out[41]:



Part 2.2: Cities Venue Data (A general Data, nothing to Customer yet!)

After obtaining and cleaning Socioeconomic data in part 2.1, now it is time to get all the venues for concerned cities from the foursquare api calls.


Fetching Venue Data of the Cities from the Foursquare api with a function
Again Cleaning, wrangling,changing data types of necessary columns
One hot encoding of All veneues for all cities in the DF
Getting sorted frequency of each city (1st 2nd 3rd... Most Common Venue)



```
City_venues_sorted
```

6]:

	City	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Co Ve
0	Aachen	Italian Restaurant	German Restaurant	Café	Bar	Bakery	Pa
1	Augsburg	Café	Italian Restaurant	Bar	Hotel	Pub	Bu
2	Bergisch Gladbach	Shopping Mall	Drugstore	Supermarket	Café	Clothing Store	Ele Sto
3	Berlin	Hotel	History Museum	Theater	German Restaurant	Art Gallery	Go Sh
4	Bielefeld	Bar	Mediterranean Restaurant	Burger Joint	German Restaurant	Restaurant	Bal
5	Bochum	Supermarket	Café	Bakery	Ice Cream Shop	Market	Re

List of Data Sources:

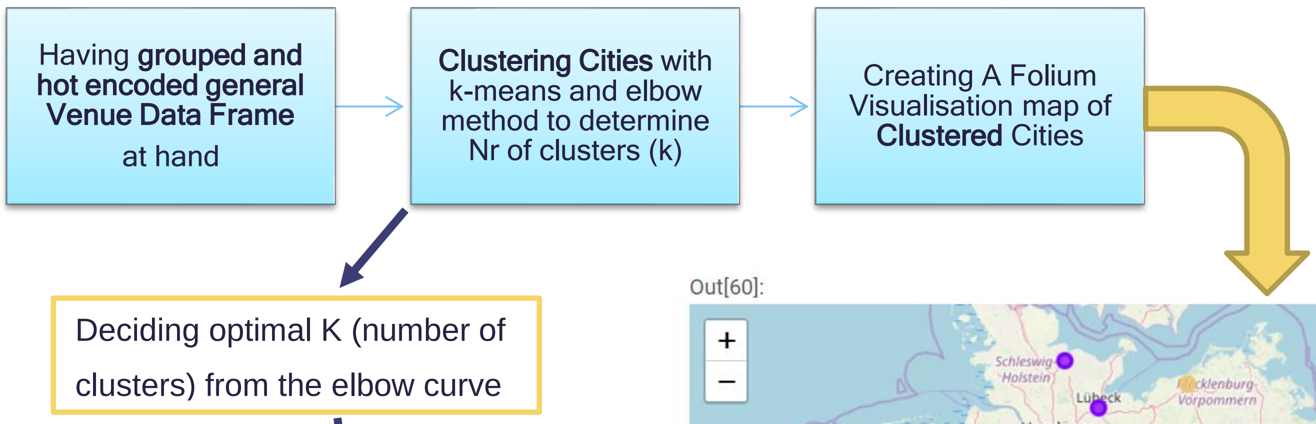
- Foursquare API data based on free API calls
- Wikipedia site for Major Cities List in Germany
- GeoPy library to fetch coordinates of the Major cities https://geopy.readthedocs.io/en/stable/
- Federal Statistical Office of Germany – (Statistisches Bundesamt) https://www.destatis.de/

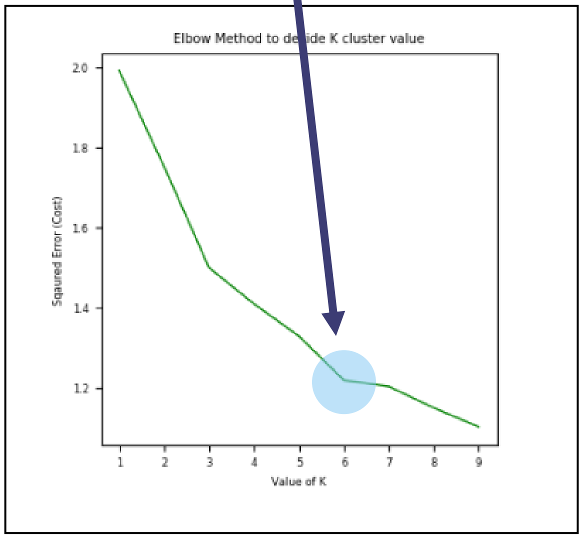
3) Methodology

After the preparation of both **Socioeconomic** and **general purpose (Hot encoded) Venue** information **into the two different data frames**, we still should combine / interpret these two data frames for a logical solution. This is where the investor (customer) comes into play.

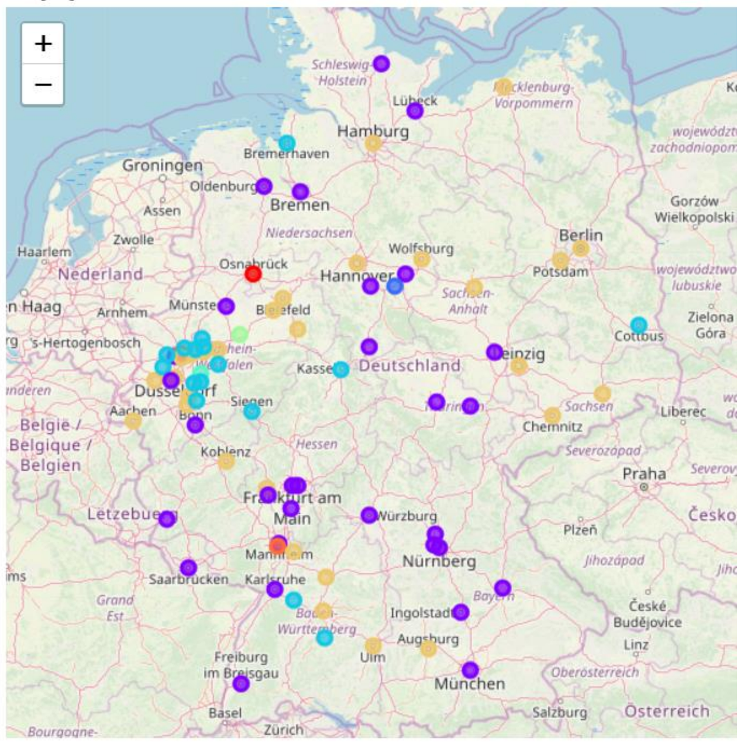
3.1) General Purpose Venue Data Frame Clusterization (as usual in lectures example):

Before starting the Customer Requirements Section, we will still cluster the General-Purpose Venue Data Frame (obtained at in part 2.2), In our special problem it is not necessary to cluster this data frame but, in order to prevent confusion among students, I did it too:


Having grouped and hot encoded general Venue Data Frame at hand
Clustering Cities with k-means and elbow method to determine Nr of clusters (k)
Creating A Folium Visualisation map of Clustered Cities
Deciding optimal K (number of clusters) from the elbow curve



Out[60]:



3.2) Analyzing the Investor Requirements (where Solving the Business Problem Begins):

Beginning from this section, we will only deal with the features (variables) those only interested the customer.
This means: presenting to the investor a Clusterization analyze / map, with Greek Restaurants / Sushi Restaurants / Supermarkets / Gas Stations / Lakes... are in it , is irrelevant to his business problem, and even such an approach guides us to wrong results .

In order to understand investor's requirements, we will introduce a new concept called **Customer Wishes_Matrix**. So what is the Wishes_Matrix? :

It is a features weight matrix (array) to express / to get investor desires in a scientific way..
Every feature that investor prerequisites from us, will be weighted to a scale from 1 to 10 in a manner of importance:

1. Opened in one of the big Cities in Germany (Population over 100.000 and more)

   This is already satisfied because we have fetched cities only with population >100k

2. Within the max. 15 minutes walking distance from the Geographical coordinates of the City Center

   This request is converted to a variable to use in Foursquare Api call (Radius = 1250 meters in search)

3. As far away from other Clothing Stores as possible

   This request is very important for him and weighted as 9 points over 10 points. **(0,9)**

4. As close as possible to Italian Restaurants….

   This request is somehow second degree and weighted as 5 points over 10 points. **(0,5)**

5. As close as possible to Hotels ….

   This request is second degree in importance but still weighted as 7 point over 10 points. **(0,7)**

6. Cities that statistically less possible salaries are paid ….

   For investors salary issues are always important (☹), So it is weighted as 8 point. **(0,8)**

7. Cities with highest possible unemployment rate …

   This request is also important but not more than salaries. So, gets 7 points over 10 pts. **(0,7)**

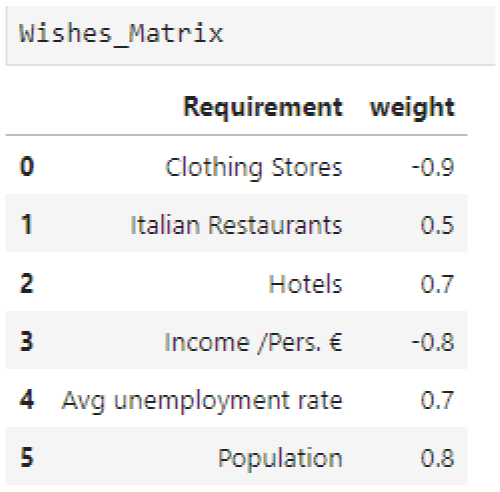
8. Population of the city also counts as a positive measure too. (City should be as crowded as possible)

   Population value is directly related to the number of potential customers therefore it will be weighted as 8 points over 10. **(0,8)**

3.2.a) Creating A Weight ("Wishes Matrix"):

$$\begin{array}{cc} \underline{\text{Re}} & \underline{\text{W}} \\ \multicolumn{2}{c}{\left\{\begin{matrix} Request\ 3 & 0{,}9 \\ Request\ 4 & 0{,}5 \\ Request\ 5 & 0{,}7 \\ Request\ 6 & 0{,}8 \\ Request\ 7 & 0{,}7 \\ Request\ 8 & 0{,}8 \end{matrix}\right\}} \end{array} \text{ x } \begin{array}{c} \underline{\text{C1}} \\ \left\{\begin{matrix} -1 \\ 1 \\ 1 \\ -1 \\ 1 \\ 1 \end{matrix}\right\} \end{array} = \text{Wishes Matrix}$$

Correlation bit: (-1 for negatively correlated variables and +1 is for positive correlation) For example increase on value of the Request 3 variable (Amount of Clothing stores in the area) will be added as punishment to end Data Frame, as it is not desired. But increase on population is positively counts (+) to end Data frame, as it is an advantage)



```
Wishes_Matrix
```

	Requirement	weight
0	Clothing Stores	-0.9
1	Italian Restaurants	0.5
2	Hotels	0.7
3	Income /Pers. €	-0.8
4	Avg unemployment rate	0.7
5	Population	0.8

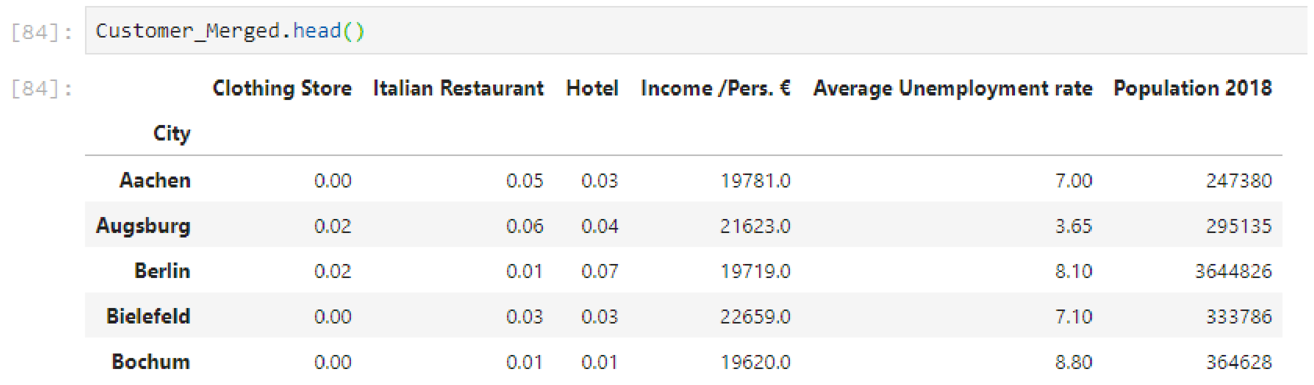
3.2.b) Preparing an Investor (customer) point of interest Data Frame:

In this section we made following steps:

Step 1) Getting only the necessary and related 'Venue' columns (features) to a new data frame called Custumer_Venues

Step 2) Again getting only the necessary 'Social Data' columns (features) to a new data frame called Customer_Social

Step 3) Merging these two newly created Data Frames based on Key 'City'. Calling New DF as Customer_Merged



```
[84]: Customer_Merged.head()
```

[84]:

City	Clothing Store	Italian Restaurant	Hotel	Income /Pers. €	Average Unemployment rate	Population 2018
Aachen	0.00	0.05	0.03	19781.0	7.00	247380
Augsburg	0.02	0.06	0.04	21623.0	3.65	295135
Berlin	0.02	0.01	0.07	19719.0	8.10	3644826
Bielefeld	0.00	0.03	0.03	22659.0	7.10	333786
Bochum	0.00	0.01	0.01	19620.0	8.80	364628

3.2.c) Applying a Normalization to the Customer_Merged Data Frame:

<u>**Methodology and importance of the normalization**</u>

Let's discuss why the normalization is very important:

In short: **Normalization will bring our interested features to a comparable grade.**

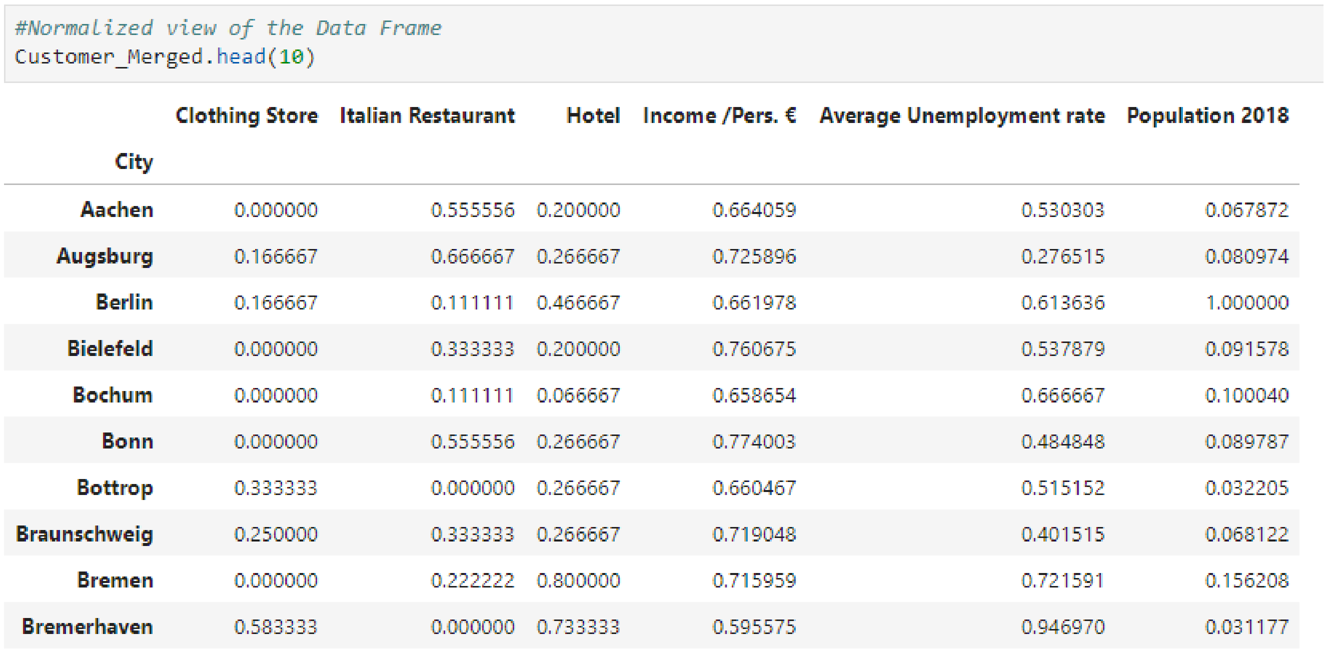
How? For example: population of Berlin is 3644826 but Frequency of the 'clothing stores' is 0.02. If we want a total score to evaluate the total DF fairly, scales of the features should be the same.

otherwise population would be rendered as billions of times more important than the 'clothing stores'. But in reality, it is not like that, we don’t want this!

Understanding this fact, although there are many normalization techniques, we will just encode all variables in our 6 Features (columns) to the numeric values between 0 and 1.

As an example, the highest Population value will be converted to 1.0, and at the same time highest frequency of the Hotel also will be converted to 1.

This will ensure that each feature (each column) have the same WEIGHT (same importance) as the others.



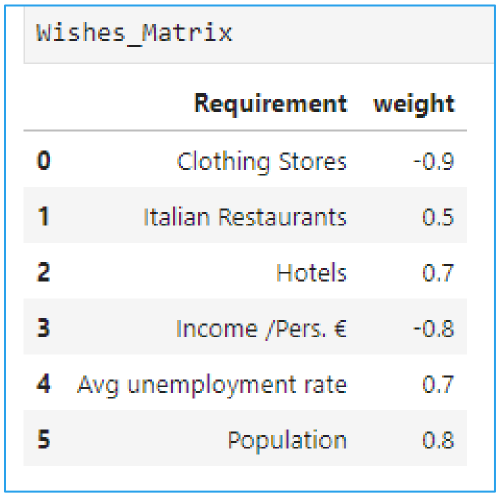
```
#Normalized view of the Data Frame
Customer_Merged.head(10)
```

City	Clothing Store	Italian Restaurant	Hotel	Income /Pers. €	Average Unemployment rate	Population 2018
Aachen	0.000000	0.555556	0.200000	0.664059	0.530303	0.067872
Augsburg	0.166667	0.666667	0.266667	0.725896	0.276515	0.080974
Berlin	0.166667	0.111111	0.466667	0.661978	0.613636	1.000000
Bielefeld	0.000000	0.333333	0.200000	0.760675	0.537879	0.091578
Bochum	0.000000	0.111111	0.066667	0.658654	0.666667	0.100040
Bonn	0.000000	0.555556	0.266667	0.774003	0.484848	0.089787
Bottrop	0.333333	0.000000	0.266667	0.660467	0.515152	0.032205
Braunschweig	0.250000	0.333333	0.266667	0.719048	0.401515	0.068122
Bremen	0.000000	0.222222	0.800000	0.715959	0.721591	0.156208
Bremerhaven	0.583333	0.000000	0.733333	0.595575	0.946970	0.031177

After the normalization, we get a neutral Data, in this way 'First of all' no column is more or less important than the others, until we apply the investor (customer) wishes.
Only after this normalization, we can apply the Wishes_Matrix to this neutral Data Frame, to get customers end results.

3.2.d) Multiplying the Wishes_Matrix (already having the same features) with the Customer_Merged Data Frame:

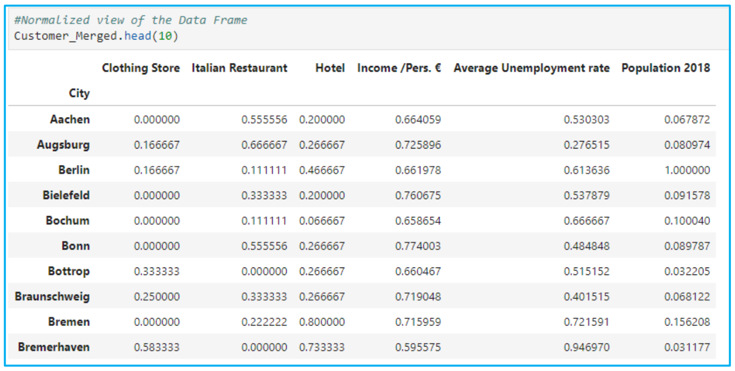
In this section we have assigned the weight column of the Wishes_Matrix to a NumPy array object. And applying an arithmetic multiplication of 6 features to the Customers normalized data frame (keeping the features order same as the wishes matrix).



```
Wishes_Matrix
```

	Requirement	weight
0	Clothing Stores	-0.9
1	Italian Restaurants	0.5
2	Hotels	0.7
3	Income /Pers. €	-0.8
4	Avg unemployment rate	0.7
5	Population	0.8

X

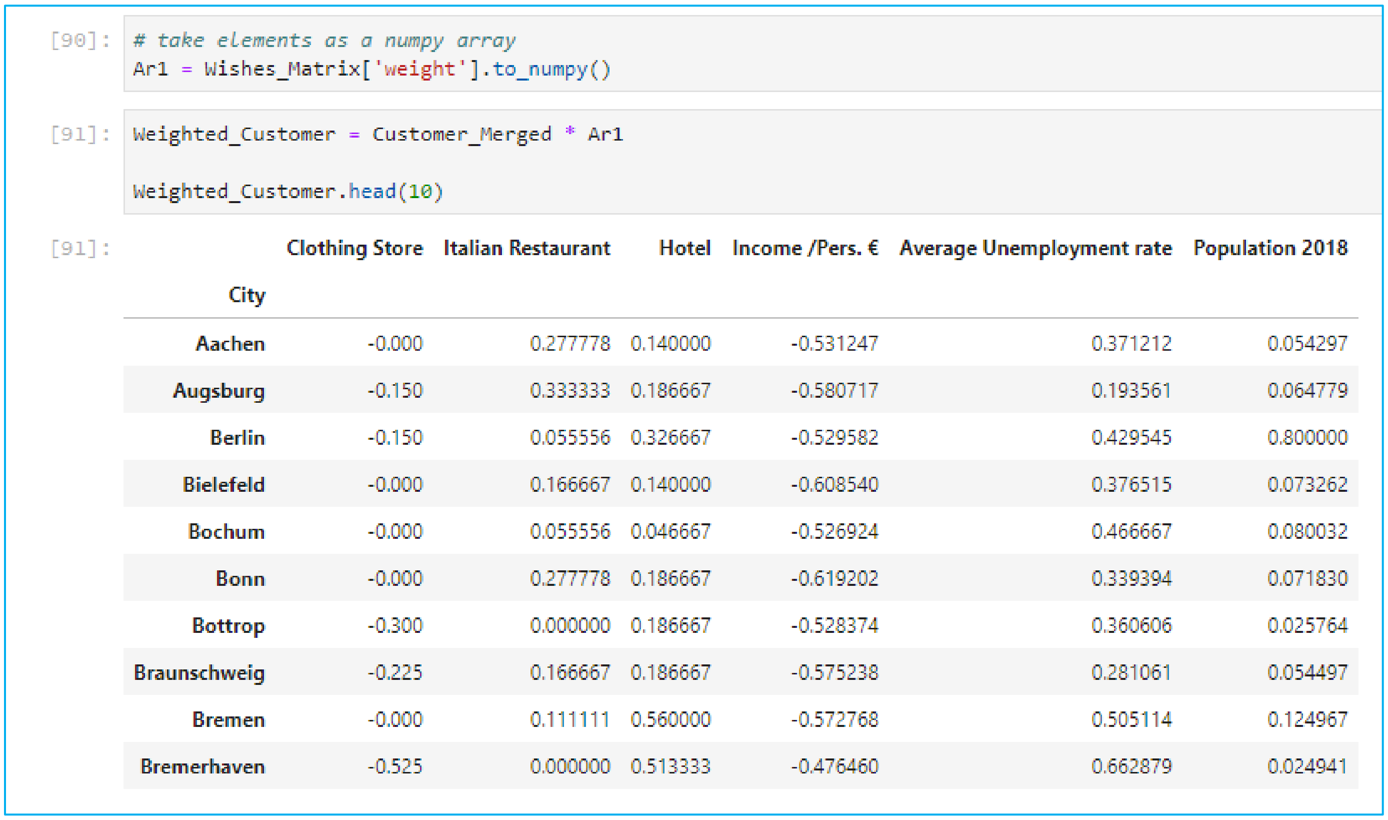


```
#Normalized view of the Data Frame
Customer_Merged.head(10)
```

	Clothing Store	Italian Restaurant	Hotel	Income /Pers. €	Average Unemployment rate	Population 2018
City						
Aachen	0.000000	0.555556	0.200000	0.664059	0.530303	0.067872
Augsburg	0.166667	0.666667	0.266667	0.725896	0.276515	0.080974
Berlin	0.166667	0.111111	0.466667	0.661978	0.613636	1.000000
Bielefeld	0.000000	0.333333	0.200000	0.760675	0.537879	0.091578
Bochum	0.000000	0.111111	0.066667	0.658654	0.666667	0.100040
Bonn	0.000000	0.555556	0.266667	0.774003	0.484848	0.089787
Bottrop	0.333333	0.000000	0.266667	0.660467	0.515152	0.032205
Braunschweig	0.250000	0.333333	0.266667	0.719048	0.401515	0.068122
Bremen	0.000000	0.222222	0.800000	0.715959	0.721591	0.156208
Bremerhaven	0.583333	0.000000	0.733333	0.595575	0.946970	0.031177

=

Weighted_Customer Data Frame (Normalized)



```
[90]: # take elements as a numpy array
      Ar1 = Wishes_Matrix['weight'].to_numpy()
```

```
[91]: Weighted_Customer = Customer_Merged * Ar1

      Weighted_Customer.head(10)
```

[91]:

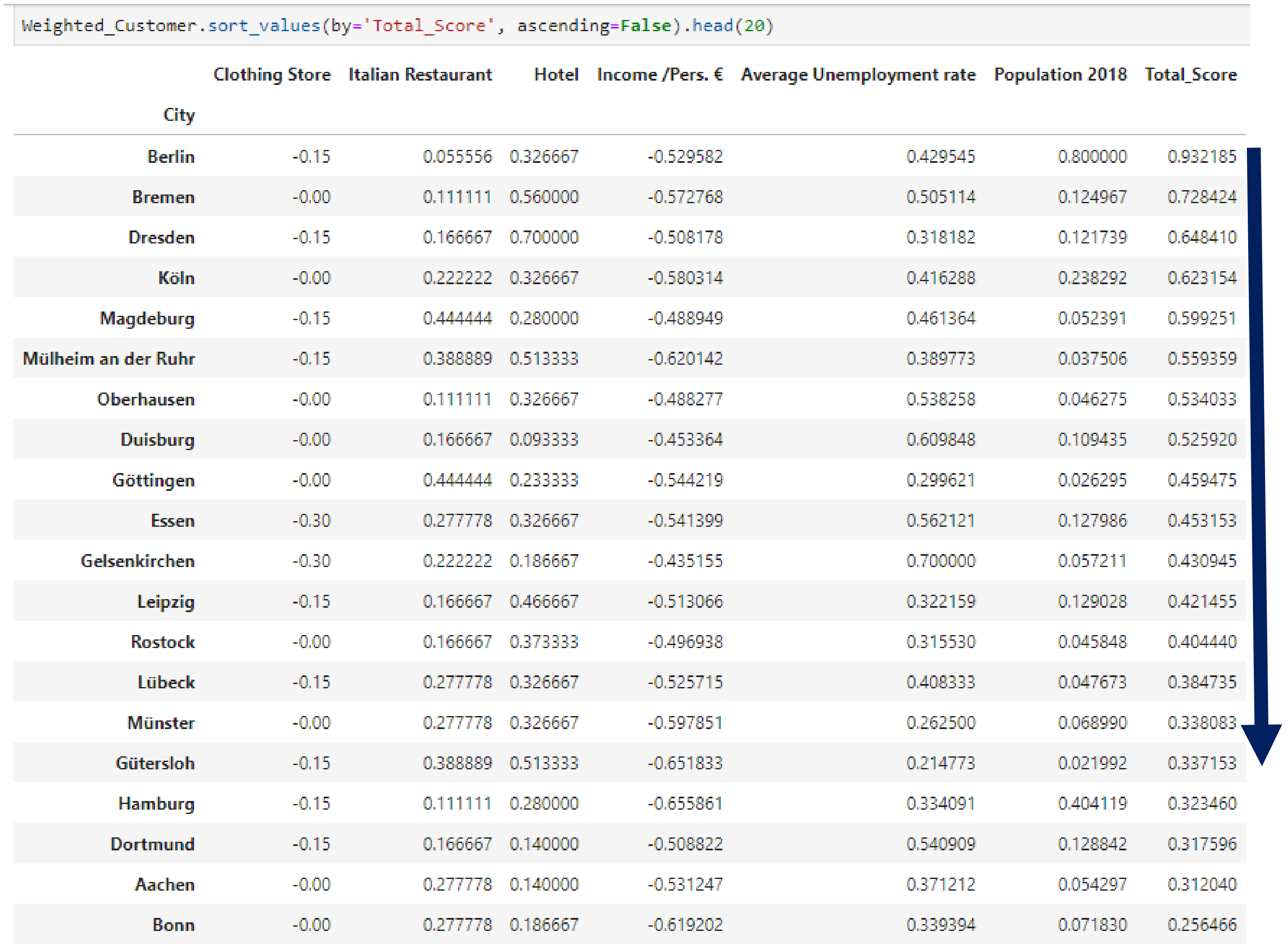
	Clothing Store	Italian Restaurant	Hotel	Income /Pers. €	Average Unemployment rate	Population 2018
City						
Aachen	-0.000	0.277778	0.140000	-0.531247	0.371212	0.054297
Augsburg	-0.150	0.333333	0.186667	-0.580717	0.193561	0.064779
Berlin	-0.150	0.055556	0.326667	-0.529582	0.429545	0.800000
Bielefeld	-0.000	0.166667	0.140000	-0.608540	0.376515	0.073262
Bochum	-0.000	0.055556	0.046667	-0.526924	0.466667	0.080032
Bonn	-0.000	0.277778	0.186667	-0.619202	0.339394	0.071830
Bottrop	-0.300	0.000000	0.186667	-0.528374	0.360606	0.025764
Braunschweig	-0.225	0.166667	0.186667	-0.575238	0.281061	0.054497
Bremen	-0.000	0.111111	0.560000	-0.572768	0.505114	0.124967
Bremerhaven	-0.525	0.000000	0.513333	-0.476460	0.662879	0.024941

3.2.e) Obtaining a Weighted_Customer Data Frame at hand, adding a Total_Score Column on it

Now It is time to add a Total Score for each data row, showing it at the end in a new column as Total_Score

After adding a Total_Score column at the end we have iterated each City row by row to see the sum() of each row in the Total_Score column.

Sorting the Weighted_Customer Data Frame descending order with respect to Total Scores :



```
Weighted_Customer.sort_values(by='Total_Score', ascending=False).head(20)
```

	Clothing Store	Italian Restaurant	Hotel	Income /Pers. €	Average Unemployment rate	Population 2018	Total_Score
City							
Berlin	-0.15	0.055556	0.326667	-0.529582	0.429545	0.800000	0.932185
Bremen	-0.00	0.111111	0.560000	-0.572768	0.505114	0.124967	0.728424
Dresden	-0.15	0.166667	0.700000	-0.508178	0.318182	0.121739	0.648410
Köln	-0.00	0.222222	0.326667	-0.580314	0.416288	0.238292	0.623154
Magdeburg	-0.15	0.444444	0.280000	-0.488949	0.461364	0.052391	0.599251
Mülheim an der Ruhr	-0.15	0.388889	0.513333	-0.620142	0.389773	0.037506	0.559359
Oberhausen	-0.00	0.111111	0.326667	-0.488277	0.538258	0.046275	0.534033
Duisburg	-0.00	0.166667	0.093333	-0.453364	0.609848	0.109435	0.525920
Göttingen	-0.00	0.444444	0.233333	-0.544219	0.299621	0.026295	0.459475
Essen	-0.30	0.277778	0.326667	-0.541399	0.562121	0.127986	0.453153
Gelsenkirchen	-0.30	0.222222	0.186667	-0.435155	0.700000	0.057211	0.430945
Leipzig	-0.15	0.166667	0.466667	-0.513066	0.322159	0.129028	0.421455
Rostock	-0.00	0.166667	0.373333	-0.496938	0.315530	0.045848	0.404440
Lübeck	-0.15	0.277778	0.326667	-0.525715	0.408333	0.047673	0.384735
Münster	-0.00	0.277778	0.326667	-0.597851	0.262500	0.068990	0.338083
Gütersloh	-0.15	0.388889	0.513333	-0.651833	0.214773	0.021992	0.337153
Hamburg	-0.15	0.111111	0.280000	-0.655861	0.334091	0.404119	0.323460
Dortmund	-0.15	0.166667	0.140000	-0.508822	0.540909	0.128842	0.317596
Aachen	-0.00	0.277778	0.140000	-0.531247	0.371212	0.054297	0.312040
Bonn	-0.00	0.277778	0.186667	-0.619202	0.339394	0.071830	0.256466

Berlin seems to be the optimal place to open a clothing store, in respect to Customers wishes.

Following Berlin, the Cities:

Bremen --> Dresden --> Köln --> Magdeburg --> Mülheim an der Ruhr are the other best choices.

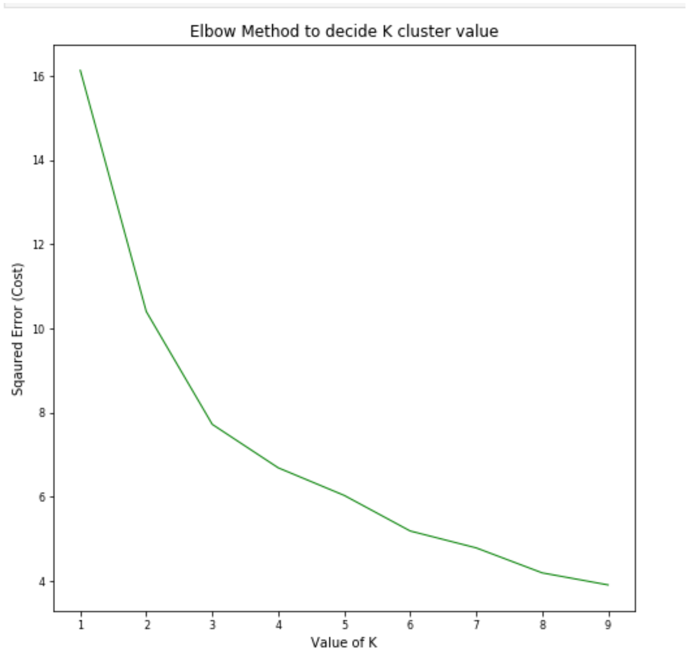
Will be discussed at Results Section.

4) K-Means Clustering and Analyzing the Weighted Customer Data:

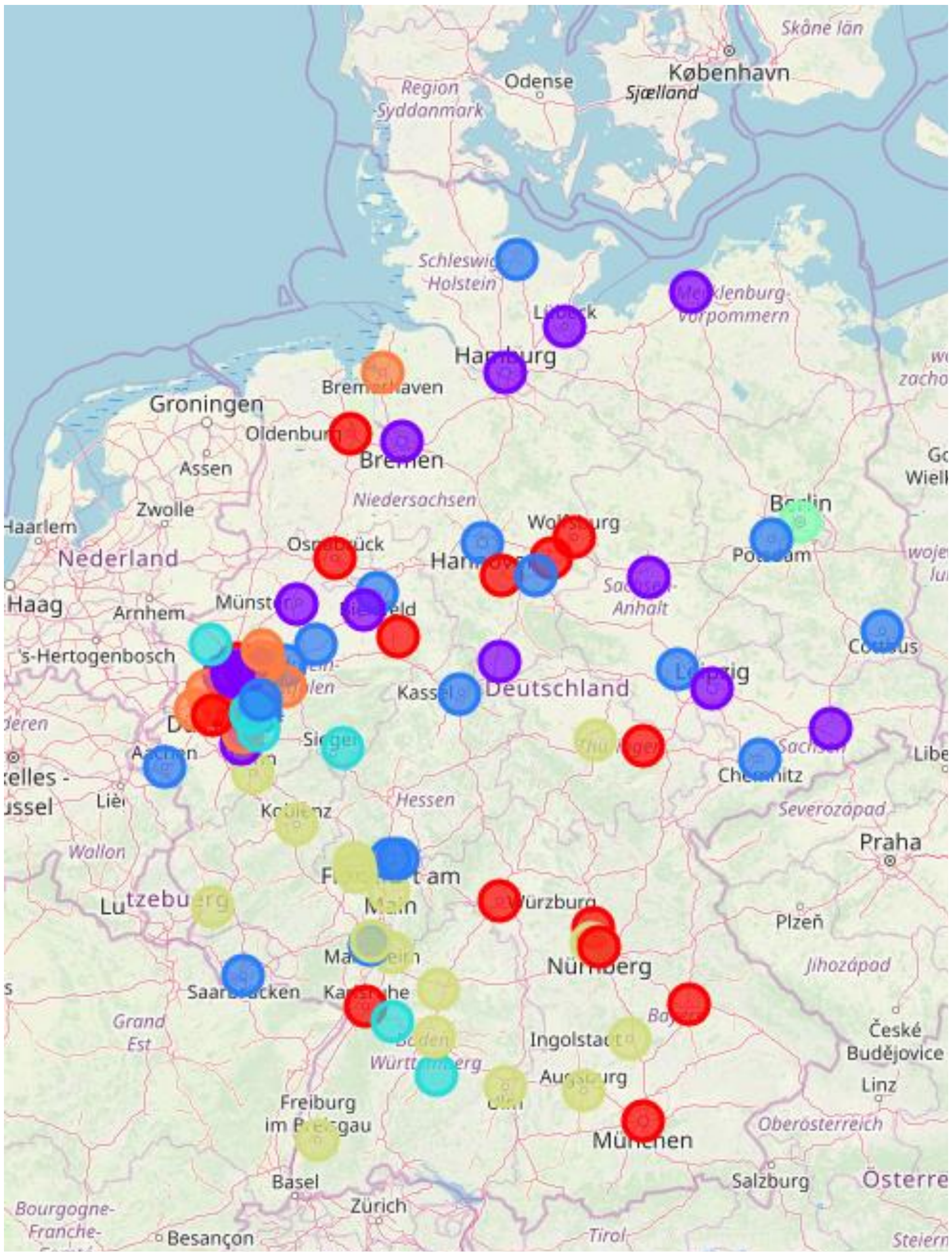
In this section we will use a Machine Learning Algorithm Namely K-Means clustering to our Custer Data set. Why K-Means?

In our case Cities are unlabeled data. And we want to categorize them according to their features. So, under the category of unsupervised Machine Learning Algorithms, k-Means Clustering algorithm seem to be fit best to cluster those cities. Please pay attention even if the cities and venues have their names, it is nothing to do labeled data. We are dealing with the unlabeled data (unsupervised ML) to model.

From the Scikit-learn Machine Learning library we had already imported KMeans module. Now giving initially as an arbitrary k (number of clusters) we will start to our analyze:



K-Means Clusterization of the weighted Customer Data Frame



81 Major cities in Germany are Clustered using the Machine Learning Algorithm: K-Means.

We will discuss the final words in Results section. But Here I want to add a very important note: Please pay attention during the model fitting in K-Means , Data Frame had no (Total_Scores) Column at the end. But the results of the K-Means Clustering and results of the Total_Score method are very consistent!

And indeed, the closer Total_Scored cities, are clustered by K-Means algorithm together too!

You may want to zoom below.

Tabular Form of Clustered Cities

Cluster Labels	City	Total_Score
0	['Bottrop', 'Braunschweig', 'Düsseldorf', 'Erlangen', 'Hildesheim', 'Jena', 'Karlsruhe', 'München', 'Nürnberg', 'Oldenburg', 'Osnabrück', 'Paderborn', 'Regensburg', 'Rhein-Kreis Neuss', 'Wolfsburg', 'Würzburg']	[-0.25533681193957203, -0.11134699528232164, -0.26312197167726953, -0.14164759161416676, -0.03526558624138465, -0.17014798712413495, -0.12601854078927446, -0.10873703494246706, -0.028261982229973837, -0.17347924605989012, -0.09940808957003175, -0.26036597684075197, -0.23517484131749875, -0.3209417118055023, -0.11776645745900668, -0.25425157571628226]
1	['Bremen', 'Dresden', 'Duisburg', 'Essen', 'Gelsenkirchen', 'Göttingen', 'Gütersloh', 'Hamburg', 'Köln', 'Leipzig', 'Lübeck', 'Magdeburg', 'Mülheim an der Ruhr', 'Münster', 'Oberhausen', 'Rostock']	[0.7284238139902227, 0.6484101691176261, 0.5259198422443647, 0.4531525724639104, 0.43094453489877604, 0.4594748687752179, 0.3371534991030384, 0.3234595341169916, 0.623154087028349, 0.4214550320737896, 0.3847353434017572, 0.5992509282273633, 0.5593589274719458, 0.3380825914061856, 0.5340328867988623, 0.40444016289305484]
2	['Aachen', 'Bielefeld', 'Bochum', 'Chemnitz', 'Cottbus', 'Dortmund', 'Frankfurt am Main', 'Halle (Saale)', 'Hamm', 'Hannover', 'Kassel', 'Kiel', 'Ludwigshafen am Rhein', 'Offenbach am Main', 'Potsdam', 'Remscheid', 'Saarbrücken', 'Salzgitter', 'Wuppertal']	[0.31203965895517594, 0.147903893704504, 0.12199720922753482, 0.05589081977943942, 0.09153662535228713, 0.31759577749894863, 0.051483342432015244, 0.04941811803374464, 0.019750804981948485, 0.19195454640655196, 0.10483788101295313, 0.22191677455307413, 0.07308731413700206, 0.15908118174617858, 0.1669995186493847, 0.0948525638656014, 0.10023621515299729, -0.007115330024467653, -0.039578550140776866]
3	['Pforzheim', 'Reutlingen', 'Rheinisch-Bergischer Kreis', 'Siegen-Wittgenstein', 'Solingen', 'Wesel']	[-0.45525705384306253, -0.5747096054585087, -0.6570313577347104, -0.845771228206921, -0.42162653590759047, -0.5225479627003179]
4	['Berlin']	[0.9321852946003613]
5	['Augsburg', 'Bonn', 'Darmstadt', 'Erfurt', 'Freiburg im Breisgau', 'Fürth', 'Heidelberg', 'Heilbronn', 'Ingolstadt', 'Koblenz', 'Mainz', 'Mannheim', 'Stuttgart', 'Trier', 'Ulm', 'Wiesbaden']	[0.047622488452474854, 0.25646562221289015, 0.06925098738542243, 0.1694184644328781, 0.02050800054379448, 0.060343498676200175, 0.1513360630744524, 0.1550215128451404, 0.032927444541146045, 0.22963139056504725, 0.15885364061232438, 0.08916824295268397, 0.024220916407503845, 0.08185342214408324, -0.05610202616857089, 0.20388893585093293]
6	['Bremerhaven', 'Hagen', 'Herne', 'Krefeld', 'Leverkusen', 'Mönchengladbach', 'Recklinghausen']	[0.1996932391539828, -0.2625461432032092, -0.0057417131683230385, -0.0877505982892337, 0.05011577758917075, -0.04809286777550614, -0.2808489582817421]

5) Visualizations

I have already added some extra visualizations to present achieved data frames better. They are not in the scope of the assignment but I hope, readers can find useful information and see how to apply some other methods to their projects.

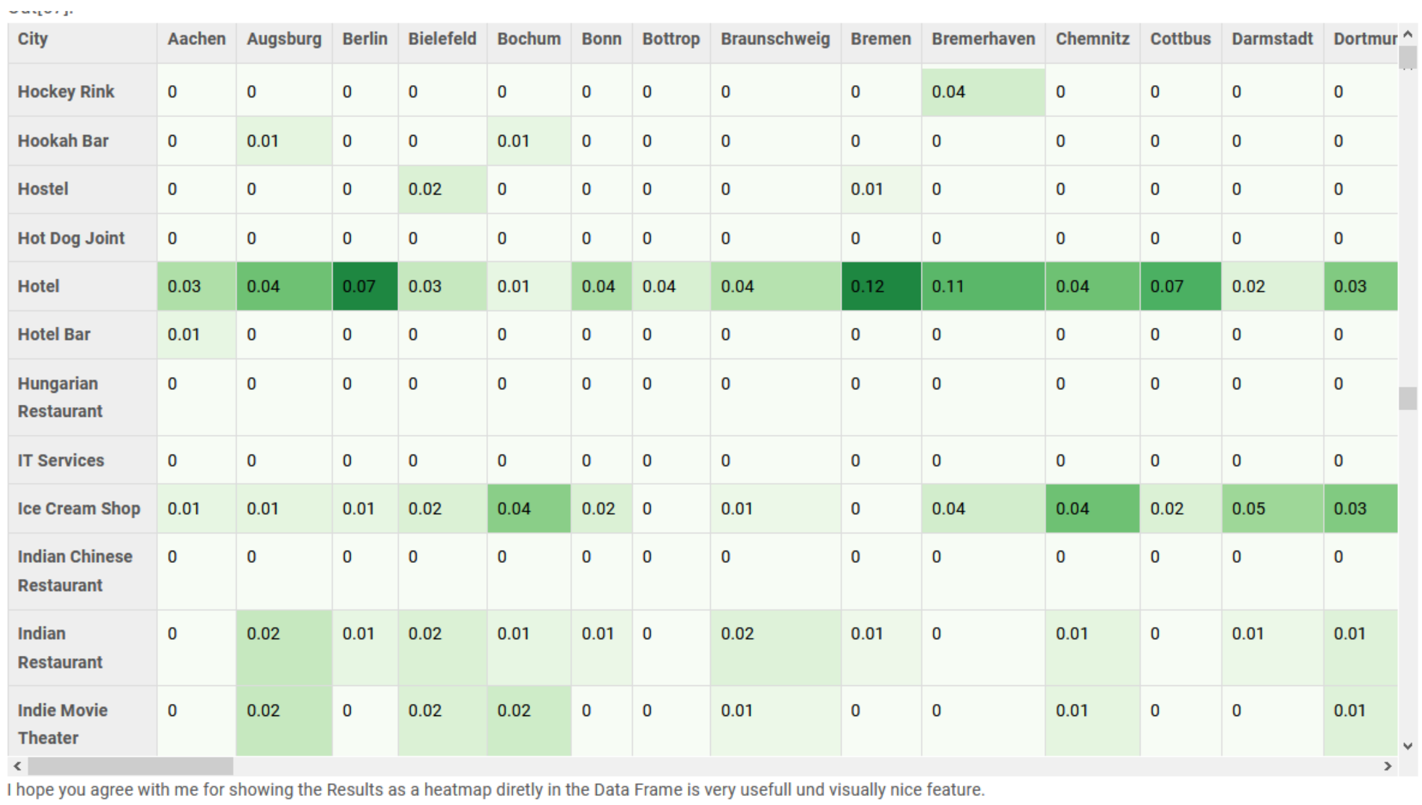
Pandas Data frame as a Heatmap :

This is a Very useful visual feature of Pandas Data Frame .

Please pay attention this feature of Pandas DF library allows us the show (visually) better results

in the usual tabular output of the Data Frame , even without importing any other 3rd python library .

You can easly see the green color code as an indicator. Darker the green color , more the venue frequency in that city :

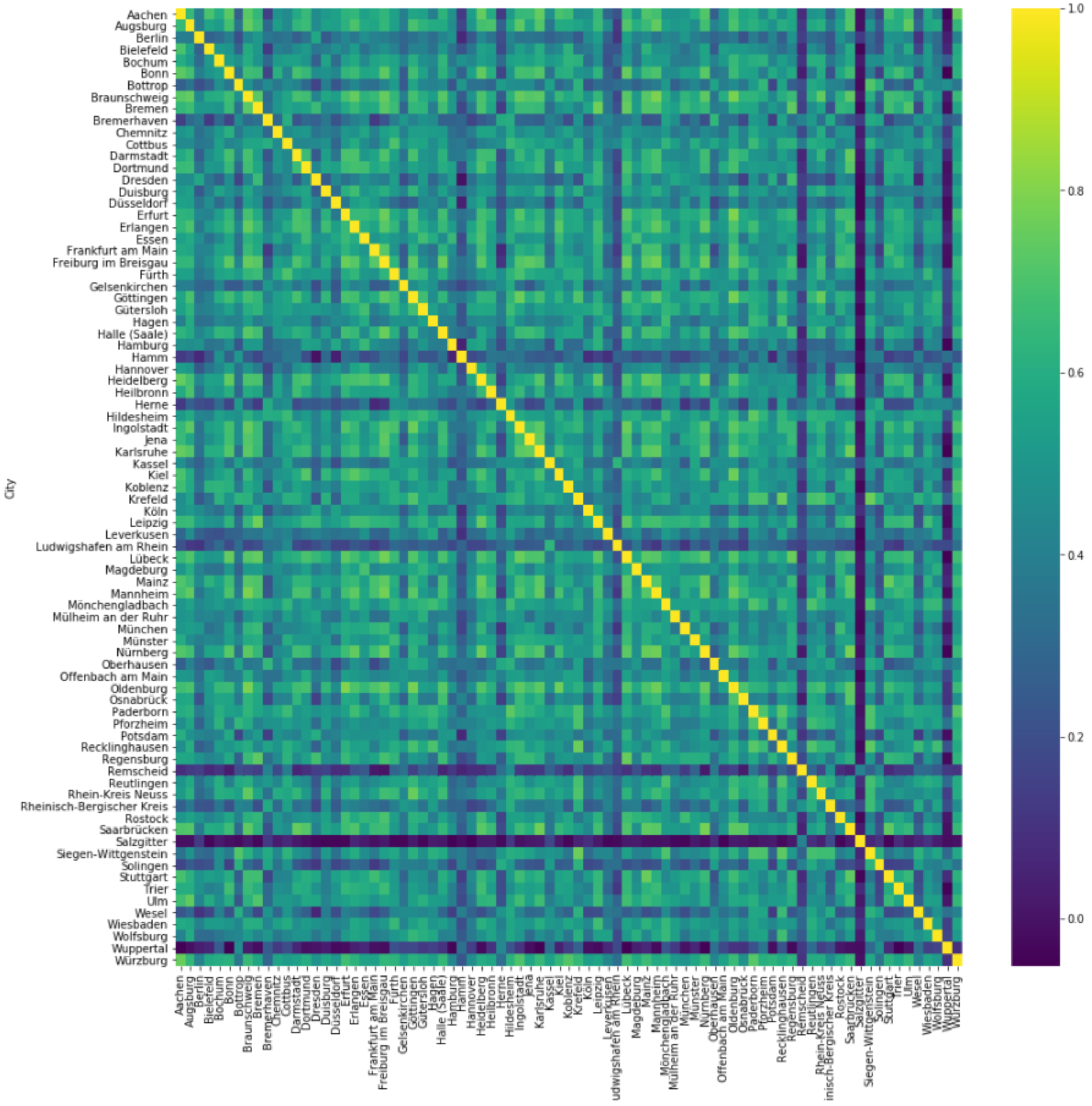


City	Aachen	Augsburg	Berlin	Bielefeld	Bochum	Bonn	Bottrop	Braunschweig	Bremen	Bremerhaven	Chemnitz	Cottbus	Darmstadt	Dortmur
Hockey Rink	0	0	0	0	0	0	0	0	0	0.04	0	0	0	0
Hookah Bar	0	0.01	0	0	0.01	0	0	0	0	0	0	0	0	0
Hostel	0	0	0	0.02	0	0	0	0	0.01	0	0	0	0	0
Hot Dog Joint	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Hotel	0.03	0.04	0.07	0.03	0.01	0.04	0.04	0.04	0.12	0.11	0.04	0.07	0.02	0.03
Hotel Bar	0.01	0	0	0	0	0	0	0	0	0	0	0	0	0
Hungarian Restaurant	0	0	0	0	0	0	0	0	0	0	0	0	0	0
IT Services	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Ice Cream Shop	0.01	0.01	0.01	0.02	0.04	0.02	0	0.01	0	0.04	0.04	0.02	0.05	0.03
Indian Chinese Restaurant	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Indian Restaurant	0	0.02	0.01	0.02	0.01	0.01	0	0.02	0.01	0	0.01	0	0.01	0.01
Indie Movie Theater	0	0.02	0	0.02	0.02	0	0	0.01	0	0	0.01	0	0	0.01

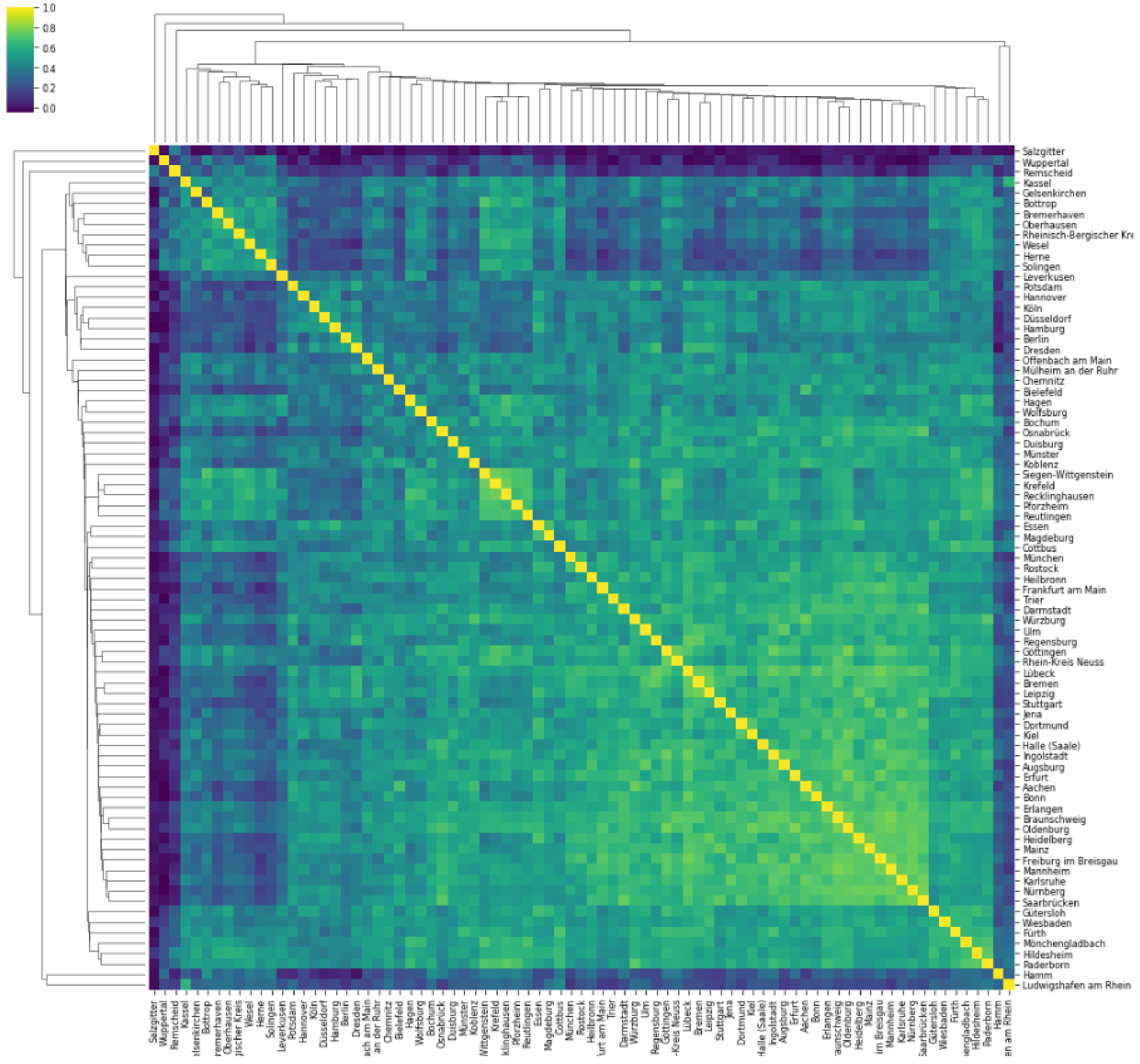
I hope you agree with me for showing the Results as a heatmap diretly in the Data Frame is very usefull und visually nice feature.

Seaborn Correlation Matrix of Cities (General Venues Data):

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff39cb91668>
```



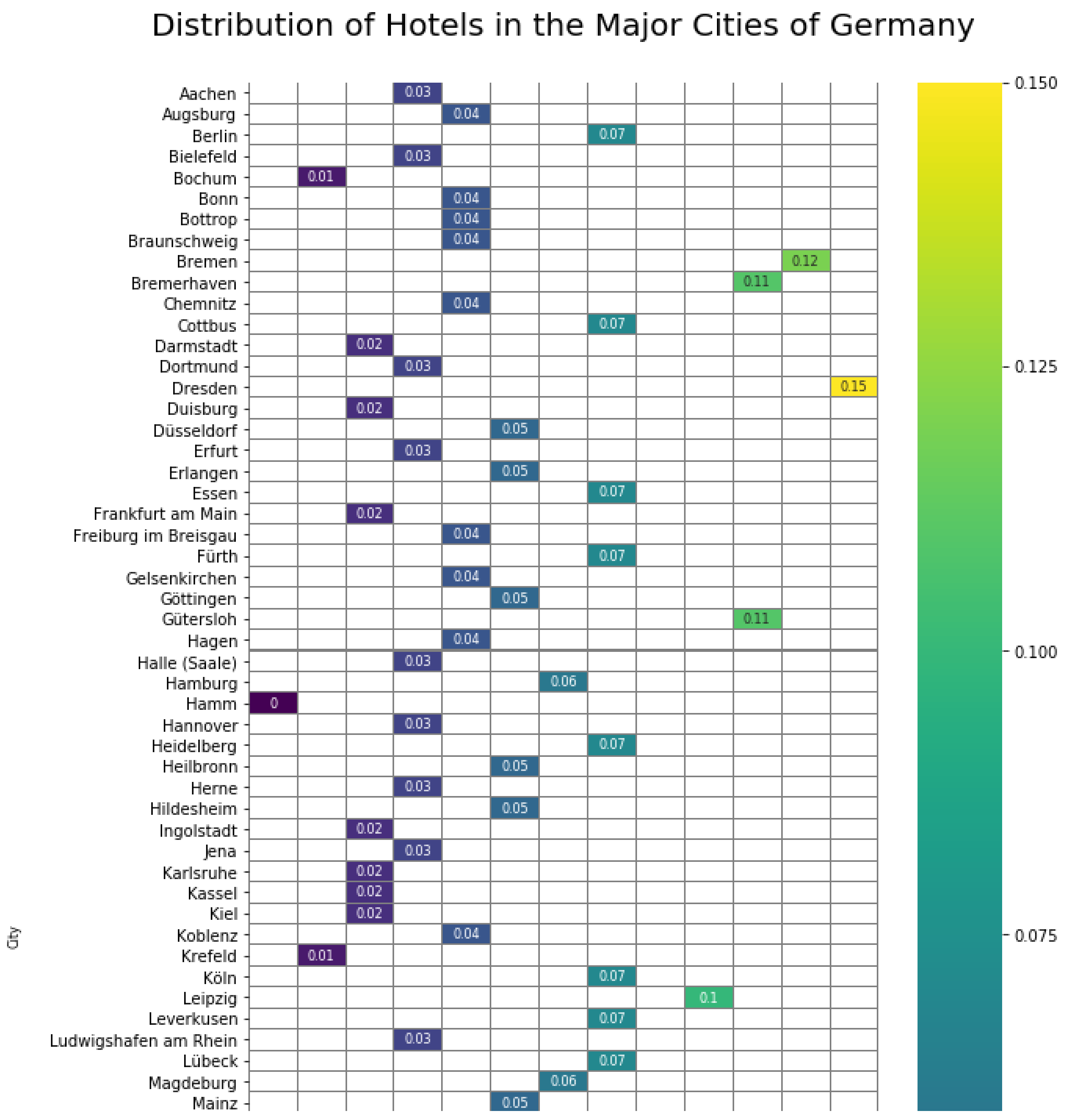
Seaborn Cluster map with Heatmap (General Venues Data):

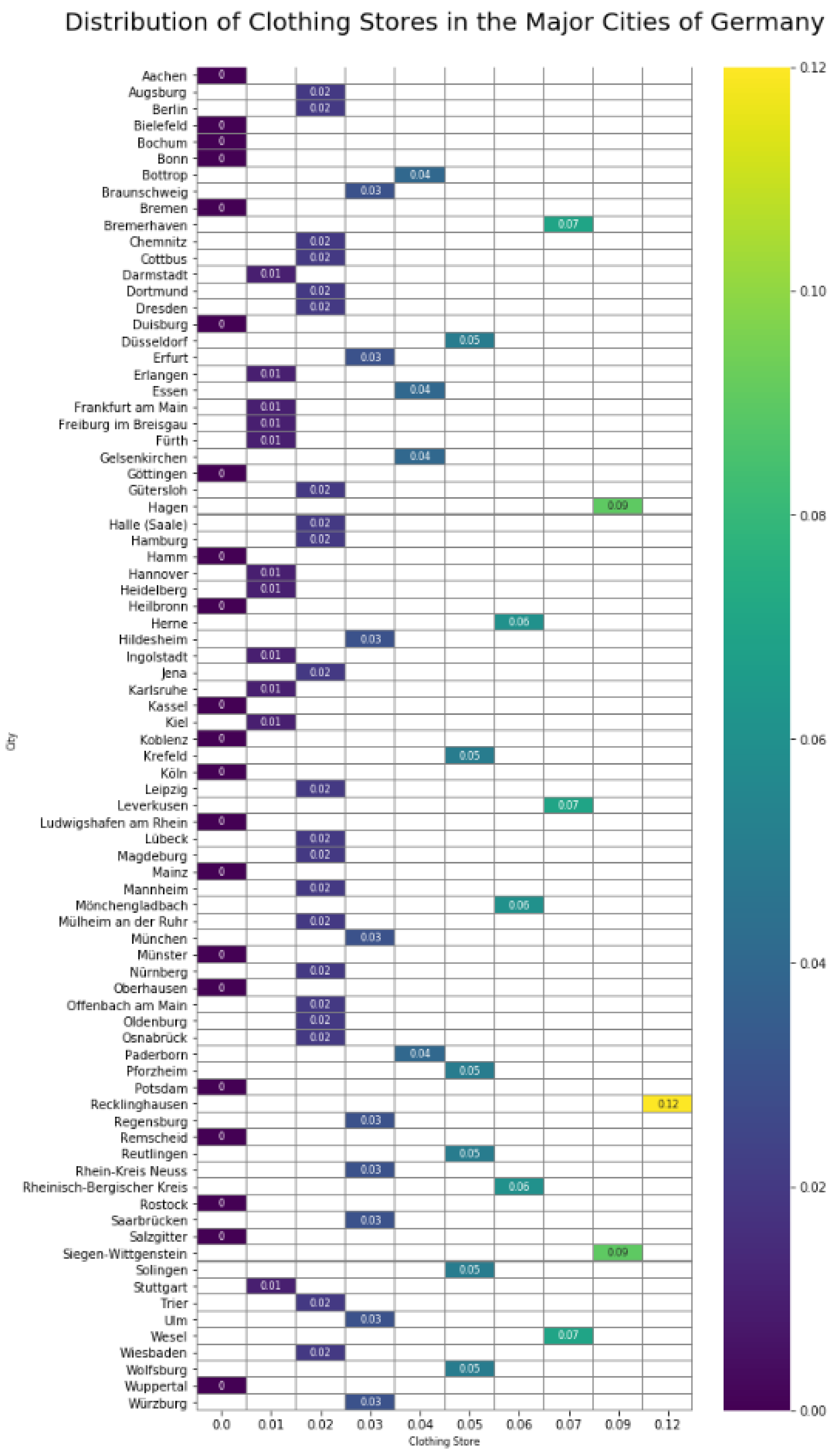


'Target Business' script to Visualize A Single City:

In the notebook under the section 8.d, I have created a cell code to call and visualize the Business of interest easily. Just changing 'Target_Business' variable to get desired Venue's distribution on Cities For example: If the Target_Business ='Hotel' we see that City 'Dresden' is the most competitive City with 0.15 percent venue occurrence in 1250 meters from the City center...

After calling this cell we will see such visualizations:

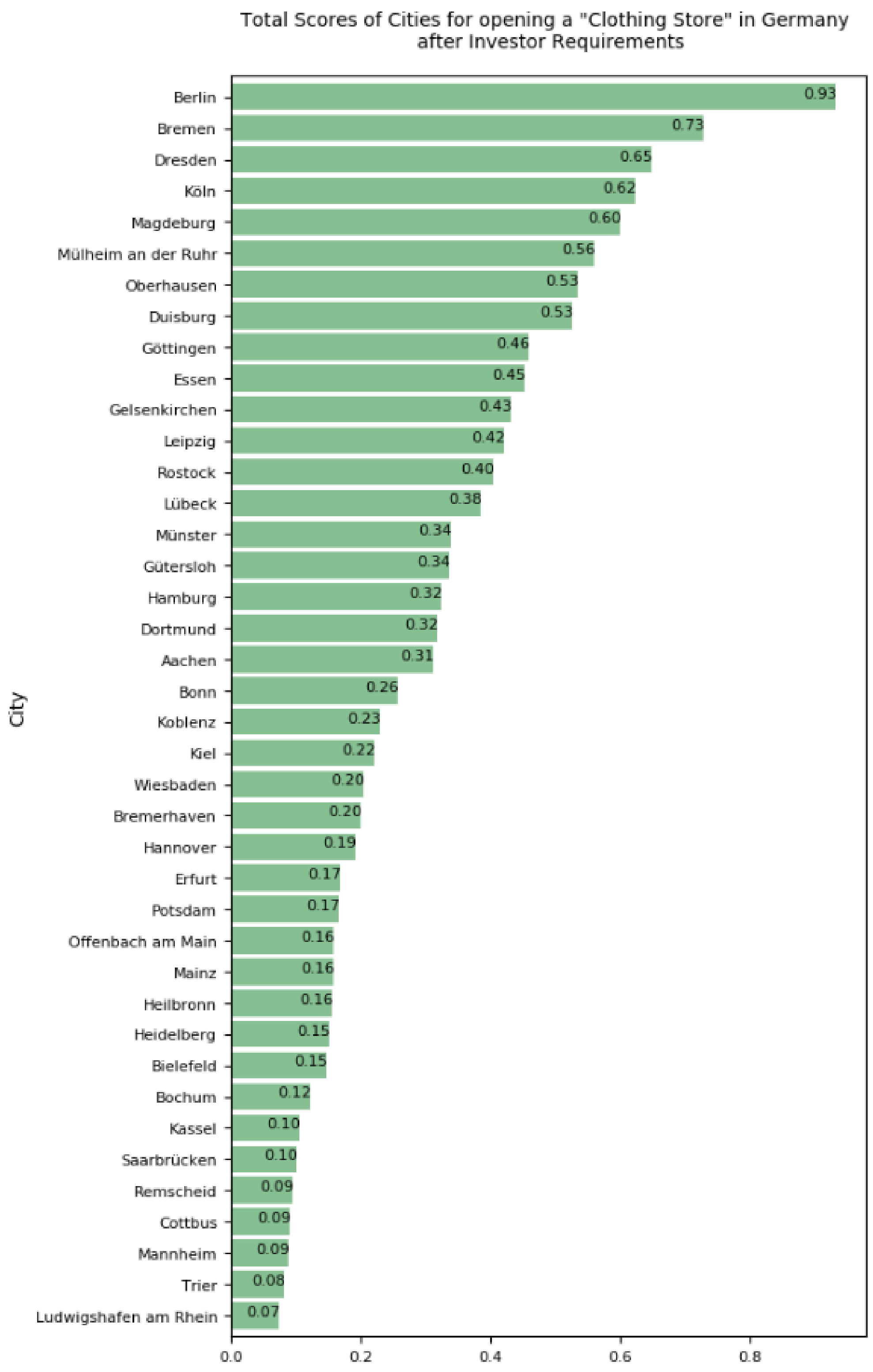



Distribution of Clothing Stores in the Major Cities of Germany
City
Clothing Store
Aachen 0
Augsburg 0.02
Berlin 0.02
Bielefeld 0
Bochum 0
Bonn 0
Bottrop 0.04
Braunschweig 0.03
Bremen 0
Bremerhaven 0.07
Chemnitz 0.02
Cottbus 0.02
Darmstadt 0.01
Dortmund 0.02
Dresden 0.02
Duisburg 0
Düsseldorf 0.05
Erfurt 0.03
Erlangen 0.01
Essen 0.04
Frankfurt am Main 0.01
Freiburg im Breisgau 0.01
Fürth 0.01
Gelsenkirchen 0.04
Göttingen 0
Gütersloh 0.02
Hagen 0.09
Halle (Saale) 0.02
Hamburg 0.02
Hamm 0
Hannover 0.01
Heidelberg 0.01
Heilbronn 0
Herne 0.06
Hildesheim 0.03
Ingolstadt 0.01
Jena 0.02
Karlsruhe 0.01
Kassel 0
Kiel 0.01
Koblenz 0
Krefeld 0.05
Köln 0
Leipzig 0.02
Leverkusen 0.07
Ludwigshafen am Rhein 0
Lübeck 0.02
Magdeburg 0.02
Mainz 0
Mannheim 0.02
Mönchengladbach 0.06
Mülheim an der Ruhr 0.02
München 0.03
Münster 0
Nürnberg 0.02
Oberhausen 0
Offenbach am Main 0.02
Oldenburg 0.02
Osnabrück 0.02
Paderborn 0.04
Pforzheim 0.05
Potsdam 0
Recklinghausen 0.12
Regensburg 0.03
Remscheid 0
Reutlingen 0.05
Rhein-Kreis Neuss 0.03
Rheinisch-Bergischer Kreis 0.06
Rostock 0
Saarbrücken 0.03
Salzgitter 0
Siegen-Wittgenstein 0.09
Solingen 0.05
Stuttgart 0.01
Trier 0.02
Ulm 0.03
Wesel 0.07
Wiesbaden 0.02
Wolfsburg 0.05
Wuppertal 0
Würzburg 0.03
0.0 0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.09 0.12
0.12 0.10 0.08 0.06 0.04 0.02 0.00

6) Results

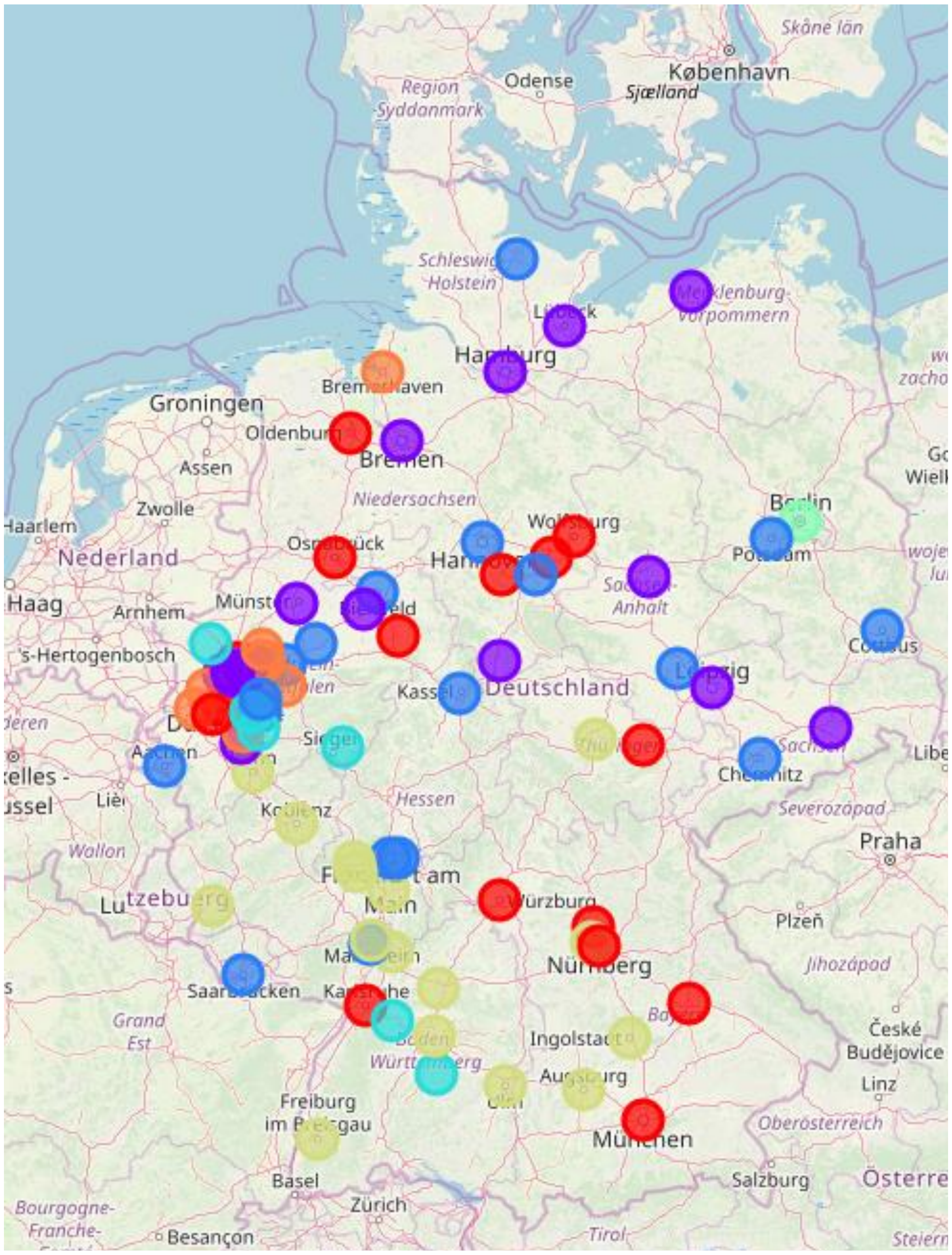
In our project we have used 2 different technics to decide best possible (group) of Cities to our investor can open his Clothing Store.

I would call **method 1** as: **Total_ Score metric** it gives us the numeric results of the City scores after all Customer wishes are applied to Customer point of interest Data Frame :



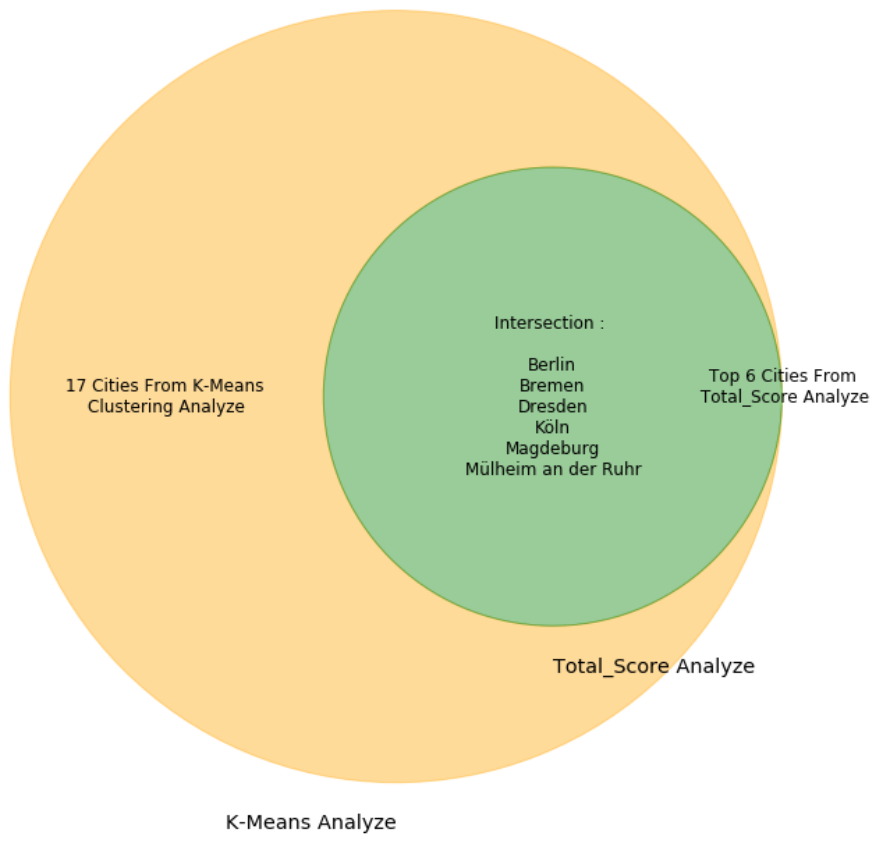
And **Method 2** is: **K-means Analyze**, it gives us may be not directly the descending order of the best cities numerically but, it shows us the which cities are clustered together in accordance to the customer requirements and supports its results with a beautiful folium cluster map, telling us a story :

K-Means Clusterization of the weighted Customer Data Frame

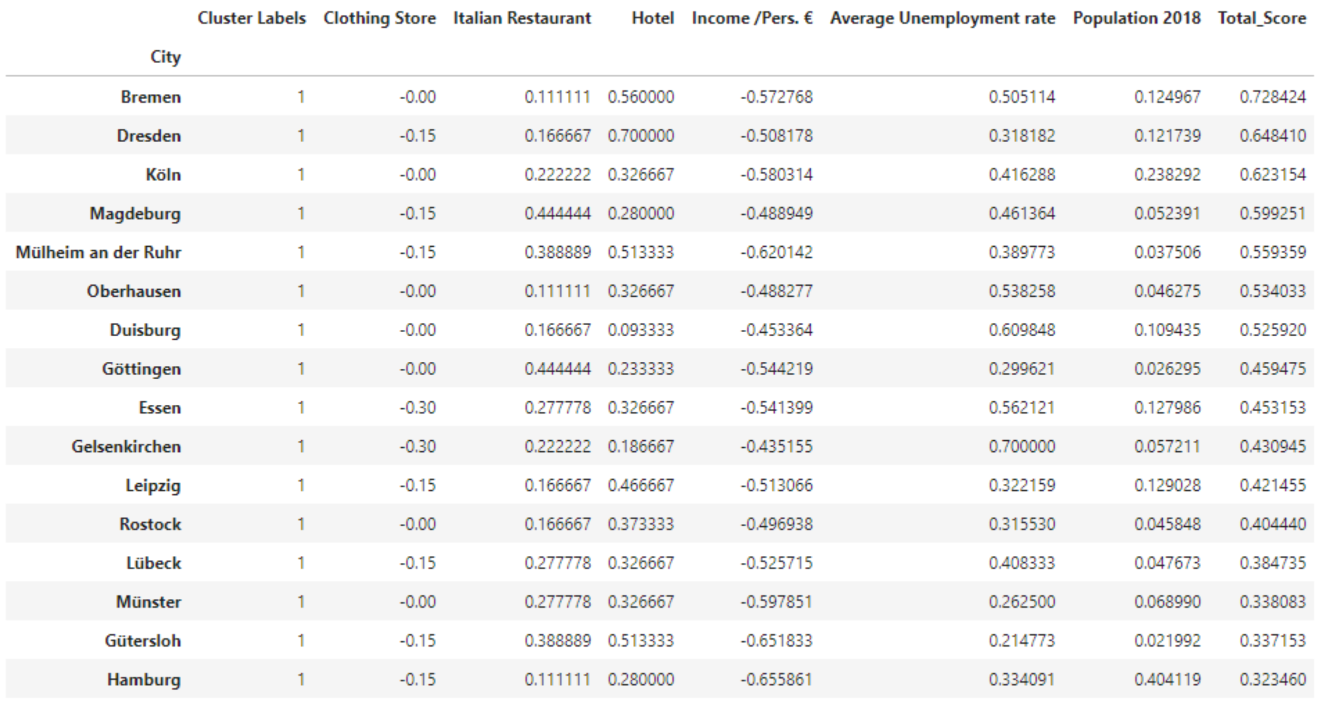


In the Cluster map on the left side Cities with Cluster Labels 4 and 1 are our candidate cities from the Machine Learning Clustering analyze.
Berlin has its own cluster 4. There is no other City in that cluster. (green marker)
Cluster 1: (Purple markers)

Combining 2 Methods together:



The matplotlib Venn diagram on the left side and screenshot from the Data Frame below with cluster 1 Cities gives us to very important clues : We understand that the Cities with Cluster Labels 4 and 1 are the our candidate cities from the Machine Learning Clustering analyze.



City	Cluster Labels	Clothing Store	Italian Restaurant	Hotel	Income /Pers. €	Average Unemployment rate	Population 2018	Total_Score
Bremen	1	-0.00	0.111111	0.560000	-0.572768	0.505114	0.124967	0.728424
Dresden	1	-0.15	0.166667	0.700000	-0.508178	0.318182	0.121739	0.648410
Köln	1	-0.00	0.222222	0.326667	-0.580314	0.416288	0.238292	0.623154
Magdeburg	1	-0.15	0.444444	0.280000	-0.488949	0.461364	0.052391	0.599251
Mülheim an der Ruhr	1	-0.15	0.388889	0.513333	-0.620142	0.389773	0.037506	0.559359
Oberhausen	1	-0.00	0.111111	0.326667	-0.488277	0.538258	0.046275	0.534033
Duisburg	1	-0.00	0.166667	0.093333	-0.453364	0.609848	0.109435	0.525920
Göttingen	1	-0.00	0.444444	0.233333	-0.544219	0.299621	0.026295	0.459475
Essen	1	-0.30	0.277778	0.326667	-0.541399	0.562121	0.127986	0.453153
Gelsenkirchen	1	-0.30	0.222222	0.186667	-0.435155	0.700000	0.057211	0.430945
Leipzig	1	-0.15	0.166667	0.466667	-0.513066	0.322159	0.129028	0.421455
Rostock	1	-0.00	0.166667	0.373333	-0.496938	0.315530	0.045848	0.404440
Lübeck	1	-0.15	0.277778	0.326667	-0.525715	0.408333	0.047673	0.384735
Münster	1	-0.00	0.277778	0.326667	-0.597851	0.262500	0.068990	0.338083
Gütersloh	1	-0.15	0.388889	0.513333	-0.651833	0.214773	0.021992	0.337153
Hamburg	1	-0.15	0.111111	0.280000	-0.655861	0.334091	0.404119	0.323460

Being the first City Berlin (Cluster 4), following cities are in our second-best cluster (cluster 1):

Bremen / Dresden / Köln / Magdeburg / Mülheim an der Ruhr /Oberhausen / Duisburg /Göttingen /Essen / Gelsenkirchen / Leipzig / Rostock /Lübeck /Münster /Gütersloh /Hamburg.

And all these Cities have obviously Higher Total_Score too. But please pay attention, during the clustering we did not have the Total_Score Column, it didn't count to clustering mechanism. But we can cross validate compare the results obtained from the clustering with K-means, and results from the Total_Score evaluation.

And yes really after comparing these two result: we have a 100 % of overlapping Results. (i.e. All the Cluster 4 and 1 Cities have already very top Total_Scores from the other metric. So, this conclude and assures our confidence to to tell the story to our investor (customer) ....

So, our final List in the right order to advice our investor would be:

1) **Berlin**
2) **Bremen**
3) **Dresden**
4) **Köln**
5) **Magdeburg**
6) **Mülheim an der Ruhr**

We have listed 6 cities; it is for, customer has other top 5 alternatives if capital city is not desired by him.

7)Discussion Recommendations

Like in every engineering problem happens we have also made some assumptions and simplifications to understand and present solution in a better way. One of the biggest problems in this project, I would say source of the Venue Data. We are using a USA based Data supplier , Foursquare to judge places venues in the Germany , this data seem to be missing and in some parts are not suitable to our problem. Tagging of venues are made from the Foursquare users , I am assuming they are mainly not Germans. It is highly possible some of venues are wrong tagged or skipped, not listed in api call results. So, as a further improvement one can go professional and buy a real data set especially prepared for the Germany (or any other country in your analyzes).

Second issue I would mention, dimension of the features that shapes the customer wishes matrix, are limited and imaginary, every weight change in that matrix would change the results dramatically. For example if he would give the population less weight as 0.8 , the Berlin could be skipped from the first place.

8)Conclusion

In this project I wanted to create and solve a real-life problem rather than repeating the classical clustering assignment from our course. I made this assignment much more complicated than the necessary and because of that, I have to spent too much time than the normal case as you appreciate. But just because of that complexity and, having no example in front of me, I have learned too many things from this project. I had to think and find ideas myself like, weight matrix, customer wishes, adapting all those to still our assignment guidelines. So, it may include some wrong conceptions, but still I hope enjoy reading and evaluating it. I hope this project will help me the find a right role in the Data Science world.

Beside the project complexity, I also decided to go with a new website for this and other following projects. Doing everything from the scratch, was maybe the not best idea, but I hope worth it.

Thank You for your Reading and understanding, good luck to all new comer friends who will read these lines in the future.

Best Regards
Bsc. Ing Uygar Hizal
10.12.2019

List of Data Sources References:

- **Foursquare API data based on free API calls**
- **Wikipedia site for Major Cities List in Germany**
- **GeoPy library to fetch coordinates of the Major cities https://geopy.readthedocs.io/en/stable/**
- **Federal Statistical Office of Germany – (Statistisches Bundesamt) https://www.destatis.de/**
- **The jupyterlab environment usage: https://labs.cognitiveclass.ai/tools/jupyterlab/**