Predicting your next Vacation

Lucas Gärtner

Bremen, Germany

e-mail: bartner@msn.com

August 08, 2020

Abstract- A vacation in the first step is a process(e.g. Obtain, Explore and many thoughts) usually consists different steps and can be understood as a workflow. Most modern techniques and tools for extracting the information's are recommendation systems. Based on their knowledge they will recommend you some cities which are popular for vacation. Developing a suitable IT Infrastructure for a self-service platform is an interesting point of view, just to make vacation based on your local shops and hometown. In this paper, I show a prototypical implementation for this purpose.

1. Introduction

One of the most important aspects of building service portfolios for the public is to make them as simple and usable as possible for the end user. The research was done with the interpreted programming language Python3, version 3.7.4. Jupyter notebooks are one of the best friends for data scientists, at the moment the whole workflow is only available for programmers or the end user with experience with Jupyter notebooks.

1.1 Problem

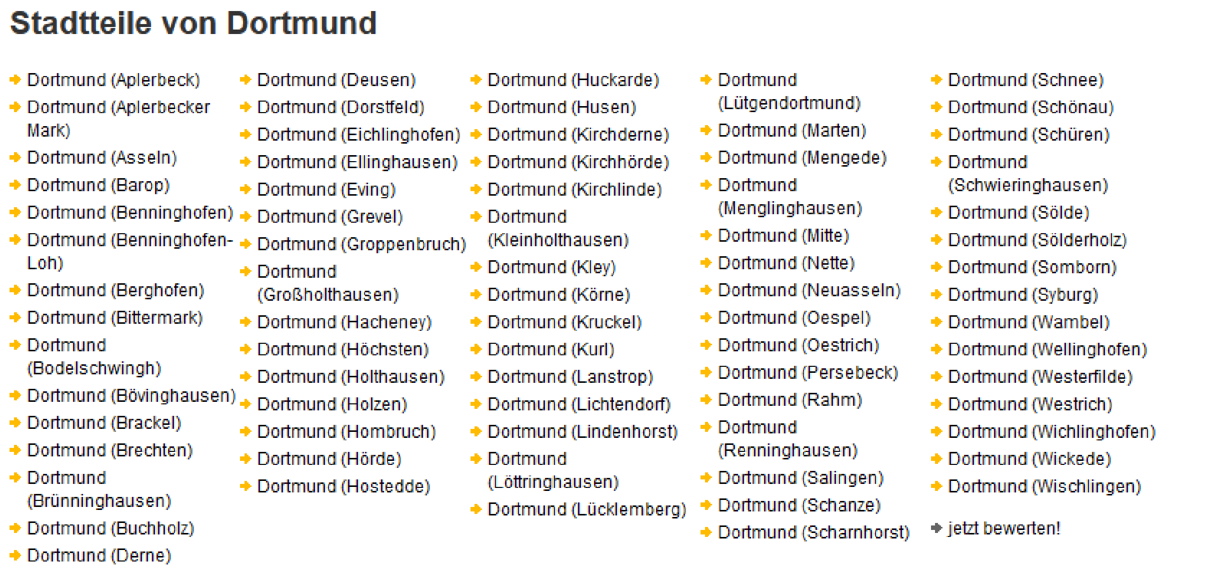
Data that might contribute to determining districts which got the same name as other countries got. This project aims to predict wether and how much shops you got in your hometown. This may got difficult for some Villages without shops.

1.2 Interests

Obviously, some end users would love to make the next weekend trip to a city next their hometown with the same or familiar shopping vibes.

2. Data acquisition and cleaning

In Germany there is a website(https://www.immowelt.de) where you are able to extract all districts from a city. In the past there was an Web-API for this, but unfortunately they didn't give access at the moment, so I hard coded the link to the Districts from Dortmund(German: Stadtteile von Dortmund)



Stadtteile von Dortmund

- Dortmund (Aplerbeck)
- Dortmund (Aplerbecker Mark)
- Dortmund (Asseln)
- Dortmund (Barop)
- Dortmund (Benninghofen)
- Dortmund (Benninghofen-Loh)
- Dortmund (Berghofen)
- Dortmund (Bittermark)
- Dortmund (Bodelschwingh)
- Dortmund (Bövinghausen)
- Dortmund (Brackel)
- Dortmund (Brechten)
- Dortmund (Brünninghausen)
- Dortmund (Buchholz)
- Dortmund (Derne)
- Dortmund (Deusen)
- Dortmund (Dorstfeld)
- Dortmund (Eichlinghofen)
- Dortmund (Ellinghausen)
- Dortmund (Eving)
- Dortmund (Grevel)
- Dortmund (Groppenbruch)
- Dortmund (Großholthausen)
- Dortmund (Hacheney)
- Dortmund (Höchsten)
- Dortmund (Holthausen)
- Dortmund (Holzen)
- Dortmund (Hombruch)
- Dortmund (Hörde)
- Dortmund (Hostedde)
- Dortmund (Huckarde)
- Dortmund (Husen)
- Dortmund (Kirchderne)
- Dortmund (Kirchhörde)
- Dortmund (Kirchlinde)
- Dortmund (Kleinholthausen)
- Dortmund (Kley)
- Dortmund (Körne)
- Dortmund (Kruckel)
- Dortmund (Kurl)
- Dortmund (Lanstrop)
- Dortmund (Lichtendorf)
- Dortmund (Lindenhorst)
- Dortmund (Löttringhausen)
- Dortmund (Lücklemberg)
- Dortmund (Lütgendortmund)
- Dortmund (Marten)
- Dortmund (Mengede)
- Dortmund (Menglinghausen)
- Dortmund (Mitte)
- Dortmund (Nette)
- Dortmund (Neuasseln)
- Dortmund (Oespel)
- Dortmund (Oestrich)
- Dortmund (Persebeck)
- Dortmund (Rahm)
- Dortmund (Renninghausen)
- Dortmund (Salingen)
- Dortmund (Schanze)
- Dortmund (Scharnhorst)
- Dortmund (Schnee)
- Dortmund (Schönau)
- Dortmund (Schüren)
- Dortmund (Schwieringhausen)
- Dortmund (Sölde)
- Dortmund (Sölderholz)
- Dortmund (Somborn)
- Dortmund (Syburg)
- Dortmund (Wambel)
- Dortmund (Wellinghofen)
- Dortmund (Westerfilde)
- Dortmund (Westrich)
- Dortmund (Wichlinghofen)
- Dortmund (Wickede)
- Dortmund (Wischlingen)

- jetzt bewerten!

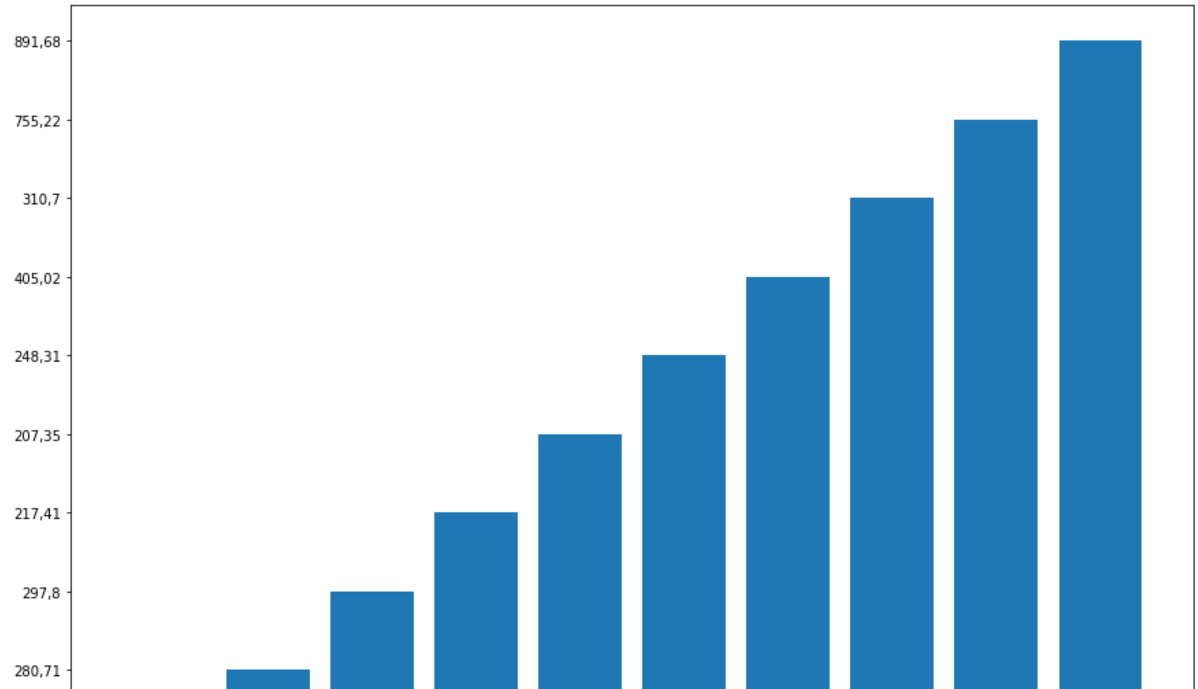
I created a web-scraping job which extracts the Districts from the div box. So I was able to create a Dictionary with Big Cities as keys and a list of all districts. I got a list list from a website where the top 10 largest Cities are listed. At the end the result was as follows:

```
The City Essen got 52 districts
The City Dortmund got 75 districts
The City Leipzig got 65 districts
The City Düsseldorf got 53 districts
The City Stuttgart got 57 districts
The City Frankfurt am Main got 46 districts
The City Cologne got 86 districts
The City Munich got 26 districts
The City Hamburg got 104 districts
The City Berlin got 101 districts
Total count of all districts:  665
```

2.1 Feature selection

The goal is to compare cities based on their shops, but I extracted one feature more just to check if the data is more distinguishable.

The extraction of the surface was an interested feature:



I decided to check one more feature. Population based on the cities is not a feature which got an impact of the shops category comparison.

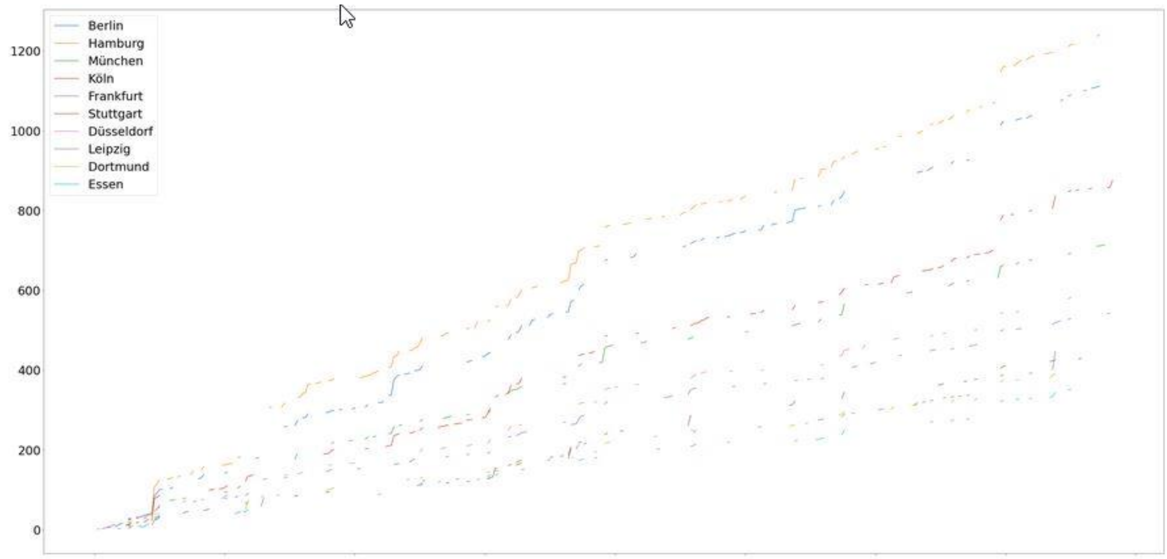
As a data scientist you will often get to the point where you will be questioning yourself: Are these features enough?

With this question in my mind I extracted the shops based on the Districts of the top 10 largest cities in Germany. I create a nested dictionary to store all informations generated from the Foursquare API.

3. Exploratory Data Analysis

After extracting all informations I plotted the results into a OpenstreetMap Plot, the research worked to my satisfaction. I choose to calculate the distance between shops and append them to my pandas dataframe.

The received categories from the Foursquare API was a little unclear, so I plotted the categories for each city:

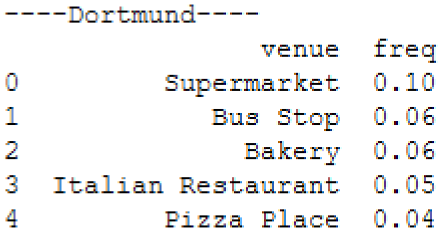


To my satisfaction some of them was overlapping, which means that each city got mostly the same categories. Some of them got more and different categories like Berlin or Hamburg for example. The most known cities in Germany are these two, so that was satisfactory.

After creating a nested dictionary within all information's which we gained from scraping, or the API I transformed it into a pandas dataframe. It is easier to handle the dataframe then a dictionary for the next cases.

So, we created a dataframe and want to cluster cities based on their shops.

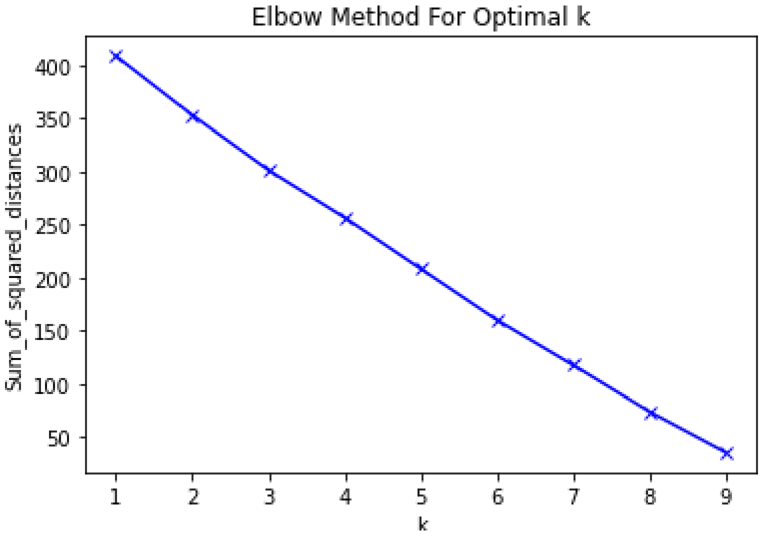
First we have to one-hot encode the categories from the dataframe. With the one-hot encoded categories we are able to create a frequency how often it is represented in a city, the top 5 categories will look like:



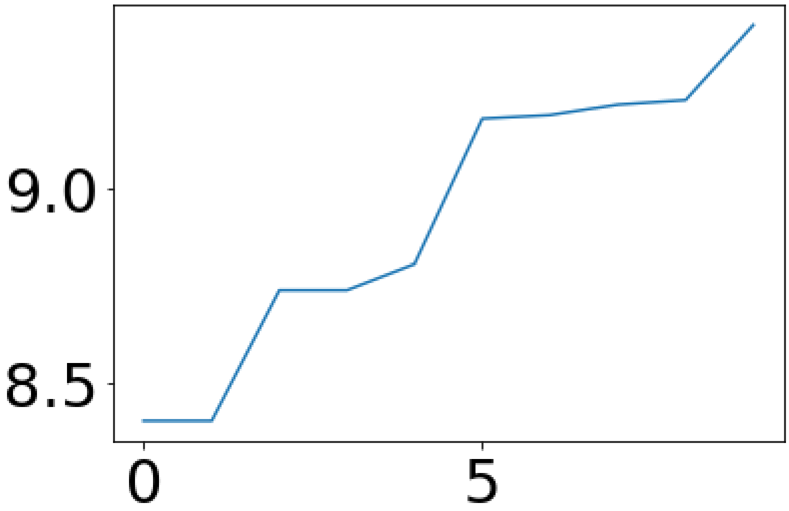
```
----Dortmund----
                venue  freq
0         Supermarket  0.10
1            Bus Stop  0.06
2              Bakery  0.06
3  Italian Restaurant  0.05
4         Pizza Place  0.04
```

4. Methods

I tried the elbow method for the optimal k and the DBSCAN Method both did not give me a good k:



When I tried the nearest neighbor Method, it was quite satisfactory.

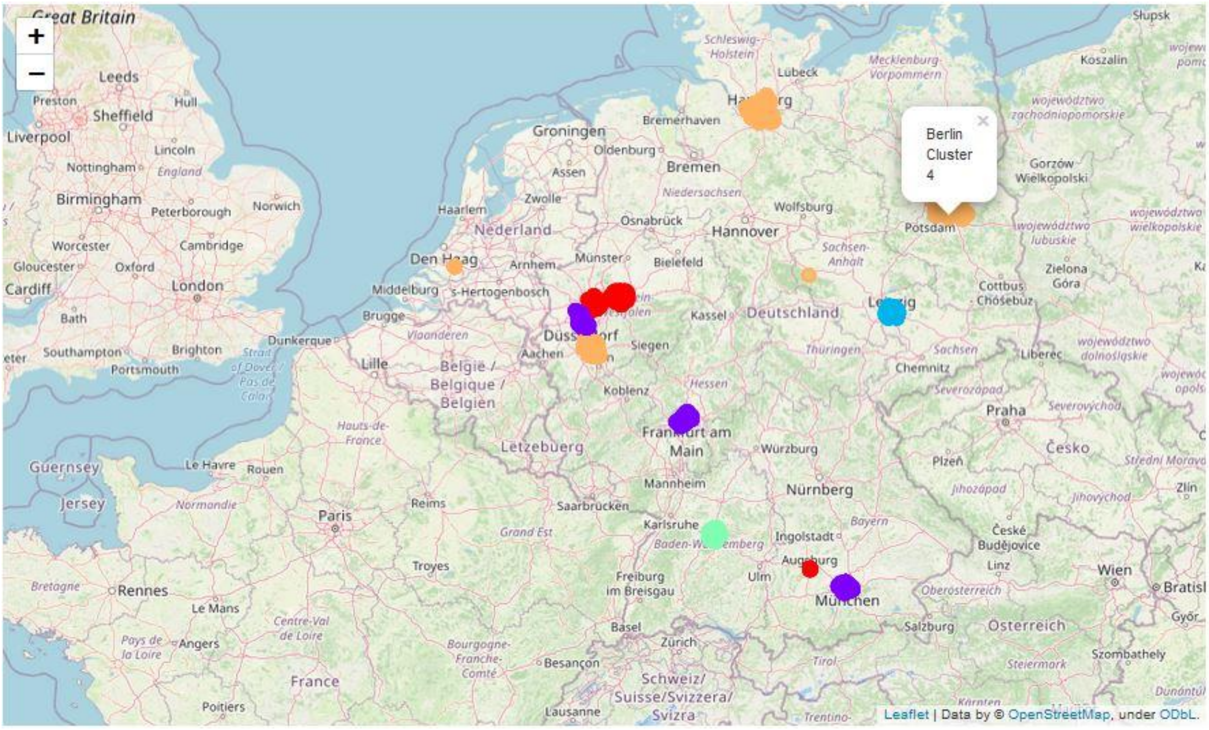


The best k is 5. AS you can see the elbow method did not work to my satisfaction for this issue I took DBSCAN which also was not the best mathematical optimization method. Maybe nearest neighbor is not the best method for my result, but I helped me to find a good K. I will read more about optimization methods to find out a better way.

5. Conclusions

The point where I faced there is too less data to publish this post was discontented. Nevertheless, after a long sleep I decided to publish this post. I did not find a satisfying mathematical optimization method. After the Elbow Method, DBSCAN, I decided to go with nearest Neighbor. I will go the extra mile and find my satisfying optimization method.

Satisfactory was the point where I saw that the clusters based on the shops from the whole Citie and districts makes sense the way it is clustered, as you can see here :



Berlin, Hamburg and Cologne are the diversity cities of Germany, next ones are Munich, Frankfurt am Main and Düsseldorf. The moment I realized that, was the point where I saw that there is no need for more features at the moment.

6. End user conclusion

With this project you can identify which city has some overlapping shops to your hometown. With this approach you may able to identify which city has the most overlapping interests based on the categories from the shops. So, look at it when you want to travel to a new City. Give it a try and do not hesitate to contact me.

7. Future directions

I will set up an Airlfow Workflow which will grab cities around each country save it into a database and will create a dockerized Web app. Within this app you will be able to choose your cities and based on your chosen city you will get an recommendation which cities are comparably. After setting up this process I will update this post.