The Battle of Neighborhoods

Find the Best Location for a New Restaurant in Dusseldorf

Xinya Xu

April 04, 2021

1 Business Problem and Background

In this project we will try to find an optimal location for a restaurant. Specifically, this report will be targeted to stakeholders interested in opening a new restaurant in Düsseldorf, Germany.

Since there are lots of restaurants in Düsseldorf we will try to detect the locations which are already crowded. We will compare the result with the population density of Düsseldorf to find the optimal location for a new restaurant, which would be in an area of dense population and with no or few restaurants in close vicinity.

We will use our data science powers to generate a map which will show the discussed information. Advantages of each area will then be clearly expressed so that best possible final location can be chosen by stakeholders.

2 Data

2.1 Required Data

Based on our requirements, the following data will be needed:

- Population density of each neighborhood in Dusseldorf
- Restaurants in each neighborhood in Dusseldorf and their coordinates (latitude and longitude)
- To visualize the density of population of neighborhoods by using Folium, we will need a GeoJSON of Dusseldorf with the boundaries between different neighborhoods
- To find the restaurants in each neighborhood by using Foursquare API, we will need the coordinates of each neighborhood in Dusseldorf

2.2 Data Source

There is a website named "Open Data Düsseldorf" which provides extensive data relating to Düsseldorf:

- For population of each neighborhood, the csv file "Einwohnerzahl nach Stadtteilen_seit2012.csv" is downloaded from the website. (Link to population data)
- To calculate the density, we need to know the area of each neighborhood. For this, the csv file "Fläche der Stadtteile von Düsseldorf in Quadratkilometern_1_0.csv" is downloaded from the website. (Link to area data)
- For the boundaries between different neighborhoods, the json file "Stadtteile_WGS84_4326.json" is downloaded from the website. (Link to GeoJSON data)

The coordinates of each neighborhood come from Wikipedia page of each neighborhood. A dataframe will be created manually by using this information.

For restaurants in each neighborhood and their coordinates, we will use Foursquare API service to pass the coordinate of each neighborhood to explore the "Restaurant" venues in the neighborhood.

2.3 Data Acquisition & Pre-Processing

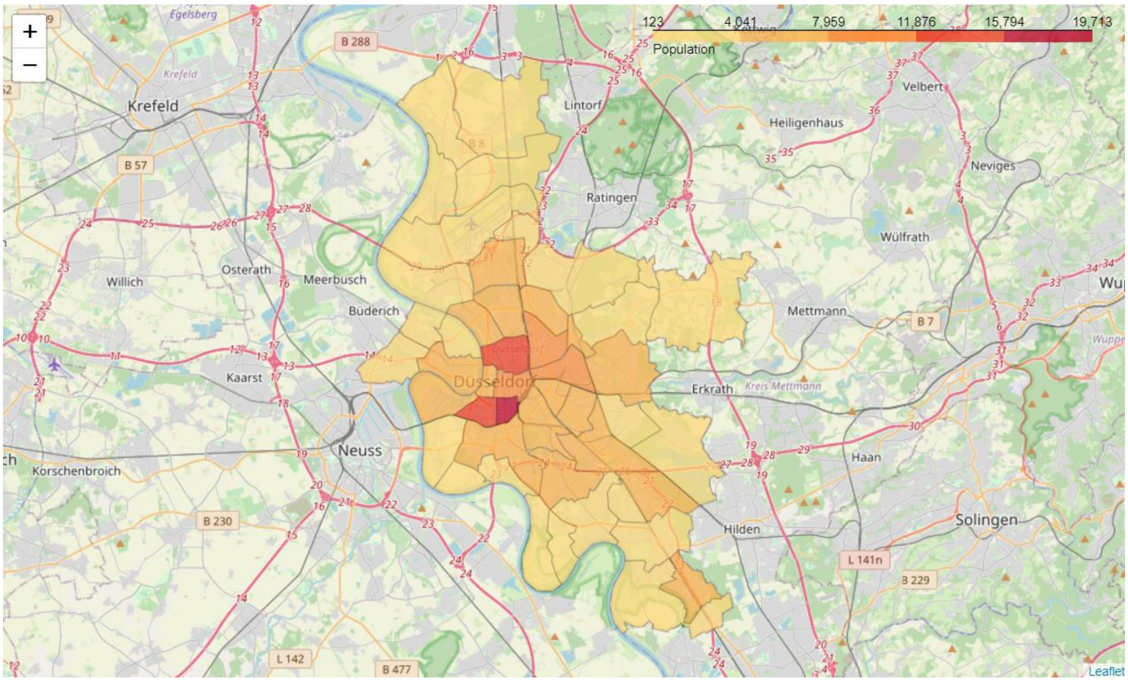
Firstly, I read the population data mentioned above into a dataframe. However, there were several problems with the datasets. Two neighborhoods have names which contain additional character. I needed to remove it. Moreover, the neighborhood Unterbilk and Hafen are in the same row. I needed to split them up to two individual rows. It is very important to have the same data content and size as the data in GeoJSON file since the visualization tool will compare the geographical data with the population data based on the same neighborhood names.

Secondly, I read the area data into a dataframe to further calculate the population density. Then I merged the two dataframe – population and area – in to one dataframe based on the neighborhoods. At the end, the density was calculated as division of population by area and saved in the same dataframe.

Thirdly, I manually created a dataframe with the coordinates of each neighborhood in Dusseldorf.

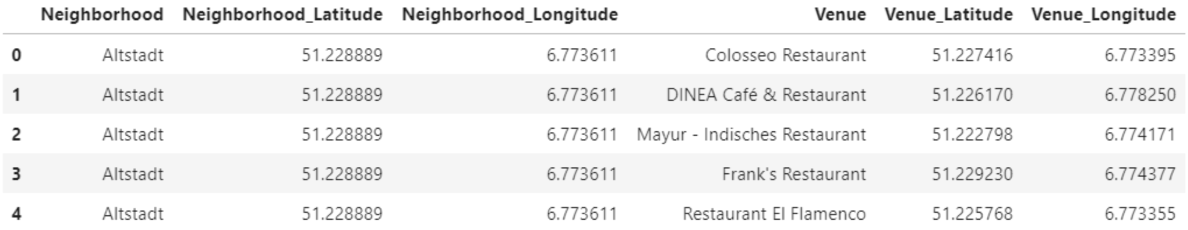
At the last, I read the GeoJSON file into a json variable to be used later for visualization.

After the required data was acquired and pre-processed. An initial sample map could be generated to show the population density of each neighborhood in Dusseldorf. To create the map, the Folium library and its function “Choropleth” was used. A Choropleth map is a thematic map in which areas are shaded or patterned in proportion to the measurement of the statistical variable being displayed on the map, such as population density or per-capita income. The choropleth map provides an easy way to visualize how a measurement varies across a geographic area or it shows the level of variability within a region. The GeoJSON variable, which contains the boundaries between different neighborhoods and the population density data were passed to the Choropleth graph function. The following map was generated.



3 Venues Exploration

One of the most important information to gain are the restaurants and their distribution in Dusseldorf across every neighborhood. For this purpose, I used the Foursquare API service to search the restaurants in each neighborhood. Here is where the coordinate data came into use. In addition with a search query which represents the venue to be searched, in our case ‘Restaurant’, was this data passed to the API call and a response with results was returned. The most important information was stored to a dataframe.

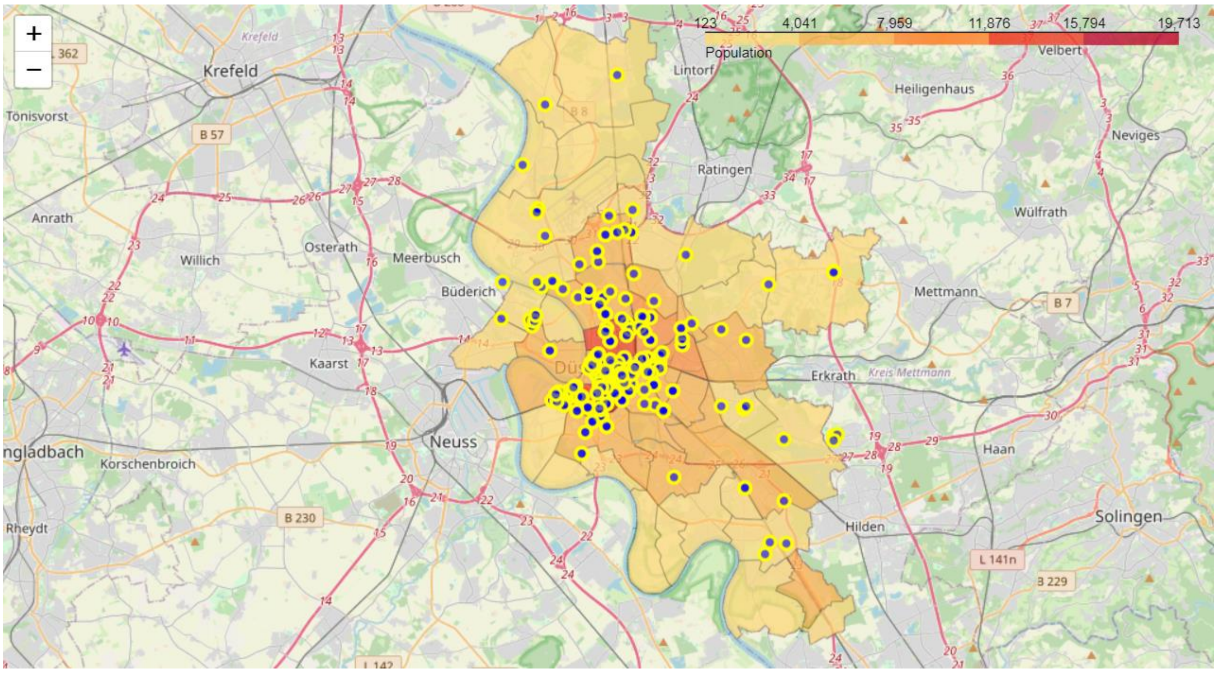


	Neighborhood	Neighborhood_Latitude	Neighborhood_Longitude	Venue	Venue_Latitude	Venue_Longitude
0	Altstadt	51.228889	6.773611	Colosseo Restaurant	51.227416	6.773395
1	Altstadt	51.228889	6.773611	DINEA Café & Restaurant	51.226170	6.778250
2	Altstadt	51.228889	6.773611	Mayur - Indisches Restaurant	51.222798	6.774171
3	Altstadt	51.228889	6.773611	Frank's Restaurant	51.229230	6.774377
4	Altstadt	51.228889	6.773611	Restaurant El Flamenco	51.225768	6.773355

During the project I recognized that there is a limitation of the results number returned by the API service up to 50. My solution is to split the search into smaller areas to gain a high resolution and therefore less than 50 venues per search. That is why I decided to take the neighborhoods (Stadtteil) instead boroughs (Stadtbezirk). However, there were still two neighborhoods which have 50 venues. After a closer investigation I discovered that this wouldn’t cause a problem since the both neighborhoods are in the center of the city and surrounded by enough other neighborhoods which will find the rest of venues.

There is another problem with the possibility of getting uncovered search area around the city due to the unequal distribution of search points in the neighborhoods. Therefore, I increased the radius for the search which results a high number of duplicates of found venues. The duplication will be handled later before the model training.

After the restaurants were found, a Choropleth map was generated again to see the population density with the restaurant locations on top.



4 Machine Learning – Clustering Modeling

4.1 Choosing Algorithm

For Clustering, the algorithm – DBSCAN – was selected due to the following reasons:

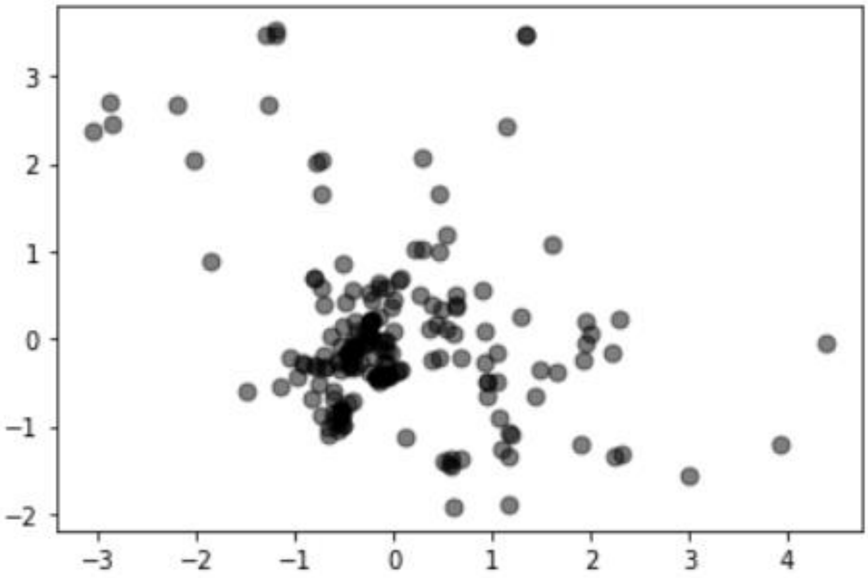
- The restaurant distribution in Dusseldorf doesn't follow any certain pattern. DBSCAN is especially fitted for arbitrarily shaped clusters.
- DBSCAN is robust to outliers. Unlike K-mean algorithm, DBSCAN can detect and label the outliers, which fits perfectly the purpose of this project to see where are the outliers and where densest clusters of restaurants, that the stakeholder should avoid.
- The number of clusters is not required to be specified.

In addition, the coordinate data – latitude and longitude – of the restaurants will be used for clustering.

4.2 Model Training

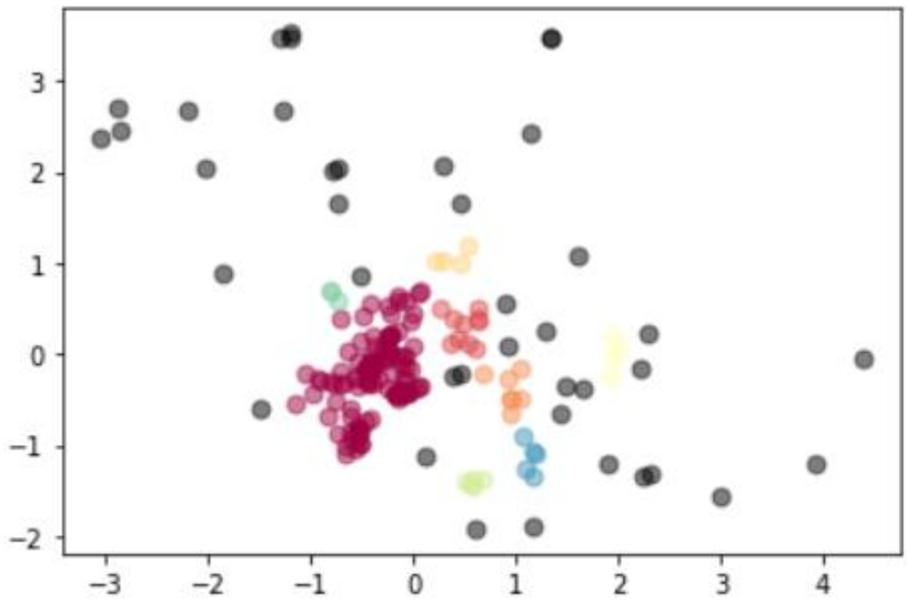
Before the actual training, the data must be cleaned at first, by eliminating the duplicates in the dataframe. This is very important. The algorithm DBSCAN has the parameter "min_samples", which defines the threshold how many samples should be at least within a radius (parameter epsilon). The duplicates will increase the number of data samples within the defined radius, so that this will affect our results. Therefore, I needed to drop the duplicates at first.

After data cleaning, the data was normalized for the model training. A scatter plot was generated to see the distribution of data after normalization.



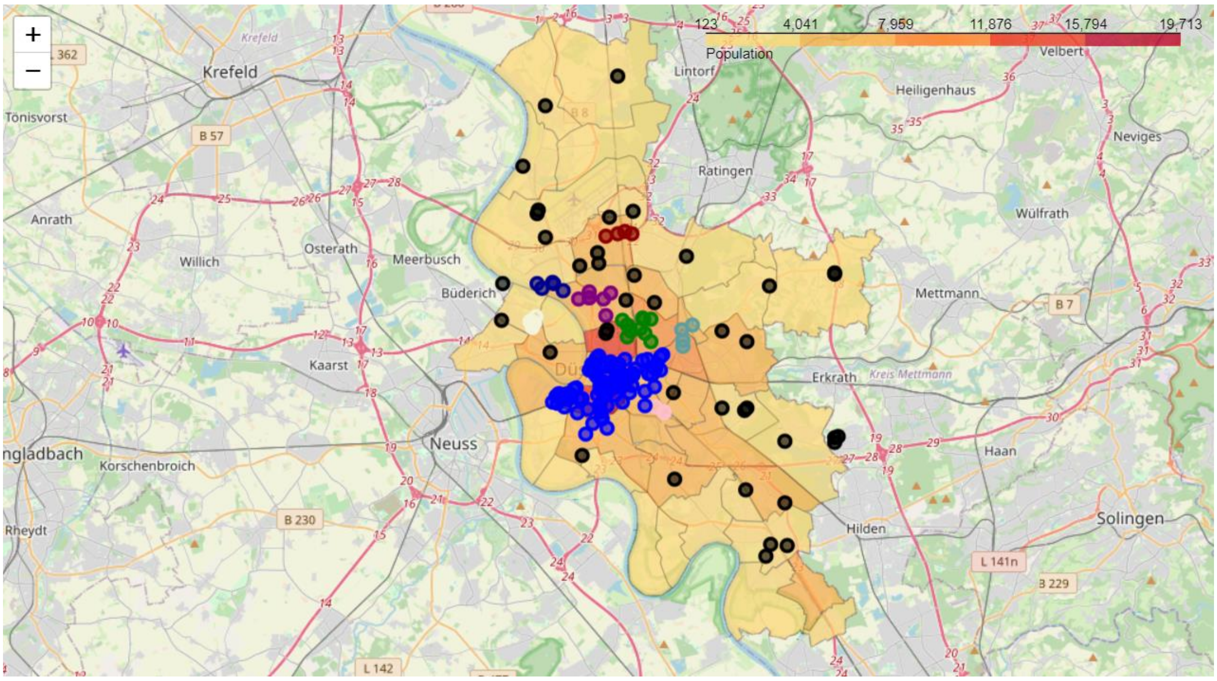
Finally, the model was trained with the normalized latitudes and longitudes of restaurants. There are two important parameters – epsilon and min_samples. Epsilon defines the radius that the algorithm will look after for each data sample. The min_samples defines the threshold how many samples should be at least within a radius (epsilon). After many trainings, the values – epsilon = 0.27 and min_samples = 4 – were applied from the best practice.

The algorithm returned a list of labels, which identify to which cluster each data sample should belong. A scatter plot was generated to display the modeling results. The restaurants which belong to a cluster are marked in a colorful color, each color represents a cluster. The outliers are marked in black. We could see there are eight clusters found by the model.



5 Results

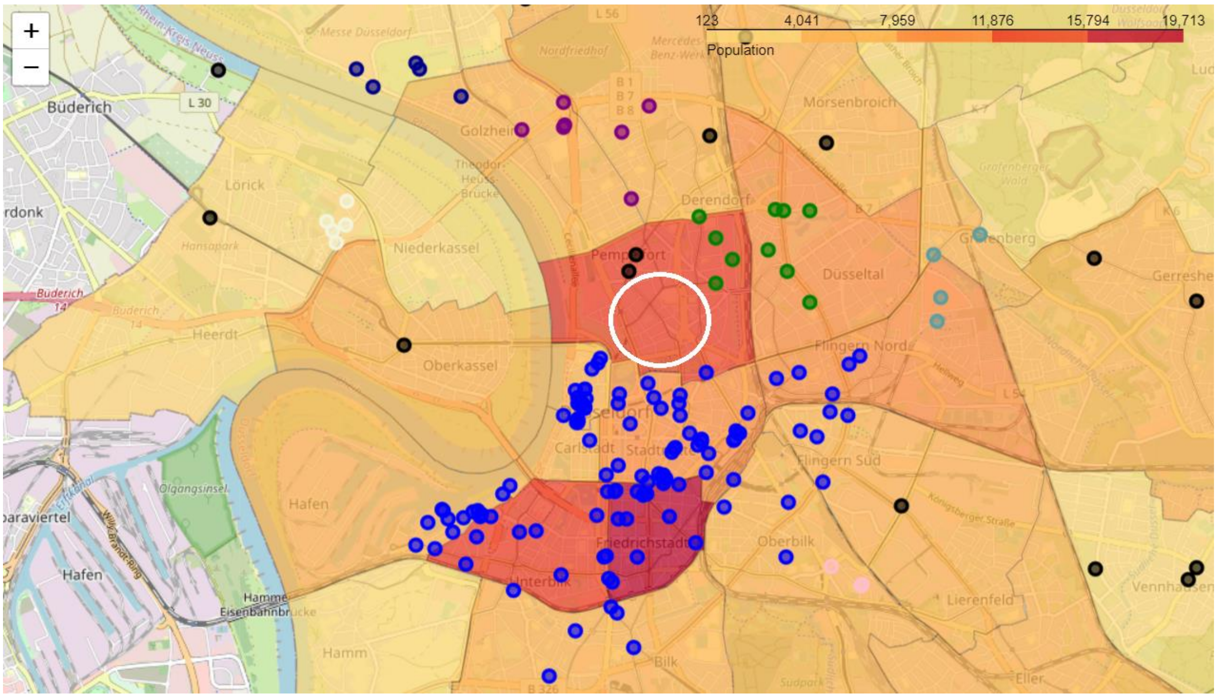
The labels which were returned by the model were added to the dataframe with all venues in Dusseldorf. Again, the Choropleth map was generated with the restaurants on top. This time I marked the restaurants with different colors to represent the clustering results from model training. The restaurants which belong to a cluster are marked in a color, each color represents a cluster. The outliers are marked in black.



In the project, around 200 restaurants were found in Dusseldorf across 50 neighborhoods. Our analysis by using DBSCAN algorithm segments those restaurants in 8 clusters based on the density of the restaurants. The other restaurants are marked as outliers due to the low density (no other restaurants in close vicinity).

It is not recommended to open a new restaurant in the area of any cluster.

Furthermore, our analysis shows the population density correlates with the number of restaurants in the neighborhoods. However, there are a few areas which has high population density but a low number of restaurants, for example the center of Pempelfort, which has one of the highest population density in Dusseldorf and is near to the city center. Therefore, we recommend this area to open a new restaurant (white circle in the following map).



6 Discussion

During the project I recognized that there is a limitation of the results number returned by the API service up to 50. My solution is to split the search into smaller area to gain a high resolution and therefore less than 50 venues per search. I decided to take the neighborhood location points instead boroughs. The problem with that is the possibility of getting uncovered search area around the city due to the unequal distribution of search points in the neighborhoods. Therefore, I increased the radius for the search which results a high number of duplications of found venues.

Since the algorithm "DBSCAN" will have the parameter min_samples, the duplicates will increase the numbers of data samples within the defined radius (epsilon), so that this will affect our results. Therefore, the duplicates need to be dropped.

For a bigger project it would be appropriate to use another service or level up the account to avoid the limitation.

7 Conclusion

The project aims to help a stakeholder to find a location for a new restaurant in Dusseldorf. The approach is to visualize the both information of population density of each neighborhood and restaurant clustering in Dusseldorf. Therefore, the stakeholder could easily find the best location which has a higher population density, but not within a restaurant cluster.

Based on the approach, we used the geographical data to gain the boundaries information between neighborhoods, the population and area data of each neighborhood to calculate the population density. This data is available to be downloaded from the website "Open Data Düsseldorf" as csv or json file.

During the analysis part, I successfully used the Foursquare API service to find restaurants in each neighborhood, and applied the machine learning algorithm "DBSCAN" to cluster the restaurants. With the help of the Folium visualization I could display clearly both information - population density and restaurant clustering - for the stakeholder on a map at the same time.

At the end we found an area - Pempelfort - recommended to open a new restaurant there since it has a high population density but without being in any restaurant cluster and it is close to the city center.