Battle of the Neighborhoods

APPLIED DATA SCIENCE CAPSTONE – DIEGO TALAVERA

Obtaining venues clusters

- If you need to shop items in different stores, you may want to find a place where all the stores are close to each others
- While in some cities malls are commonly found, some others might not have them, therefore the location of individual stores must be known
- A simple recommendation tool was developed to find the best option for the shopping on a specific set of different venue categories

Data acquisition and pre-processing

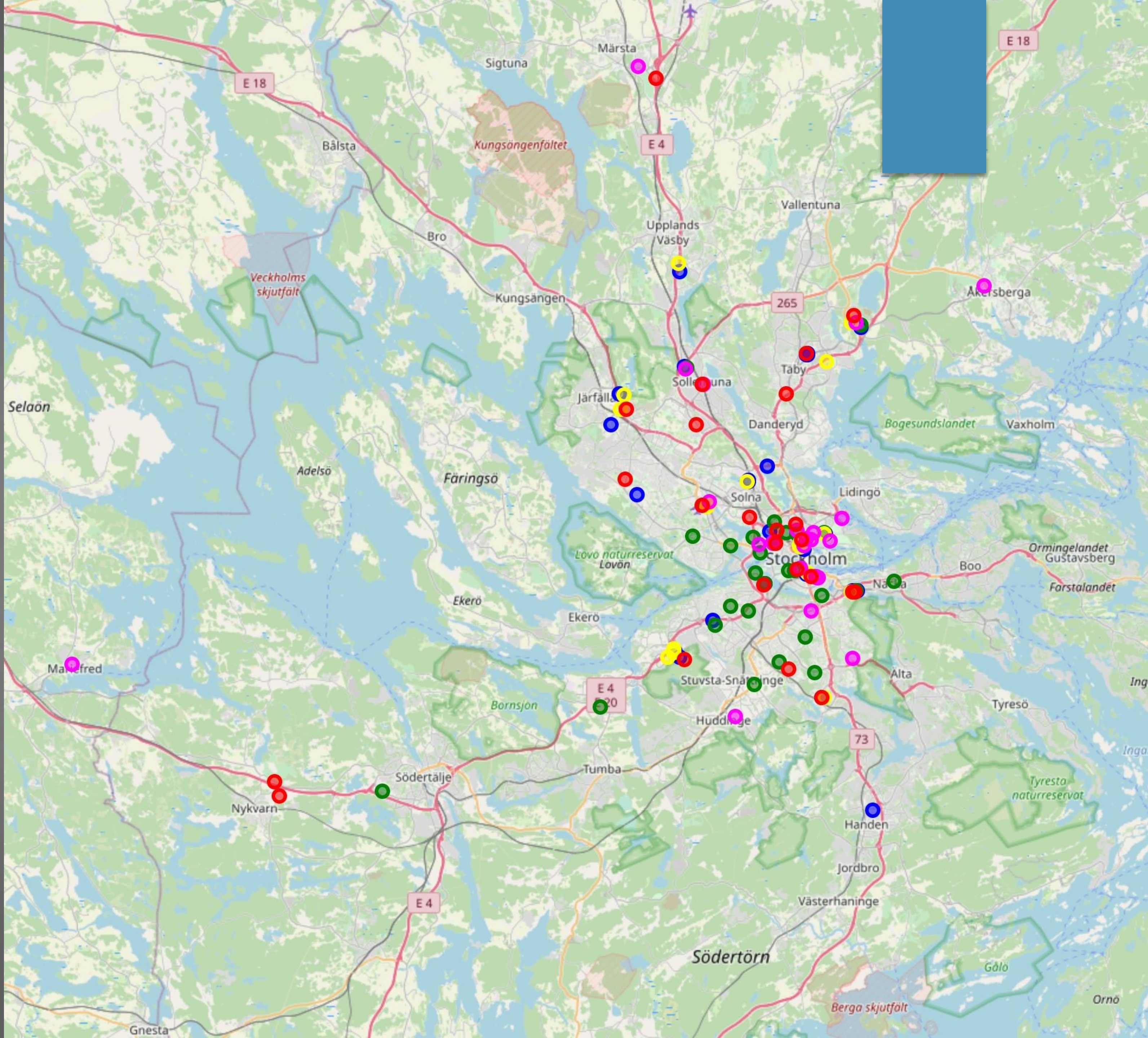
- Data was obtained with the foursquare API through a Jupyter notebook using python.
- After cleaning the raw data from the API a dataframe was constructed containing a few attributes as can be seen in the image in this slide

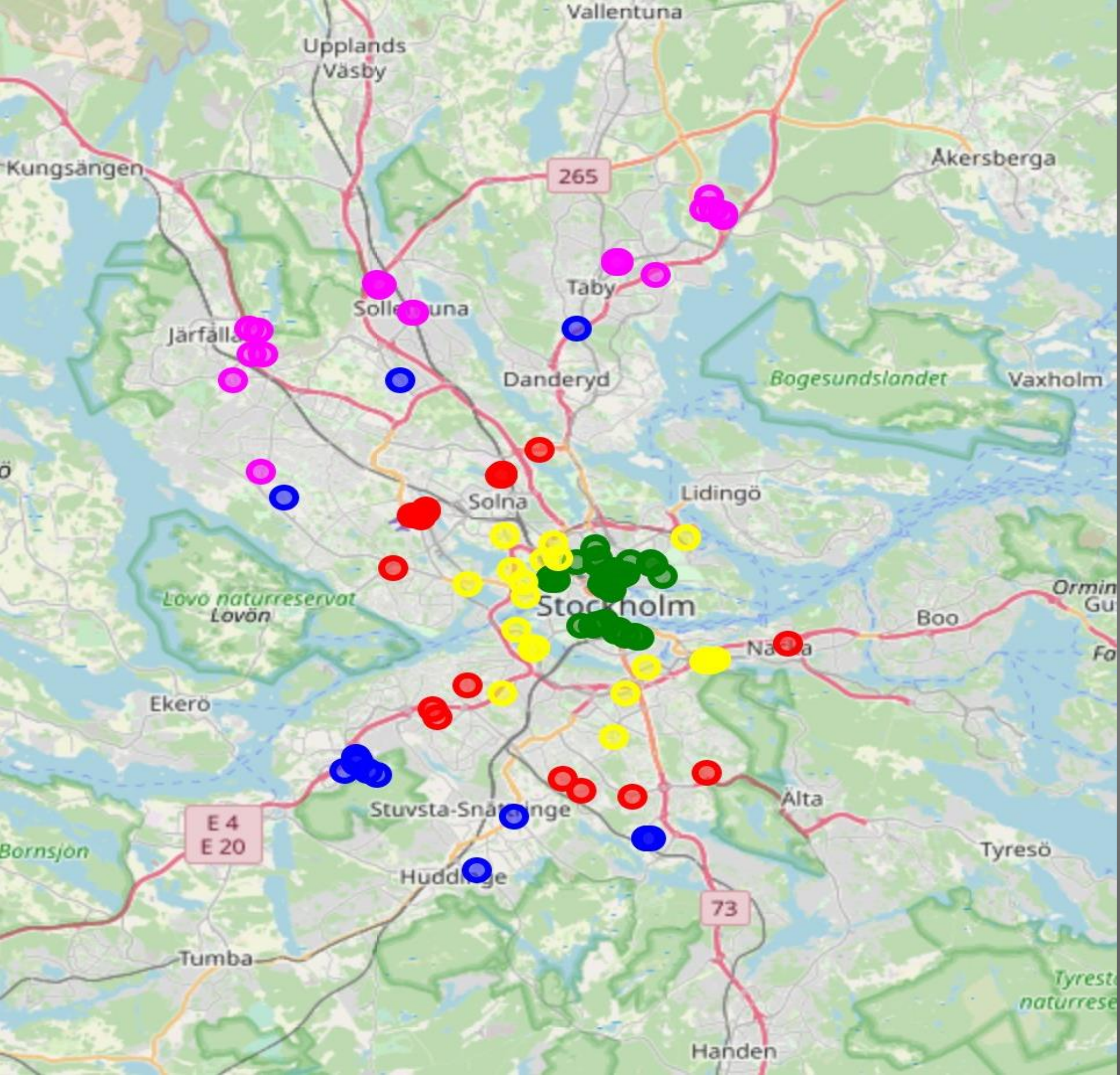
```python
Venues_df_test.reset_index(inplace=True)
Venues_df_test.drop(['level_0'], axis=1, inplace=True)
Venues_df_test.drop(['index'], axis=1, inplace=True)
Venues_df_test.head()
```

	Venue Name	Venue ID	Venue Latitude	Venue Longitude	Venue Category
0	Media Markt	4ecb821eb6346e2d39bfc4a9	59.330468	18.068199	Electronics
1	Kjell & Company	592d2d17a423627b6514e260	59.266931	17.923742	Electronics
2	Kjell & Company	4b64556bf964a52052ab2ae3	59.362560	17.874540	Electronics
3	Webhallen	54957924498eb41f84b3ebbe	59.242876	18.089323	Electronics
4	Webhallen	56e7d404498e53ae0a1f099d	59.309788	18.023252	Electronics

Venue visualization

- Venues where plotted using the library Folium in order to visualize them and proceed with any further cleaning and processing that was deemed necessary.
- Afterwards, venues farther away than 16.5km were dropped as they clearly don't form dense enough areas



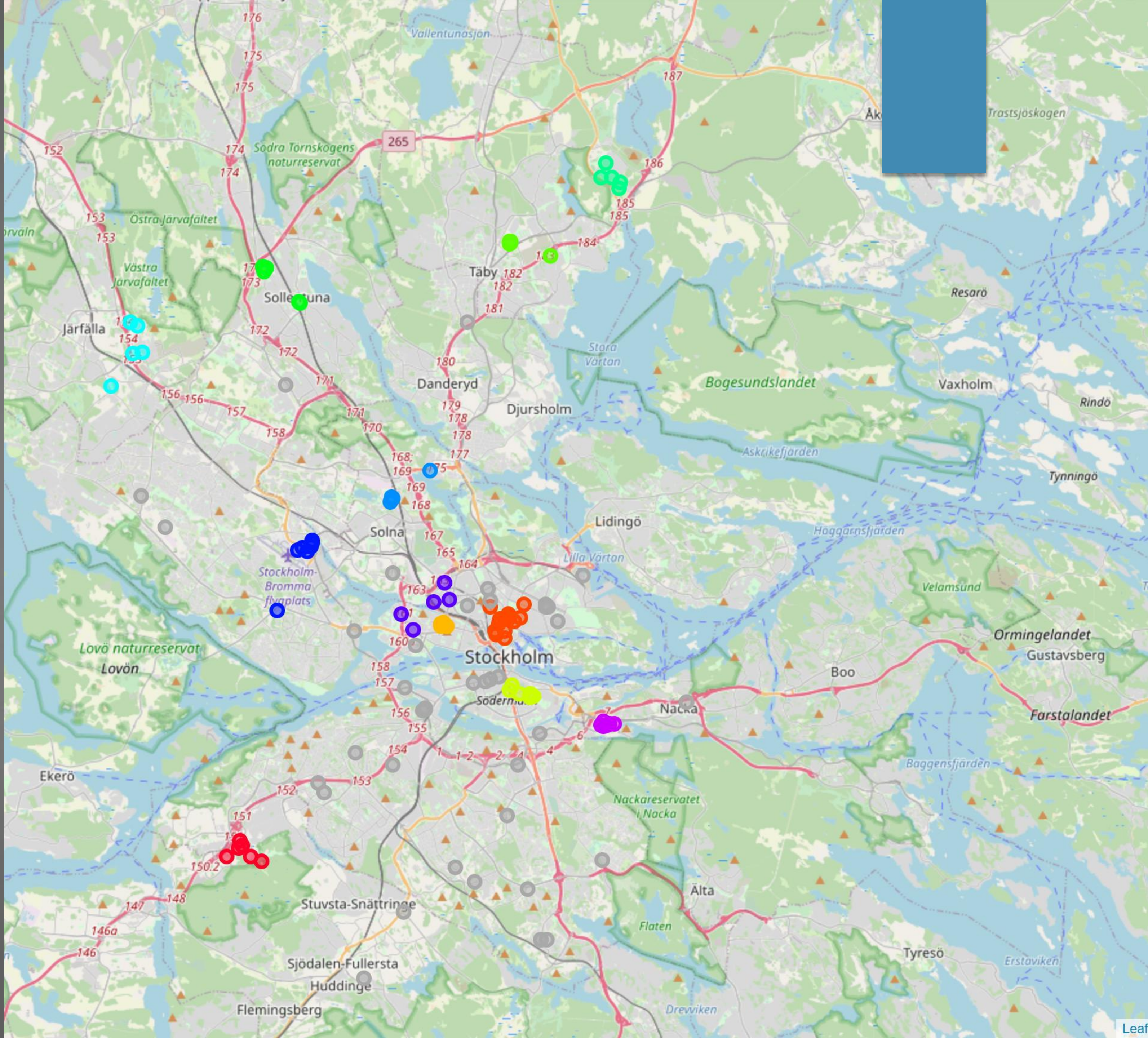


K-Means algorithm

- K-Mans was used to generate some first clusters or zones that helped visualize different regions around our starting point.
- As it can be seen, there are several places that could potentially have the venues we needed.

DBSCAN algorithm

- This algorithm was used to form clusters in the densest regions of the map, in order to find the best venue cluster. This resulted in a clustering as follows:



Useful Clusters Selection

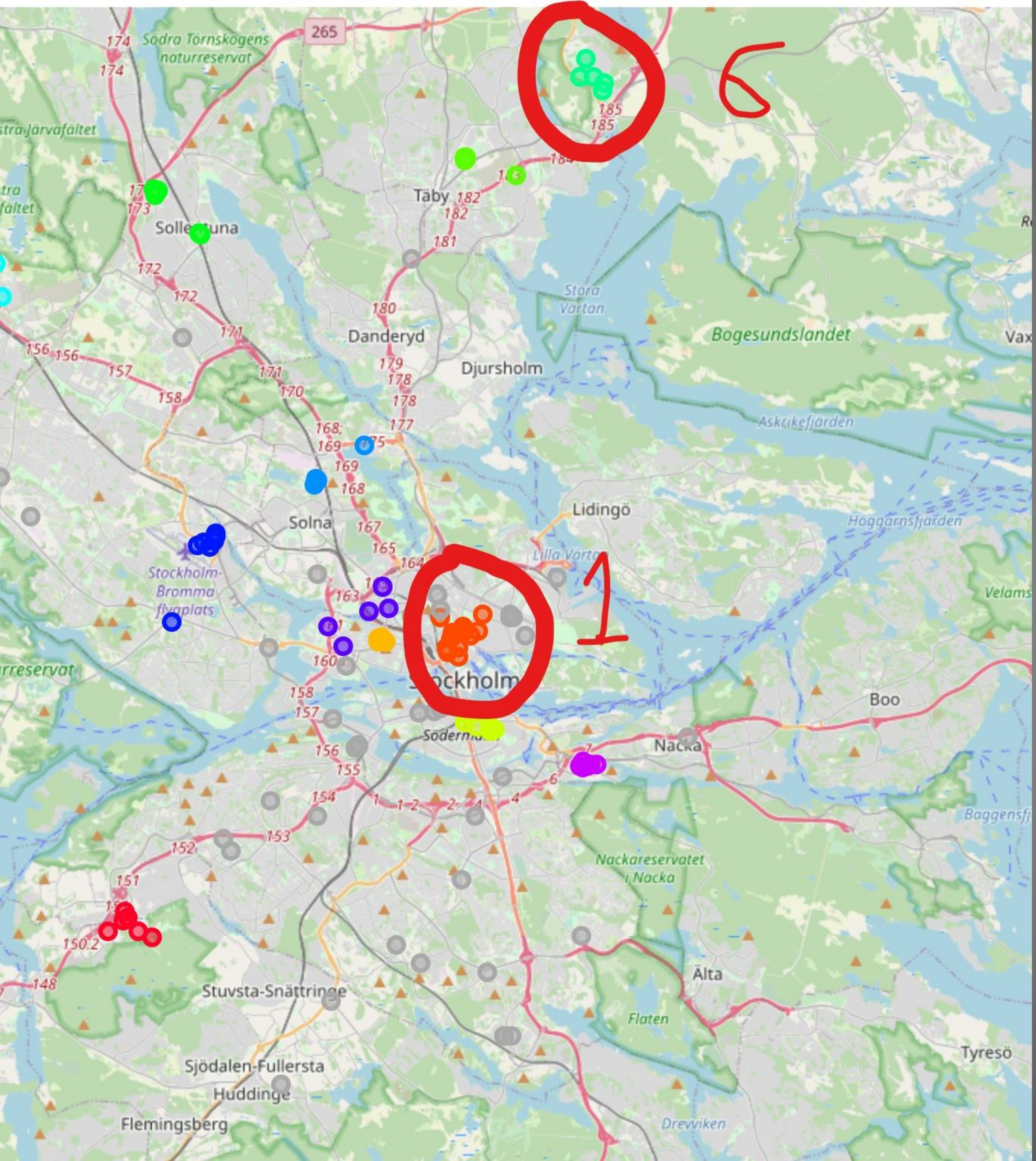
- Further analysis showed that only 2 clusters had the venue categories that we needed. The code used can be seen in the following image
- Note that even though we have 3 “clusters” that comply with the requirements, the one with label “-1” is the one containing the noise of the DBSCAN (Grey dots) therefore it was not further considered.

```
[53]: #Let's see how many clusters we have that contain all the categories we need
for i in Venues_df_test['DBSCAN label'].unique():
    if len(set(Venues_df_test.loc[Venues_df_test['DBSCAN label']==i, 'Venue Category']) &V_Cat_set)==5:
        print('DBSCAN label:{} \n Number of Categories: {} \n'.format(i, len(set(Venues_df_test.loc[Venues_df_test['DBSCAN label']==i, 'Venue Category']) &V_Cat_set)))

DBSCAN label:1.0
 Number of Categories: 5

DBSCAN label:-1.0
 Number of Categories: 5

DBSCAN label:6.0
 Number of Categories: 5
```



- These two clusters where further analysed to determine the best option. Note that the starting point is very close to cluster number 1.

Final Cluster Selection

- A simple scoring system was chosen as follows and the best cluster was selected:

And now let's choose a simple score to choose our cluster:

$$\left(0.7 \cdot R_{avg} + 0.3 \cdot 10 \cdot \left(1 - \frac{Dist_{avg}}{Dist_{Max}}\right)\right) = \left(0.7 \cdot R_{avg} + 3 \cdot \left(1 - \frac{Dist_{avg}}{Dist_{Max}}\right)\right)$$

A little bit more explanation will be given in the report!

```
]: def Score(Rating, DAvg, DMax):
       s=0.8*Rating+(2*(1-(DAvg)/(DMax)))
       return s
```

```
]: s1=Score(rating_1, dist_1, 50)
   s6=Score(rating_6, dist_6, 50)
   print('Score for cluster 1: {} \nScore for cluster 6: {}'.format(s1, s6))
```

```
Score for cluster 1: 7.18106217560012
Score for cluster 6: 6.887664317522728
```

Now we can conclude that cluster number 1 will be the best for our purposes, which as seen in the last map with the DBSCAN clusters, happens to be also the closest! Great!!

Conclusions and Recommendations

- The tool developed here can proved insight into better shopping options in order to optimize time used.
- The development of this project very helpful to increase the familiarity with several data analysis tools and methods.
- A different clustering algorithm can be used instead of DBSCAN, however it would increase the complexity of the code. This is described further in the report provided.
- Machine learning algorithms can be extremely useful in a wide variety of situations, even if such seem as mundane. The example studied in this project is a clear example of this and there are several ways to increase its complexity and applications. Some options include:
  - Consider traffic data
  - With some web scraping, price of the selected items can be considered in the scoring parameter.
  - Consider streets/highway distance instead of straight-line distances

Thanks for reading!