Socio-Economical Analysis of Oslo's Districts

Antonin Cajka

Introduction:

Data Science is a powerful field enabling individuals to analyse and predict various situations in the multidisciplinary areas of the real word. In this project, the author is leveraging the data science tools to analyse the regions of the capital city of Norway - Oslo, in order to discover popular areas within the city, suitable for opening a service establishment.

Data:

The data used in the report are collected from three main sources - the list of the districts via scraping a wikipedia page with the up to date updated list of Oslo's district, the development of the population growth within the district web scraped from the Norwegian Bank of Statistics portal, we also use the geographical data provider API Position Stack, to collect the coordinates for each of the districts, and finally, we leverage the Foursquare API to gather the data about present venues and their specifications for each neighbourhood.

List of boroughs of Oslo

From Wikipedia, the free encyclopedia

The 15 **boroughs of Oslo** were created on 1 January 2004. They each have an elected local council with limited responsibilities.[1]

Borough	Residents	Area	Number
Alna	49 801	13,7 km^2	12
Bjerke	33 422	7,7 km^2	9
Frogner	59 269	8,3 km^2	5
Gamle Oslo	58 671	7,5 km^2	1
Grorud	27 707	8,2 km^2	10
Grünerløkka	62 423	4,8 km^2	2
Nordre Aker	52 327	13,6 km^2	8
Nordstrand	52 459	16,9 km^2	14
Sagene	45 089	3,1 km^2	3
St. Hanshaugen	38 945	3,6 km^2	4
Stovner	33 316	8,2 km^2	11
Søndre Nordstrand	39 066	18,4 km^2	15
Ullern	34 596	9,4 km^2	6
Vestre Aker	50 157	16,6 km^2	7
Østensjø	50 806	12,2 km^2	13

Table 1: Data Scraped from Wikipedia

Folkemengden etter kjønn og alder (B) – *Begge kjønn, Alder i alt, Antall*

Endre utvalg av...

Bydel | År | Kjønn | Alder

År / Bydel	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020
Oslo i alt	521 886	529 846	538 411	548 617	560 484	575 475	586 860	599 230	613 285	623 966	634 463	647 676	658 390	666 759	673 469	681 071	693 494
Bydel Gamle Oslo	34 579	35 431	36 557	37 717	39 500	41 452	42 569	43 770	44 958	46 290	48 417	49 854	51 444	53 241	54 575	55 683	58 671
Bydel Grünerløkka	36 779	37 774	38 946	40 406	42 129	43 961	45 647	47 256	49 307	50 507	52 198	54 701	56 283	57 567	58 906	60 844	62 423
Bydel Sagene	28 816	29 654	30 414	31 428	32 394	33 631	34 286	35 115	35 990	37 053	38 637	39 918	41 566	42 442	43 131	43 801	45 089
Bydel St.Hanshaugen	26 728	27 619	28 287	29 082	30 144	31 550	32 254	33 137	34 109	34 982	35 630	36 218	37 263	37 849	38 109	38 400	38 945
Bydel Frogner	45 042	45 640	46 047	46 768	47 618	49 058	50 396	51 120	52 531	53 573	54 604	55 965	57 010	57 551	58 283	58 897	59 269
Bydel Ullern	26 977	27 179	27 599	28 331	28 898	29 839	30 250	30 744	31 275	31 443	31 656	32 124	32 757	33 257	33 463	33 760	34 569
Bydel Vestre Aker	40 424	40 587	40 878	41 302	42 042	42 756	43 457	44 320	45 186	45 715	46 444	47 024	47 885	48 229	48 605	49 153	50 157
Bydel Nordre Aker	40 235	41 060	41 656	42 692	43 843	45 024	46 287	47 433	48 432	48 413	48 720	49 337	49 781	50 224	50 724	51 558	52 327
Bydel Bjerke	24 256	24 448	24 606	25 530	26 229	26 863	27 632	28 226	29 090	29 617	30 327	30 502	30 937	31 510	31 973	32 500	33 422
Bydel Grorud	24 617	24 729	25 032	25 359	25 461	26 020	26 074	26 291	26 777	26 888	27 101	27 283	27 419	27 566	27 525	27 583	27 707
Bydel Stovner	28 109	28 445	28 656	28 936	29 351	29 651	29 746	30 178	30 554	30 752	31 340	31 669	32 153	32 527	32 850	32 909	33 316
Bydel Alna	43 612	44 151	44 482	44 820	45 114	46 029	46 603	47 025	47 786	48 062	48 307	48 770	49 224	49 282	49 358	49 457	49 801
Bydel Østensjø	42 484	42 681	43 547	44 036	44 399	45 042	45 577	46 244	47 164	47 806	48 714	49 133	49 821	49 973	49 968	50 427	50 806
Bydel Nordstrand	42 939	43 297	43 824	44 423	44 802	45 710	46 419	46 888	47 696	48 347	48 931	49 428	50 082	50 645	51 169	51 882	52 459
Bydel Søndre Nordstrand	33 088	33 501	33 863	34 444	34 980	35 258	35 768	35 843	36 304	36 659	37 054	37 913	38 405	38 672	38 925	38 803	39 066
Sentrum	495	538	603	643	861	905	918	893	963	970	958	1 063	1 170	1 146	1 114	1 382	1 471
Marka	1 596	1 614	1 924	1 581	1 585	1 615	1 638	1 627	1 637	1 624	1 608	1 598	1 596	1 587	1 644	1 633	1 610
Uten registrert adresse	1 110	1 498	1 490	1 119	1 134	1 111	1 339	3 120	3 526	5 265	3 817	5 176	3 594	3 491	3 147	2 399	2 386

Table 2: Data Scraped from Statistic Bank of Norway

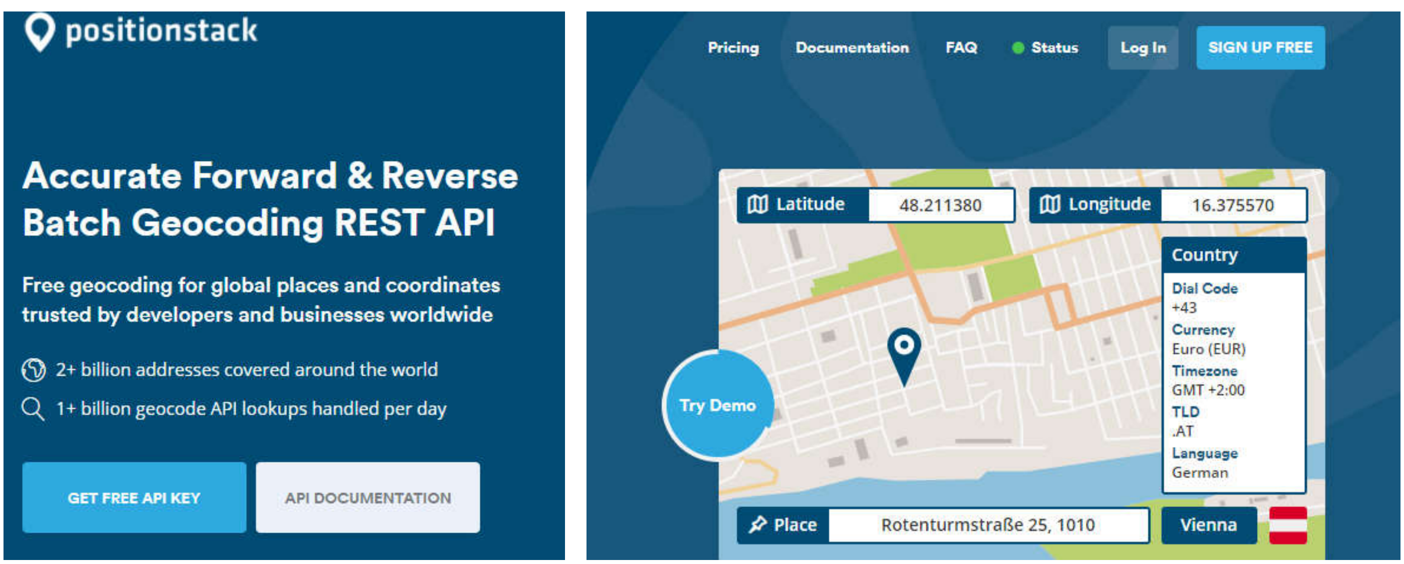


Figure 1: Positionstack API

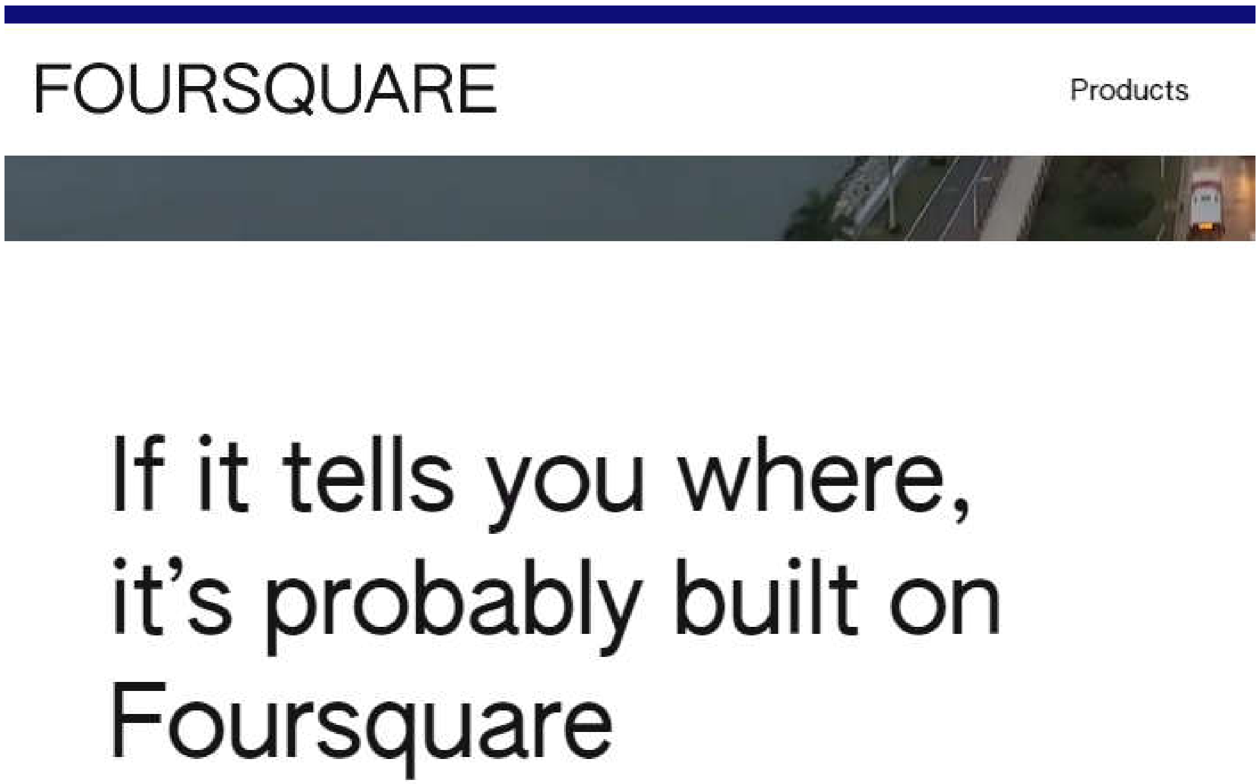


Figure 2: Foursquare API

Methodology:

Firstly, the data collection process was accomplished using web scraping tools as a panda web scraping html reader method. The data were saved into a data frame and organized using pandas data wrangling methods. At one point, also the regular expressions were used in order to ultimately clean the data before the analysis.

In order to understand the data, data consistency was checked multiple times against available standards, and a few times the data had to be adjusted manually - for example in case of the fail coordinates for a district. Data were checked for the missing values, and converted to a suitable form. Further the descriptive statistics of the data set were collected using the panda analysis methods, and data manipulation - subtraction within the dataset.

In order to visualize the data, tools from the Matplotlib python library were used for plotting the data in the form of lines plot and horizontal bar plot, and the python library Folium was used to visualize the districts over the map of the city.

Lastly, each district was assigned a set of venues collected via Foursquare API, and the data were clustered on similarity, using the k-means clustering algorithm

Importing the libraries

In [134]:
```
import numpy as np

import pandas as pd
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)

import json as json

import requests # library to handle requests
from  bs4 import BeautifulSoup #webscrapping library

from pandas import json_normalize


from sklearn.cluster import KMeans

import folium # map rendering library

print('Libraries imported.')
```

```
Libraries imported.
```

Figure 3: Libraries for the project

	Borough	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018
0	Gamle Oslo	34579	35431	36557	37717	39500	41452	42569	43770	44958	46290	48417	49854	51444	53241	54575
1	Grünerløkka	36779	37774	38946	40406	42129	43961	45647	47256	49307	50507	52198	54701	56283	57567	58906
output; double click to hide output				30414	31428	32394	33631	34286	35115	35990	37053	38637	39918	41566	42442	43131
3	St. Hanshaugen	26728	27619	28287	29082	30144	31550	32254	33137	34109	34982	35630	36218	37263	37849	38109
4	Frogner	45042	45640	46047	46768	47618	49058	50396	51120	52531	53573	54604	55965	57010	57551	58283
5	Ullern	26977	27179	27599	28331	28898	29839	30250	30744	31275	31443	31656	32124	32757	33257	33463
6	Vestre Aker	40424	40587	40878	41302	42042	42756	43457	44320	45186	45715	46444	47024	47885	48229	48605
7	Nordre Aker	40235	41060	41656	42692	43843	45024	46287	47433	48432	48413	48720	49337	49781	50224	50724
8	Bjerke	24256	24448	24606	25530	26229	26863	27632	28226	29090	29617	30327	30502	30937	31510	31973
9	Grorud	24617	24729	25032	25359	25461	26020	26074	26291	26777	26888	27101	27283	27419	27566	27525
10	Stovner	28109	28445	28656	28936	29351	29651	29746	30178	30554	30752	31340	31669	32153	32527	32850
11	Alna	43612	44151	44482	44820	45114	46029	46603	47025	47786	48062	48307	48770	49224	49282	49358
12	Østensjø	42484	42681	43547	44036	44399	45042	45577	46244	47164	47806	48714	49133	49821	49973	49968
13	Nordstrand	42939	43297	43824	44423	44802	45710	46419	46888	47696	48347	48931	49428	50082	50645	51169
14	Søndre Nordstrand	33088	33501	33863	34444	34980	35258	35768	35843	36304	36659	37054	37913	38405	38672	38925
15	Sentrum	495	538	603	643	861	905	918	893	963	970	958	1063	1170	1146	1114
16	Marka	1596	1614	1924	1581	1585	1615	1638	1627	1637	1624	1608	1598	1596	1587	1644
17	Oslo	1110	1498	1490	1119	1134	1111	1339	3120	3526	5265	3817	5176	3594	3491	3147

Figure 4: Cleaned and merged dataset for all districts, with its appropriate population values over the years

```
#loop trough all the columns and years that need to remove the locked space, define in list 'years'

y = 0
x = 0
all_years = ['2004', '2005', '2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013', '2014', '2

while x < 17:

        y = 0 #reset y
        while y < 18:

            df_tot[all_years[x]][y] = re.sub(r'\xa0', '', df_tot[all_years[x]][y])
            df_tot[all_years[x]][y]

            y = y + 1

        x = x + 1
```

```
#test of the dtype for values under the index [x]
result = df_tot['2004'][2] + df_tot['2004'][3]
result
```

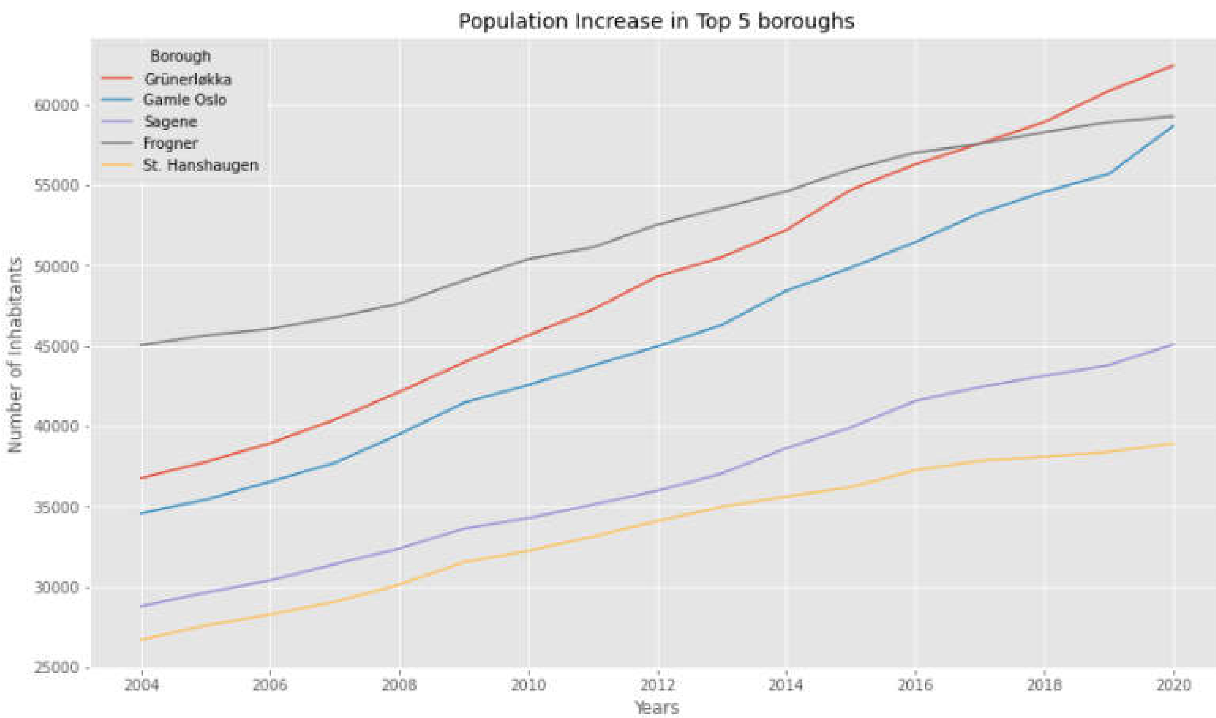
```
'28\xa081626\xa0728'
```

Figure 5: Regular Expressions: set of two while loops - to filter out the \xa0 fast space symbol from every value in the table - first over all values in the column, followed by each column

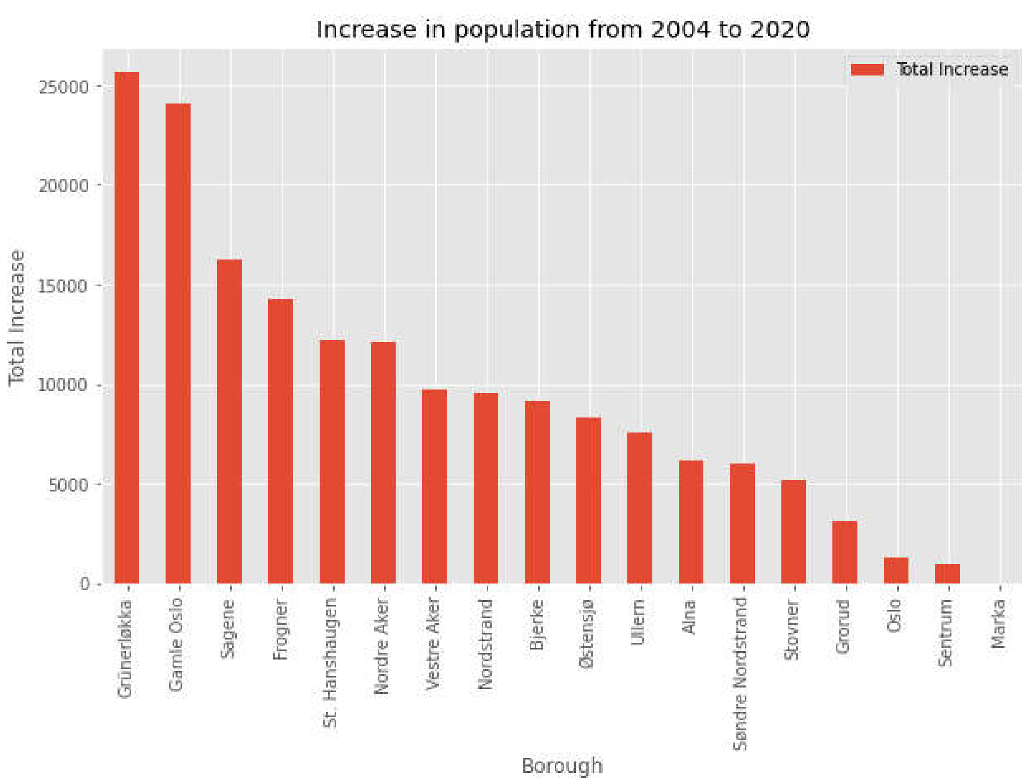
Plotting Section

Here starts the plotting of the population values over the period from 2004 to 2020 for each neighbourhood. We compare the trends for top5 districts, and bottom5 districts.

Lastly, we plot the overall population increase for each neighborhood over the years.



Graph 1: Population increase in Top 5 boroughs



Graph 2: Increase in population for all districts

From the plotted dependencies we can clearly state, that all of the districts experienced the population growth within the last 16 years, with the largest growth - the district of Grunerlokka, reaching over 25 000 increase in inhabitants.

Location Data API

In this section we attend to collect the location data of all districts. We will connect to the Position Stack open source API, and search the JSON file for the latitude and longitude information.

This is demonstrated in the capital city example, and further looped over all of the districts, while storing the information in various lists, just before appending the values to the overall dataframe.

```
import http.client, urllib.parse
url = 'http://api.positionstack.com/v1/forward'

params = urllib.parse.urlencode({
    'access_key': '######',
    'query': 'Oslo',
    'region': 'Oslo',
    'limit': 1,
    'output': 'json'
    })
resp = requests.get(url = url, params = params)
data = resp.json()

print((float(data['data'][0]['latitude'])))
print((float(data['data'][0]['longitude'])))
print((data['data'][0]['name']))

#print('')

print(data)
```

```
59.974453
10.735045
Oslo
{'data': [{'latitude': 59.974453, 'longitude': 10.735045,
```

Figure 6: Extraction of coordinates via Positionstack API

	Borough	Longitude	Latitude	2004	2005
0	Grünerløkka	10.757593	59.923269	36779.0	37774.0
1	Gamle Oslo	10.769130	59.907121	34579.0	35431.0
2	Sagene	10.753070	59.937650	28816.0	29654.0
3	Frogner	10.704321	59.921979	45042.0	45640.0
4	St. Hanshaugen	10.738939	59.929714	26728.0	27619.0
5	Nordre Aker	10.767097	59.951045	40235.0	41060.0

Figure 7: The merged data frame with coordinates.

Visualization of the Districts

Here we use the collected geographical information, and fetch it in a for loop via a folium plotting library to generate the map of the city with its all districts.

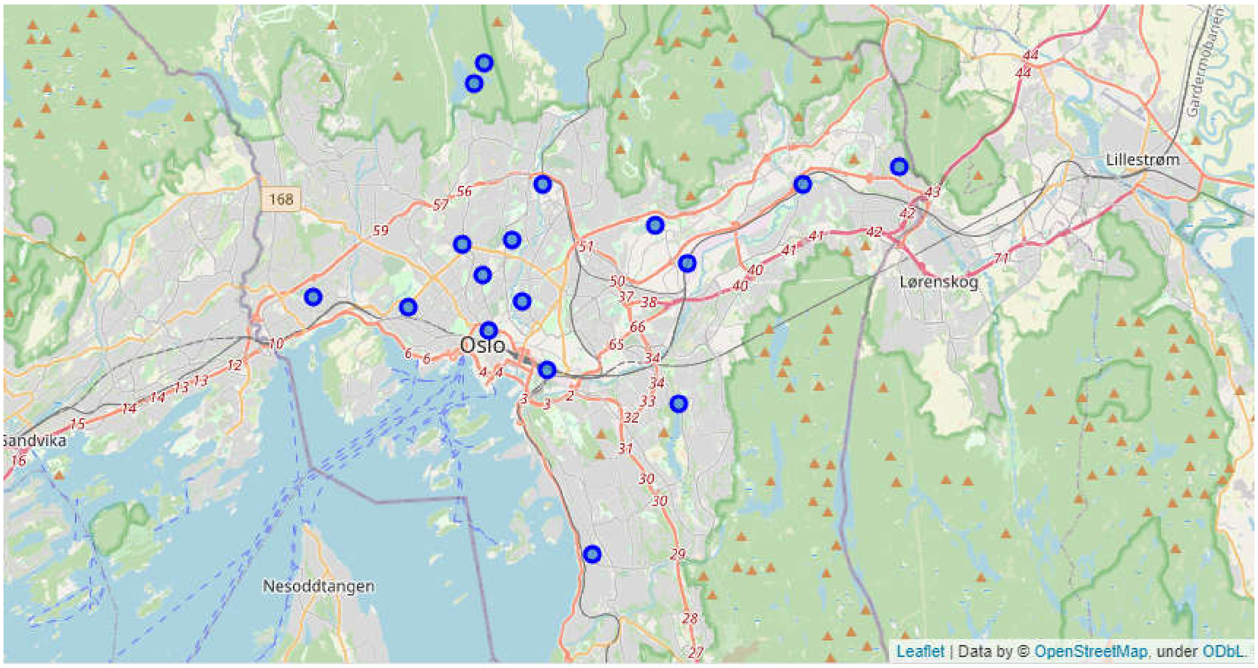


Figure 7: The districts of city Oslo

Foursquare API

Here we connect to the Foursquare API and collect the necessary data about each of the districts and it's venues. We use the radius of 1.5 km for each district! First, we demonstrate the data collection for one case, and later we automate the data collection and data storage via a for loop.

```
CLIENT_ID = '###' # your Foursquare ID
CLIENT_SECRET = '###' # your Foursquare Secret
VERSION = '20201212' # Foursquare API version
VERSION_OLD = '20040101'
LIMIT = 100 # limit of number of venues returned by Foursquare API

#radius = 500
radius_1 = 1500 # define radius

url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_s
    CLIENT_ID,
    CLIENT_SECRET,
    VERSION,
    neighborhood_latitude,
    neighborhood_longitude,
    radius_1,
    LIMIT)
url
```

	name	categories	lat	lng
0	Tim Wendelboe	Coffee Shop	59.923393	10.755494
1	Focacceria	Italian Restaurant	59.923632	10.757336
2	Grünerløkka Brygghus	Pub	59.925018	10.759250
3	Bon Lio	Spanish Restaurant	59.922238	10.761378
4	Godt Brød	Bakery	59.923942	10.759213

Figure 8: API connection and collected venues

We can clearly see that we have collected some data for each of the districts, with 151 unique categories, and overall our dataset contains 984 venues!

Clustering of the Districts

In this section, we attempt to perform the k-means clustering, depending on the variability of the venues of each district.

Firstly, we perform a technique of 'one hot encoding' in order to numerically represent our categorical variables. Later, we group them by the occurrence frequency in each neighborhood, and finally we model the data, using the 5 default clusters for our model.

```
# set number of clusters
kclusters = 5

oslo_grouped_clustering = oslo_grouped.drop('Neighborhood', 1)

# run k-means clustering
kmeans = KMeans(n_clusters=kclusters, random_state=0).fit(oslo_grouped_clustering)

# check cluster labels generated for each row in the dataframe
kmeans.labels_[0:10]
```

Figure 9: K-means algorithm clustering of districts according to the their venues

RESULTS

Finally, we can plot the cluster distribution over the map of the city, and analyse the clusters.

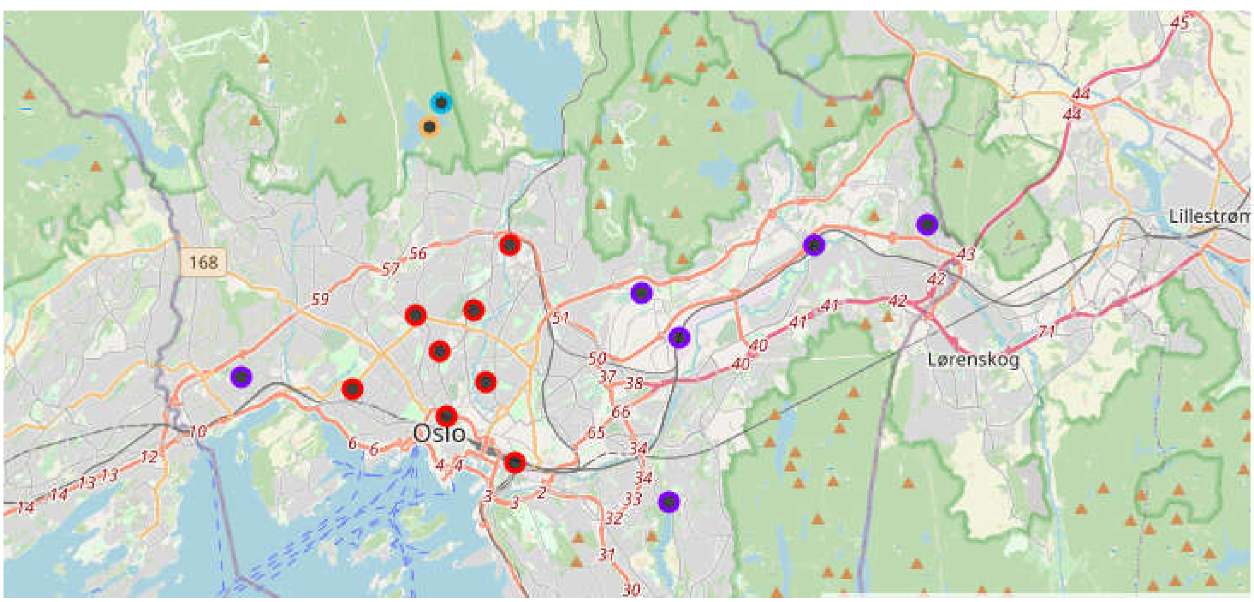


Figure 10: The districts of city Oslo clustered

DISCUSSION

We can clearly see that the cities within the city center have much faster growth of inhabitants, as well a wide selection of coffees, bars and restaurants - definitely a good spot to start a service business - we can call this cluster an enjoyment cluster.

Whereas the areas outside of the city center experience more of the family-like utilities - grocery stores, electro suppliencies - we can call this cluster a necessary cluster.

Our last clusters each contain only one district - this is due to the significant variance from the main two clusters - Marka, with it's access to a lake, and eg. the South Nordstrand with access to the beaches, and various playgrounds.

Based on the finding of this report, and tools used, there is a place for an improvement in the future: more data could be collected, in order to investigate more factors for the population growth, or the occurrence of specific venues in the area, the regression methods could be used, in order to determine the underlying correlation between the variables, as well as other forms of clustering methods - for example density- based clustering.

Conclusions

Overall we can see that the population is also increasing over the time more, the closer to the first two clusters one finds themselves! This indicates that the more densely populated areas also offer more of the entertainment venues, then the remote areas with less population growth.