Data Science Project: Battle of London Boroughs

Finding suitable locations to open a car wash in London, UK

Sergej Dzigajev (February 2021)

Contents

Introduction

Imagine for a moment that an investor is looking to open a car wash in London, United Kingdom. The investor crunched all of the numbers and on paper it is looking like a good business idea. However, the question now stands about the location. What would be the best place to open a new car wash to maximise return on investment? There is already thriving competition present in the city, and the investor wants to find out which one of Greater London's 32 Boroughs would be the optimum place.

To answer this question, the investor turns to the field of **Data Science** in order to explore all of the Boroughs from a deep analytical point of view. The main objective is to explore what each Borough contains, and what each Borough lacks. A perfect outcome would be to find an area that 1) isn't too crowded with competition, 2) has a lot of passing traffic and 3) relatively large population, and then to open up a new car wash in that area.

Data

To answer the main question of this report, data will be collected from several sources. First, a list of all 32 Greater London Boroughs (excluding city of London as it is predominantly a business district), including latitude and longitude positions and the population of each Borough which can be scraped from the Wikipedia page.

Next, it is necessary to identify the number and locations of existing car washes in London. Foursquare API is going to be used to help find and extract this information. The API allows one to search for businesses and venues around a given geographic point (like the Borough information from Wikipedia) and find out how many and what types of businesses are in that area. This data will then be extrapolated and prepared for further analysis.

In addition, vehicle traffic information (volume) of each Borough will be used from UK's Department of Transport. Data figures give the total volume of traffic across each local authority for the whole year. This will be used in order to assess how much 'potential' traffic there is in each Borough for a given car wash.

For further sources of data please refer to the References section.

Methodology

There are several algorithms that can be used to cluster location data. For this particular exercise, K-means clustering is going to be used to help group car wash locations and analyse the best potential candidate.

K-means aims to partition the observations into a predefined number of clusters (k) in which each point belongs to the cluster with the nearest mean. It starts by randomly selecting k centroids and assigning the points to the closest cluster, then it updates each centroid with the mean of all points in the cluster through iterating the data. This algorithm is convenient when you need to get a precise number of groups, and it's more appropriate for a small number of even clusters.

In order to define the right k, Elbow Method is going to be used by plotting the variance as a function of the number of clusters and picking the k that flattens the curve.

The main open source Python libraries that are used are the following:

Data Analysis and Manipulation

- Pandas
- SciPy
- NumPy

Data Visualisation

- Matplotlib
- Seaborn
- Folium

Machine Learning

- Scikit-Learn

Analysis

In [2]:

```
#---imports---
import pandas as pd
import re
from geopy.geocoders import Nominatim
import folium
import requests
import folium.plugins as plugins
from folium.plugins import HeatMap
from folium.plugins import HeatMapWithTime
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import preprocessing, cluster
from sklearn.cluster import KMeans
import numpy as np
import scipy
```

Scrape and parse London Boroughs from Wikipedia page using Pandas.

In [3]:

```
url='https://en.wikipedia.org/wiki/List_of_London_boroughs'

df=pd.read_html(url, header=0)[0]

df
```

Out[3]:

	Borough	Inner	Status	Local authority	Political control	Headquarters	Area (sq mi)	Populati (2019 e
0	Barking and Dagenham [note 1]	NaN	NaN	Barking and Dagenham London Borough Council	Labour	Town Hall, 1 Town Square	13.93	2129
1	Barnet	NaN	NaN	Barnet London Borough Council	Conservative	Barnet House, 2 Bristol Avenue, Colindale	33.49	3958
2	Bexley	NaN	NaN	Bexley London Borough Council	Conservative	Civic Offices, 2 Watling Street	23.38	2482
3	Brent	NaN	NaN	Brent London Borough Council	Labour	Brent Civic Centre, Engineers Way	16.70	3297
4	Bromley	NaN	NaN	Bromley London Borough Council	Conservative	Civic Centre, Stockwell Close	57.97	3323
5	Camden	NaN	NaN	Camden London Borough Council	Labour	Camden Town Hall, Judd Street	8.40	2700
6	Croydon	NaN	NaN	Croydon London Borough Council	Labour	Bernard Weatherill House, Mint Walk	33.41	3867
7	Ealing	NaN	NaN	Ealing London Borough Council	Labour	Perceval House, 14-16 Uxbridge Road	21.44	3418
8	Enfield	NaN	NaN	Enfield London Borough Council	Labour	Civic Centre, Silver Street	31.74	3337
9	Greenwich [note 2]	[note 3]	Royal	Greenwich London Borough Council	Labour	Woolwich Town Hall, Wellington Street	18.28	2879
10	Hackney	NaN	NaN	Hackney London Borough Council	Labour	Hackney Town Hall, Mare Street	7.36	2811

	Borough	Inner	Status	Local authority	Political control	Headquarters	Area (sq mi)	Populati (2019 e
11	Hammersmith and Fulham [note 4]	NaN	NaN	Hammersmith and Fulham London Borough Council	Labour	Town Hall, King Street	6.33	1851
12	Haringey	[note 3]	NaN	Haringey London Borough Council	Labour	Civic Centre, High Road	11.42	2686
13	Harrow	NaN	NaN	Harrow London Borough Council	Labour	Civic Centre, Station Road	19.49	2511
14	Havering	NaN	NaN	Havering London Borough Council	Conservative (council NOC)	Town Hall, Main Road	43.35	2595
15	Hillingdon	NaN	NaN	Hillingdon London Borough Council	Conservative	Civic Centre, High Street	44.67	3068
16	Hounslow	NaN	NaN	Hounslow London Borough Council	Labour	Hounslow House, 7 Bath Road	21.61	2715
17	Islington	NaN	NaN	Islington London Borough Council	Labour	Customer Centre, 222 Upper Street	5.74	2424
18	Kensington and Chelsea	NaN	Royal	Kensington and Chelsea London Borough Council	Conservative	The Town Hall, Hornton Street	4.68	1561
19	Kingston upon Thames	NaN	Royal	Kingston upon Thames London Borough Council	Liberal Democrat	Guildhall, High Street	14.38	1775
20	Lambeth	NaN	NaN	Lambeth London Borough Council	Labour	Lambeth Town Hall, Brixton Hill	10.36	3260
21	Lewisham	NaN	NaN	Lewisham London Borough Council	Labour	Town Hall, 1 Catford Road	13.57	3058
22	Merton	NaN	NaN	Merton London Borough Council	Labour	Civic Centre, London Road	14.52	2065
23	Newham	[note 3]	NaN	Newham London Borough Council	Labour	Newham Dockside, 1000 Dockside Road	13.98	3531

	Borough	Inner	Status	Local authority	Political control	Headquarters	Area (sq mi)	Populati (2019 e
24	Redbridge	NaN	NaN	Redbridge London Borough Council	Labour	Town Hall, 128-142 High Road	21.78	3052
25	Richmond upon Thames	NaN	NaN	Richmond upon Thames London Borough Council	Liberal Democrat	Civic Centre, 44 York Street	22.17	1980
26	Southwark	NaN	NaN	Southwark London Borough Council	Labour	160 Tooley Street	11.14	3188
27	Sutton	NaN	NaN	Sutton London Borough Council	Liberal Democrat	Civic Offices, St Nicholas Way	16.93	2063
28	Tower Hamlets	NaN	NaN	Tower Hamlets London Borough Council	Labour	Town Hall, Mulberry Place, 5 Clove Crescent	7.63	3247
29	Waltham Forest	NaN	NaN	Waltham Forest London Borough Council	Labour	Waltham Forest Town Hall, Forest Road	14.99	2769
30	Wandsworth	NaN	NaN	Wandsworth London Borough Council	Conservative	The Town Hall, Wandsworth High Street	13.23	3296
31	Westminster	NaN	City	Westminster City Council	Conservative	Westminster City Hall, 64	8.29	2613

Clean the data.

In [4]:

```
#copy the necessary columns
df = df[['Borough','Population (2019 est)[1]']].copy()
#delete the unnecessary notes in Borough rows
df['Borough'] = df['Borough'].str.replace(r"\[.*?\]","")
#rename the column for simplicity
df = df.rename(columns={'Population (2019 est)[1]': 'Population'})
#remove any white space before or after string for future manipulation
df['Borough'] = df['Borough'].str.strip()
df
```

Out[4]:

	Borough	Population
0	Barking and Dagenham	212906
1	Barnet	395896
2	Bexley	248287
3	Brent	329771
4	Bromley	332336
5	Camden	270029
6	Croydon	386710
7	Ealing	341806
8	Enfield	333794
9	Greenwich	287942
10	Hackney	281120
11	Hammersmith and Fulham	185143
12	Haringey	268647
13	Harrow	251160
14	Havering	259552
15	Hillingdon	306870
16	Hounslow	271523
17	Islington	242467
18	Kensington and Chelsea	156129
19	Kingston upon Thames	177507
20	Lambeth	326034
21	Lewisham	305842
22	Merton	206548
23	Newham	353134
24	Redbridge	305222
25	Richmond upon Thames	198019
26	Southwark	318830
27	Sutton	206349
28	Tower Hamlets	324745

	Borough	Population
29	Waltham Forest	276983
30	Wandsworth	329677
31	Westminster	261317

Fetch Latitude and Longitude using Nominatim

In [5]:

```
#Extract decimal lat long using Nominatim

#add London prefix to make sure it's the correct Address (similar names of places in US or
df['Borough'] = df['Borough'].astype(str) + ', London'
df['Latitude'] = None
df['Longitude'] = None
locator = Nominatim(user_agent="myGeocoder")
for i in range(len(df)):
    location = locator.geocode(df.loc[i,'Borough'])
    df.loc[i,'Latitude'] = location.latitude
    df.loc[i,'Longitude'] = location.longitude
#remove London prefix
df['Borough'] = df['Borough'].str.replace(', London', '')
df
```

Out[5]:

	Borough	Population	Latitude	Longitude
0	Barking and Dagenham	212906	51.5541	0.150504
1	Barnet	395896	51.6531	-0.200226
2	Bexley	248287	51.4417	0.150488
3	Brent	329771	51.5638	-0.27576
4	Bromley	332336	51.4028	0.0148142
5	Camden	270029	51.5423	-0.13956
6	Croydon	386710	51.3713	-0.101957
7	Ealing	341806	51.5127	-0.305195
8	Enfield	333794	51.6521	-0.0810175
9	Greenwich	287942	51.4821	-0.0045417
10	Hackney	281120	51.5432	-0.0493621
11	Hammersmith and Fulham	185143	51.492	-0.22364
12	Haringey	268647	51.6015	-0.111782
13	Harrow	251160	51.5968	-0.337316
14	Havering	259552	51.5444	-0.144307
15	Hillingdon	306870	51.5425	-0.448335
16	Hounslow	271523	51.4686	-0.361347
17	Islington	242467	51.5384	-0.0999051
18	Kensington and Chelsea	156129	51.4985	-0.199043
19	Kingston upon Thames	177507	51.4096	-0.306262
20	Lambeth	326034	51.5013	-0.117287
21	Lewisham	305842	51.4624	-0.0101331
22	Merton	206548	51.4109	-0.188097
23	Newham	353134	51.53	0.029318
24	Redbridge	305222	51.5763	0.0454097
25	Richmond upon Thames	198019	51.4406	-0.307639

	Borough	Population	Latitude	Longitude
26	Southwark	318830	51.5029	-0.103458
27	Sutton	206349	51.3575	-0.173627
28	Tower Hamlets	324745	51.5256	-0.0335853
29	Waltham Forest	276983	51.5982	-0.0178367
30	Wandsworth	329677	51.457	-0.193261
31	Westminster	261317	51.5004	-0.12654

Use geopy library to get the latitude and longitude values of London to build a map.

In [6]:

```
#Use geopy library to get the latitude and longitude values of London to build a map.
address = 'London, United Kingdom'
\
geolocator = Nominatim(user_agent="ny_explorer")
location = geolocator.geocode(address)
lon_lat = location.latitude
lon_long = location.longitude
print('The geograpical coordinate of London, UK are {}, {}.'.format(lon_lat, lon_long))
```

```
The geograpical coordinate of London, UK are 51.5073219, -0.1276474.
```

Create a Folium Map with London Boroughs

In [7]:

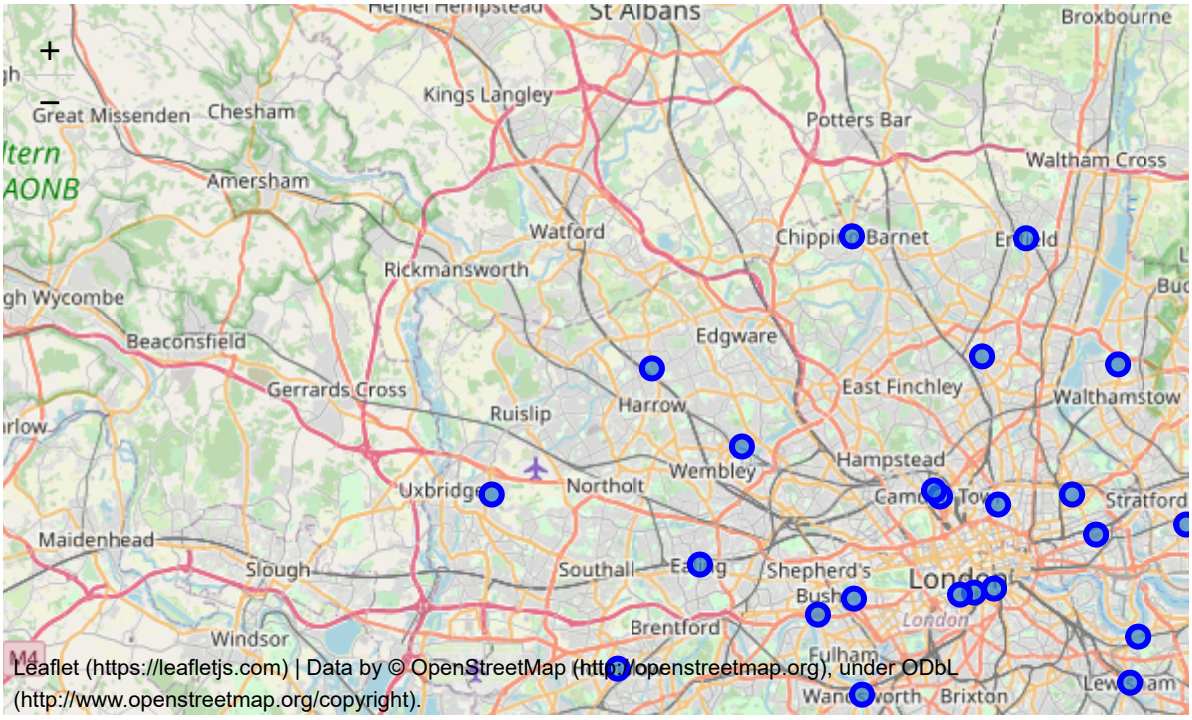
```
#Create a map of London with neighborhoods superimposed on top.

# create map of London using latitude and longitude values

map_london = folium.Map(location=[lon_lat, lon_long], zoom_start=10)

# add markers to map
for lat, lng, borough in zip(df['Latitude'], df['Longitude'], df['Borough']):
    label = '{}'.format(borough)
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius=5,
        popup=label,
        color='blue',
        fill=True,
        fill_color='#3186cc',
        fill_opacity=0.7,
        parse_html=False).add_to(map_london)
map_london
```

Out[7]:



Set-up Foursquare

In [8]:

```
CLIENT_ID = 'T4T0MTD4QWKJ5QO15HL0HCS2M2XY0BKNSOIBTLNWQXJIJOGL' # your Foursquare ID
CLIENT_SECRET = 'NF2GNFZYFARK2HV1ZYHQRMN0MISPUZFDEJWII4YJNJ2R3DQM' # your Foursquare Secret
VERSION = '20180605' # Foursquare API version
LIMIT = 200
RADIUS = 15000
category_id = '4f04ae1f2fb6e1c99f3db0ba'

print('Your credentails:')
print('CLIENT_ID: ' + CLIENT_ID)
print('CLIENT_SECRET:' + CLIENT_SECRET)
```

```
Your credentails:
CLIENT_ID: T4T0MTD4QWKJ5QO15HL0HCS2M2XY0BKNSOIBTLNWQXJIJOGL
CLIENT_SECRET:NF2GNFZYFARK2HV1ZYHQRMN0MISPUZFDEJWII4YJNJ2R3DQM
```

Method to extract car wash data

In [9]:

```
#collect car wash data
def getNearbyCarWash(categoryId, boroughs, latitudes, longitudes, radius=7500):

    carwash_list=[]
    for borough, lat, lng in zip(boroughs, latitudes, longitudes):

        # create the API request URL
        url = 'https://api.foursquare.com/v2/venues/explore?categoryId={}&client_id={}&clie
            category_id,
            CLIENT_ID,
            CLIENT_SECRET,
            VERSION,
            lat,
            lng,
            radius,
            LIMIT)

        # make the GET request
        results = requests.get(url).json()['response']['groups'][0]['items']

        # return only relevant information for each nearby borough
        carwash_list.append([(
            borough,
            lat,
            lng,
            v['venue']['name'],
            v['venue']['location']['lat'],
            v['venue']['location']['lng'],
            v['venue']['location']['distance'],
            v['venue']['categories'][0]['name']
        ) for v in results])

    nearby_carwash_df = pd.DataFrame([item for venue_list in carwash_list for item in venue
    nearby_carwash_df.columns = [
                  'Borough Name',
                  'Latitude',
                  'Longitude',
                  'Venue',
                  'Venue Latitude',
                  'Venue Longitude',
                  'Distance from Borough city centre',
                  'Venue Category']
    print('Finished')
    return(nearby_carwash_df)
```

In [10]:

```
carwash_venues_df = getNearbyCarWash(category_id, boroughs=df['Borough'], latitudes=df['Lat
```

```
Finished
```

In [11]:

```
#check the number of car washes
carwash_venues_df.shape
```

Out[11]:

```
(368, 8)
```

Since we set our radius to 7.5km around each Borough centre, there will be duplicates. Remove them:

In [12]:

```
# Examine all duplicate rows based on one column
duplicateRowsDF = carwash_venues_df[carwash_venues_df.duplicated(['Venue Latitude'])]
print("Duplicate Rows based on a single column are:", duplicateRowsDF, sep='\n')
```

```
Duplicate Rows based on a single column are:
    Borough Name   Latitude  Longitude                    Venue  \
31         Brent  51.563826  -0.275760           Hand Car Wash
34         Brent  51.563826  -0.275760           Hand Car Wash
45       Bromley  51.402805   0.014814       Olley's Posh Wosh
47        Camden  51.542305  -0.139560    The American Car Wash
52        Camden  51.542305  -0.139560                   Shell
..           ...        ...        ...                     ...
363  Westminster  51.500444  -0.126540    The American Car Wash
364  Westminster  51.500444  -0.126540  H2O Car Wash & Valeting
365  Westminster  51.500444  -0.126540       Starhand Car Wash
366  Westminster  51.500444  -0.126540         5 Star Car Wash
367  Westminster  51.500444  -0.126540                      BP

     Venue Latitude  Venue Longitude  Distance from Borough city centre  \
31         51.613247        -0.250251                                5777
34         51.602029        -0.193927                                7080
45         51.399753         0.113250                                6844
47         51.539175        -0.188431                                3401
52         51.565697        -0.225631                                6501
..               ...              ...                                 ...
363        51.539175        -0.188431                                6080
364        51.506499        -0.220130                                6519
365        51.550317        -0.075813                                6570
366        51.502649        -0.223355                                6713
367        51.437259        -0.161596                                7441

    Venue Category
31        Car Wash
34        Car Wash
45        Car Wash
47        Car Wash
52        Car Wash
..             ...
363       Car Wash
364       Car Wash
365       Car Wash
366       Car Wash
367    Gas Station

[269 rows x 8 columns]
```

In [13]:

```
#drop these duplicates
carwash_venues_df = carwash_venues_df.drop_duplicates(['Venue Latitude','Venue Longitude'])
carwash_venues_df
```

Out[13]:

	Borough Name	Latitude	Longitude	Venue	Venue Latitude	Venue Longitude	Distance from Borough city centre	Venue Category
0	Barking and Dagenham	51.554117	0.150504	BP	51.549951	0.161963	918	Car Wash
1	Barking and Dagenham	51.554117	0.150504	BP	51.565010	0.203570	3867	Car Wash
2	Barking and Dagenham	51.554117	0.150504	Waves Hand Car Wash	51.566331	0.190599	3089	Car Wash
3	Barking and Dagenham	51.554117	0.150504	Ship And Shovel Carwash	51.532009	0.117131	3375	Car Wash
4	Barking and Dagenham	51.554117	0.150504	IMO Car Wash	51.535007	0.103300	3899	Car Wash
...	...	...	...	...	...	...	...	...
222	Kingston upon Thames	51.409627	-0.306262	TheEcoSmart - Car Valeting East London	51.411908	-0.208150	6816	Car Wash
248	Merton	51.410870	-0.188097	IMO Car Wash	51.413672	-0.186490	331	Car Wash
255	Merton	51.410870	-0.188097	BP	51.364920	-0.214090	5424	Car Wash
272	Redbridge	51.576320	0.045410	Chigwell Hand Car Wash	51.597480	0.039001	2396	Car Wash
278	Redbridge	51.576320	0.045410	SupaShine Chigwell	51.628731	0.062752	5956	Car Wash

99 rows × 8 columns

Analyse the final extracted data

In [14]:

```
#check how many venues were returned for each neighborhood
carwash_venues_df.groupby('Borough Name').count()
```

Out[14]:

Borough Name	Latitude	Longitude	Venue	Venue Latitude	Venue Longitude	Distance from Borough city centre	Venue Category
Barking and Dagenham	10	10	10	10	10	10	10
Barnet	6	6	6	6	6	6	6
Bexley	7	7	7	7	7	7	7
Brent	11	11	11	11	11	11	11
Bromley	9	9	9	9	9	9	9
Camden	9	9	9	9	9	9	9
Croydon	5	5	5	5	5	5	5
Ealing	6	6	6	6	6	6	6
Enfield	6	6	6	6	6	6	6
Greenwich	5	5	5	5	5	5	5
Hackney	4	4	4	4	4	4	4
Hammersmith and Fulham	4	4	4	4	4	4	4
Harrow	2	2	2	2	2	2	2
Hillingdon	4	4	4	4	4	4	4
Hounslow	3	3	3	3	3	3	3
Kingston upon Thames	4	4	4	4	4	4	4
Merton	2	2	2	2	2	2	2
Redbridge	2	2	2	2	2	2	2

In [15]:

```
# Check the categories of venues that were returned for each borough

carwash_venues_df.groupby(['Venue Category']).count()
```

Out[15]:

Venue Category	Borough Name	Latitude	Longitude	Venue	Venue Latitude	Venue Longitude	Distance from Borough city centre
Auto Garage	1	1	1	1	1	1	1
Car Wash	94	94	94	94	94	94	94
Gas Station	3	3	3	3	3	3	3
Miscellaneous Shop	1	1	1	1	1	1	1

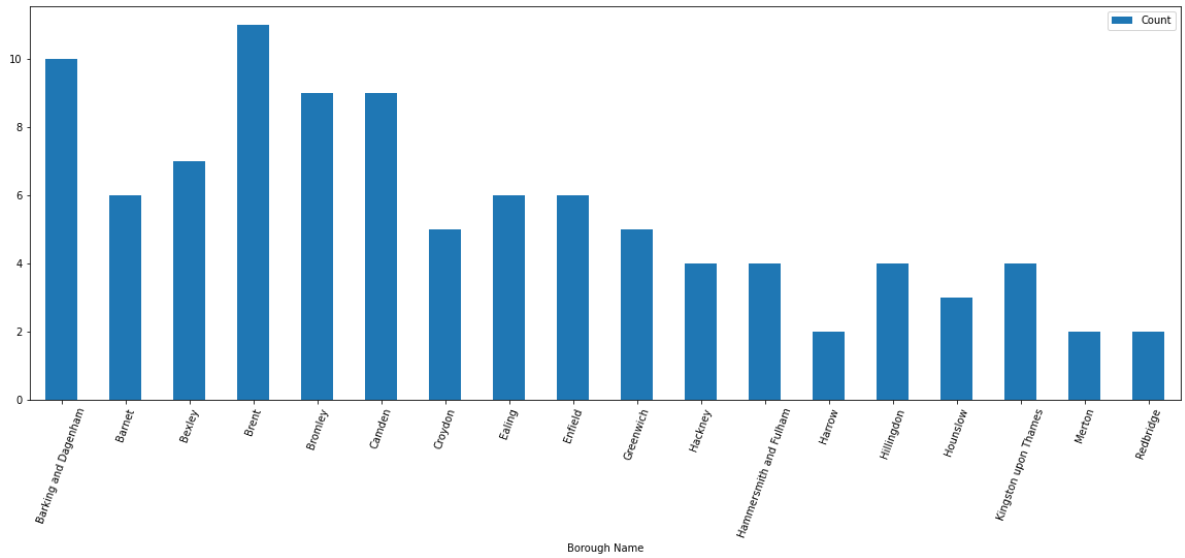
In [16]:

```
# Sort the data by frequency
sorted_venues = carwash_venues_df.groupby(['Borough Name']).size().reset_index(name='Count'
sorted_venues.columns=['Borough Name','Count']

sorted_venues.plot(x='Borough Name', y='Count', kind='bar', figsize=(20,7), rot=70)
```

Out[16]:

```
<AxesSubplot:xlabel='Borough Name'>
```

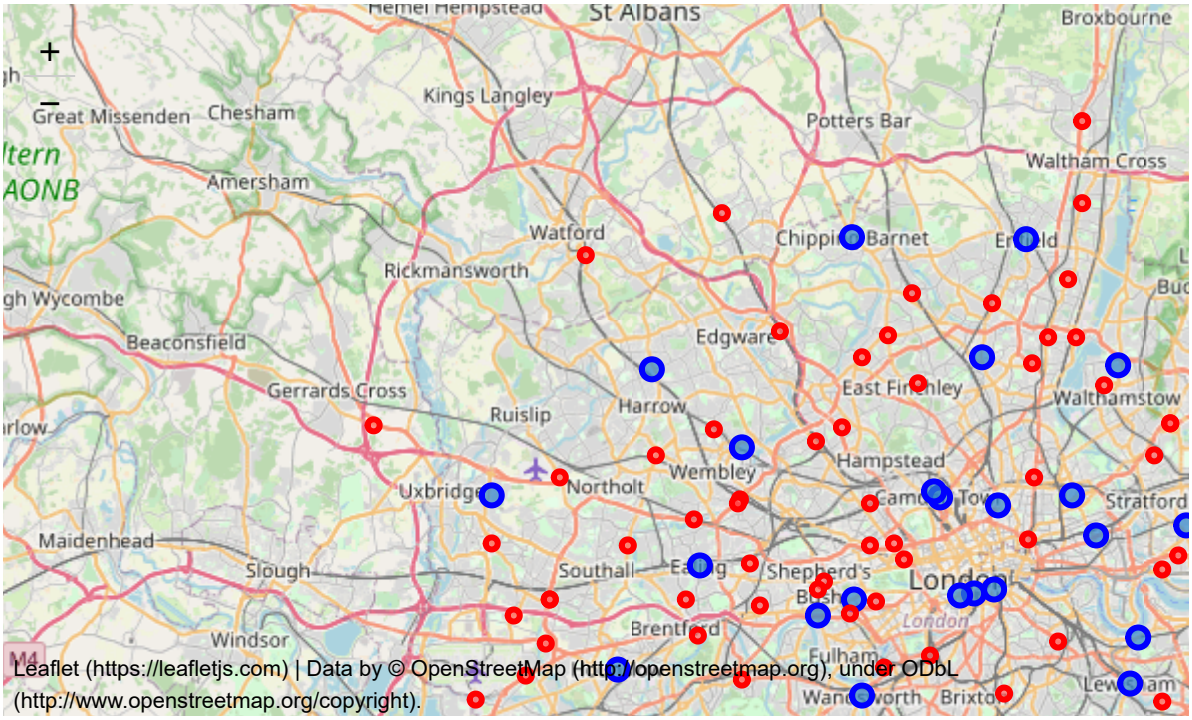


Plot existing Car Washes on the map

In [17]:

```
# add car washes to map
for lat, lng, carwash in zip(carwash_venues_df['Venue Latitude'], carwash_venues_df['Venue
    label = '{}'.format(carwash)
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius=3,
        popup=label,
        color='red',
        fill=True,
        fill_color='#ff6666',
        fill_opacity=0.7,
        parse_html=False).add_to(map_london)
map_london
```

Out[17]:

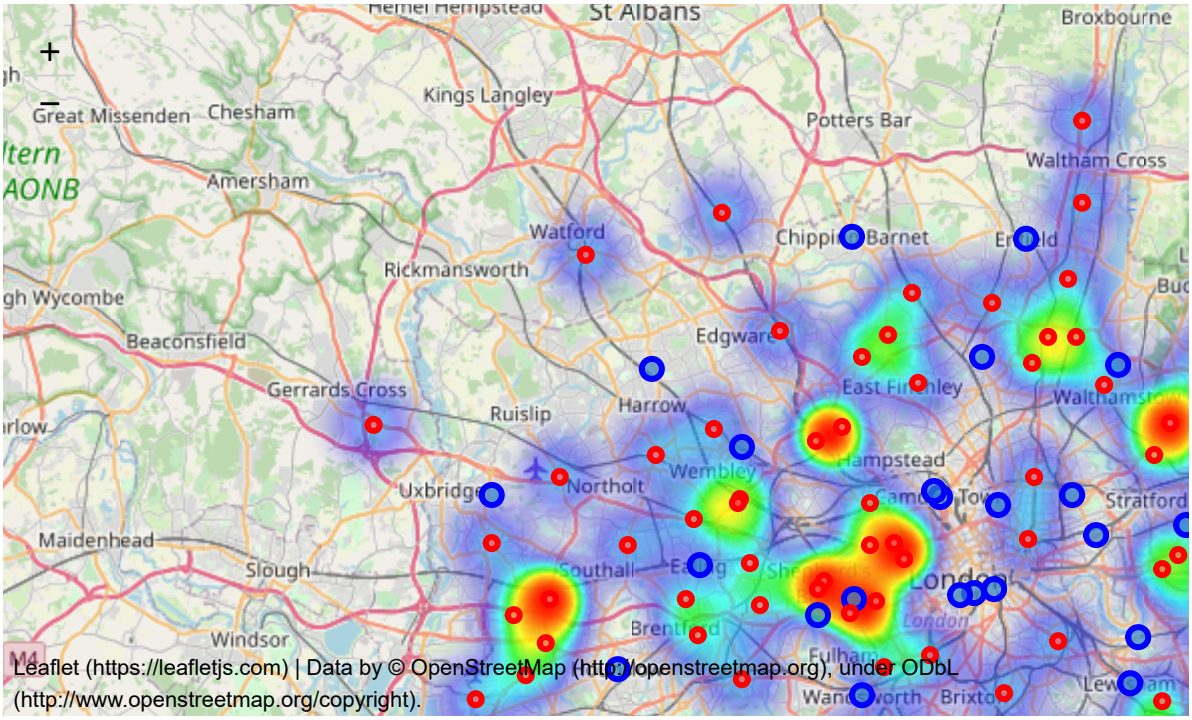


Produce a Heat Map

In [18]:

```
HeatMap(data=carwash_venues_df[['Venue Latitude','Venue Longitude']].groupby(['Venue Latitu
map_london
```

Out[18]:



Extract traffic volume information from Department for Transport and prepare the data

In [19]:

```
url = 'http://data.dft.gov.uk/road-traffic/local_authority_traffic.csv'
traffic_df = pd.read_csv(url)
#we only need data for the recent year
traffic_df = traffic_df[traffic_df['year'] == 2019]
#we only need data for each Authority (non-London are included here as well) and traffic vo
traffic_df = traffic_df[['local_authority_name','cars_and_taxis']].copy()
traffic_df.head()
```

Out[19]:

	local_authority_name	cars_and_taxis
0	Aberdeenshire	1.537817e+09
1	Lambeth	3.977109e+08
2	Newcastle upon Tyne	1.006029e+09
3	Tower Hamlets	4.652937e+08
4	St. Helens	7.281754e+08

In [20]:

```
traffic_df.shape
```

Out[20]:

```
(205, 2)
```

In [21]:

```
traffic_df = traffic_df.rename(columns={'local_authority_name': 'Borough'}) #rename the col
df.columns = df.columns.str.rstrip() #make sure again there is no white space
df_merged = pd.merge(df, traffic_df, on='Borough', how='left') #join our main table with Bo
df_merged
```

Out[21]:

	Borough	Population	Latitude	Longitude	cars_and_taxis
0	Barking and Dagenham	212906	51.5541	0.150504	3.882197e+08
1	Barnet	395896	51.6531	-0.200226	1.016422e+09
2	Bexley	248287	51.4417	0.150488	6.273994e+08
3	Brent	329771	51.5638	-0.27576	5.378248e+08
4	Bromley	332336	51.4028	0.0148142	7.759492e+08
5	Camden	270029	51.5423	-0.13956	2.184910e+08
6	Croydon	386710	51.3713	-0.101957	7.481640e+08
7	Ealing	341806	51.5127	-0.305195	6.879543e+08
8	Enfield	333794	51.6521	-0.0810175	9.518058e+08
9	Greenwich	287942	51.4821	-0.0045417	6.167056e+08
10	Hackney	281120	51.5432	-0.0493621	2.506129e+08
11	Hammersmith and Fulham	185143	51.492	-0.22364	2.695065e+08
12	Haringey	268647	51.6015	-0.111782	3.160339e+08
13	Harrow	251160	51.5968	-0.337316	4.763815e+08
14	Havering	259552	51.5444	-0.144307	8.531970e+08
15	Hillingdon	306870	51.5425	-0.448335	1.337783e+09
16	Hounslow	271523	51.4686	-0.361347	9.067470e+08
17	Islington	242467	51.5384	-0.0999051	1.984526e+08
18	Kensington and Chelsea	156129	51.4985	-0.199043	2.632956e+08
19	Kingston upon Thames	177507	51.4096	-0.306262	5.427724e+08
20	Lambeth	326034	51.5013	-0.117287	3.977109e+08
21	Lewisham	305842	51.4624	-0.0101331	4.674575e+08
22	Merton	206548	51.4109	-0.188097	3.513233e+08
23	Newham	353134	51.53	0.029318	5.119240e+08
24	Redbridge	305222	51.5763	0.0454097	6.815924e+08
25	Richmond upon Thames	198019	51.4406	-0.307639	4.548129e+08
26	Southwark	318830	51.5029	-0.103458	3.847811e+08
27	Sutton	206349	51.3575	-0.173627	4.018334e+08
28	Tower Hamlets	324745	51.5256	-0.0335853	4.652937e+08
29	Waltham Forest	276983	51.5982	-0.0178367	4.707462e+08
30	Wandsworth	329677	51.457	-0.193261	4.392001e+08

	Borough	Population	Latitude	Longitude	cars_and_taxis
31	Westminster	261317	51.5004	-0.12654	3.895788e+08

In [22]:

```
#peform another merge to create a final table for analysis
df_merged = df_merged.rename(columns={'Borough': 'Borough Name'})
df_merged_edit = df_merged[['Borough Name','Population', 'cars_and_taxis']].copy()
df_final = pd.merge(carwash_venues_df, df_merged_edit, on='Borough Name', how='left' ) #per
df_final = df_final.drop(['Venue Category'], axis=1)
df_final['cars_and_taxis'] = df_final['cars_and_taxis'].div(1000000).round(2) #devide by 1m
df_final
```

Out[22]:

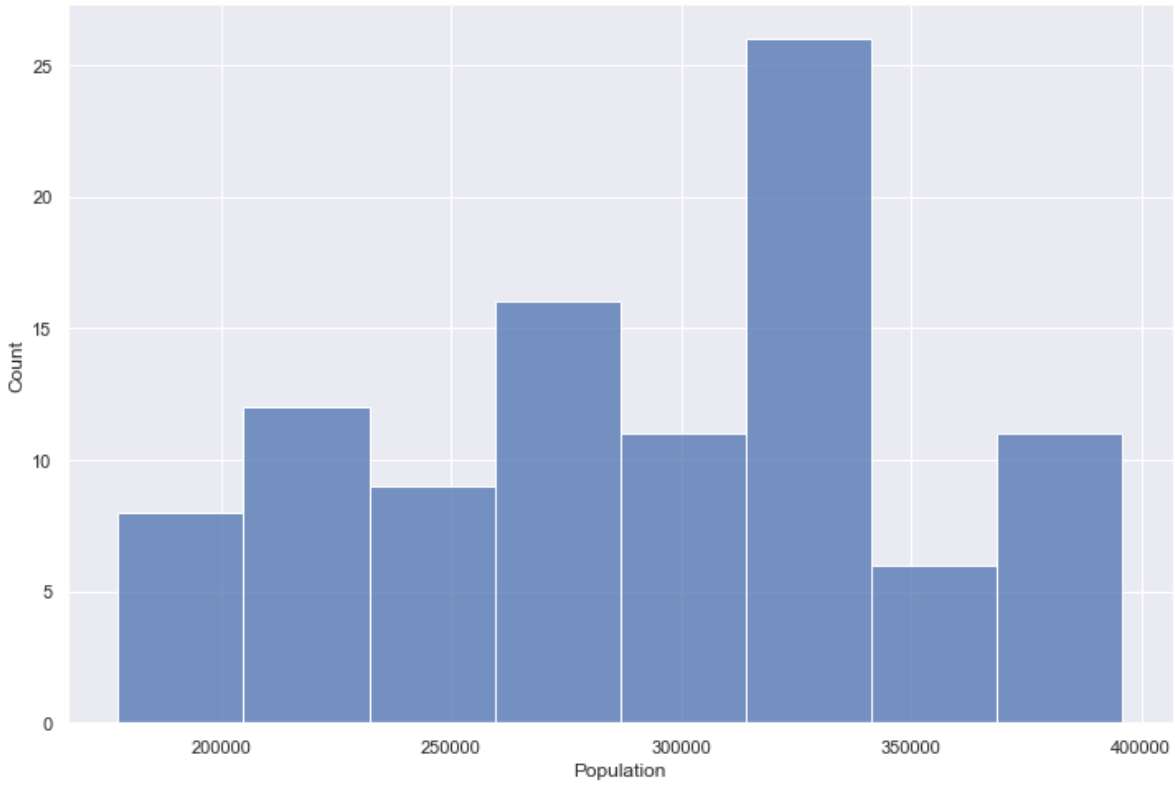
	Borough Name	Latitude	Longitude	Venue	Venue Latitude	Venue Longitude	Distance from Borough city centre	Population
0	Barking and Dagenham	51.554117	0.150504	BP	51.549951	0.161963	918	212906
1	Barking and Dagenham	51.554117	0.150504	BP	51.565010	0.203570	3867	212906
2	Barking and Dagenham	51.554117	0.150504	Waves Hand Car Wash	51.566331	0.190599	3089	212906
3	Barking and Dagenham	51.554117	0.150504	Ship And Shovel Carwash	51.532009	0.117131	3375	212906
4	Barking and Dagenham	51.554117	0.150504	IMO Car Wash	51.535007	0.103300	3899	212906
...	...	...	...	...	...	...	...	...
94	Kingston upon Thames	51.409627	-0.306262	TheEcoSmart - Car Valeting East London	51.411908	-0.208150	6816	177507
95	Merton	51.410870	-0.188097	IMO Car Wash	51.413672	-0.186490	331	206548
96	Merton	51.410870	-0.188097	BP	51.364920	-0.214090	5424	206548
97	Redbridge	51.576320	0.045410	Chigwell Hand Car Wash	51.597480	0.039001	2396	305222
98	Redbridge	51.576320	0.045410	SupaShine Chigwell	51.628731	0.062752	5956	305222

99 rows × 9 columns

Produce histograms to see how data varies

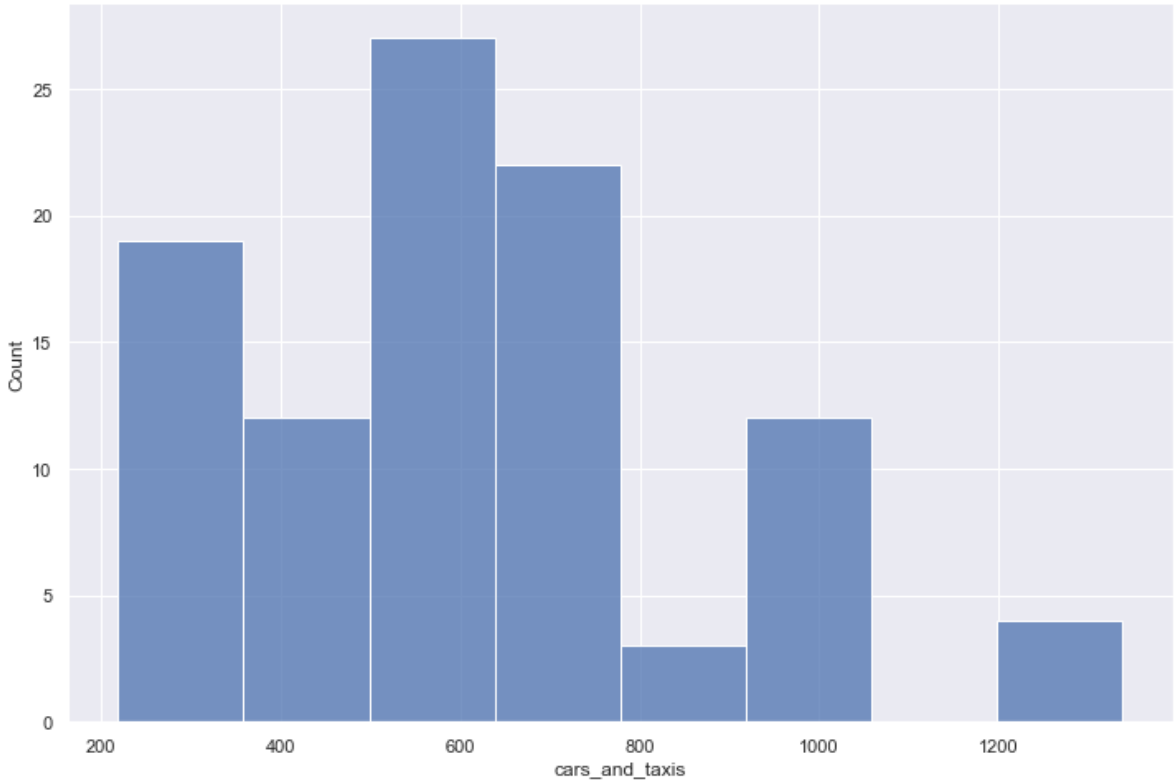
In [23]:

```
sns.set(rc={'figure.figsize':(12,8)})
ax = sns.histplot(x="Population",data=df_final)
```



In [24]:

```
ax = sns.histplot(x="cars_and_taxis",data=df_final)
```



K-means analysis.

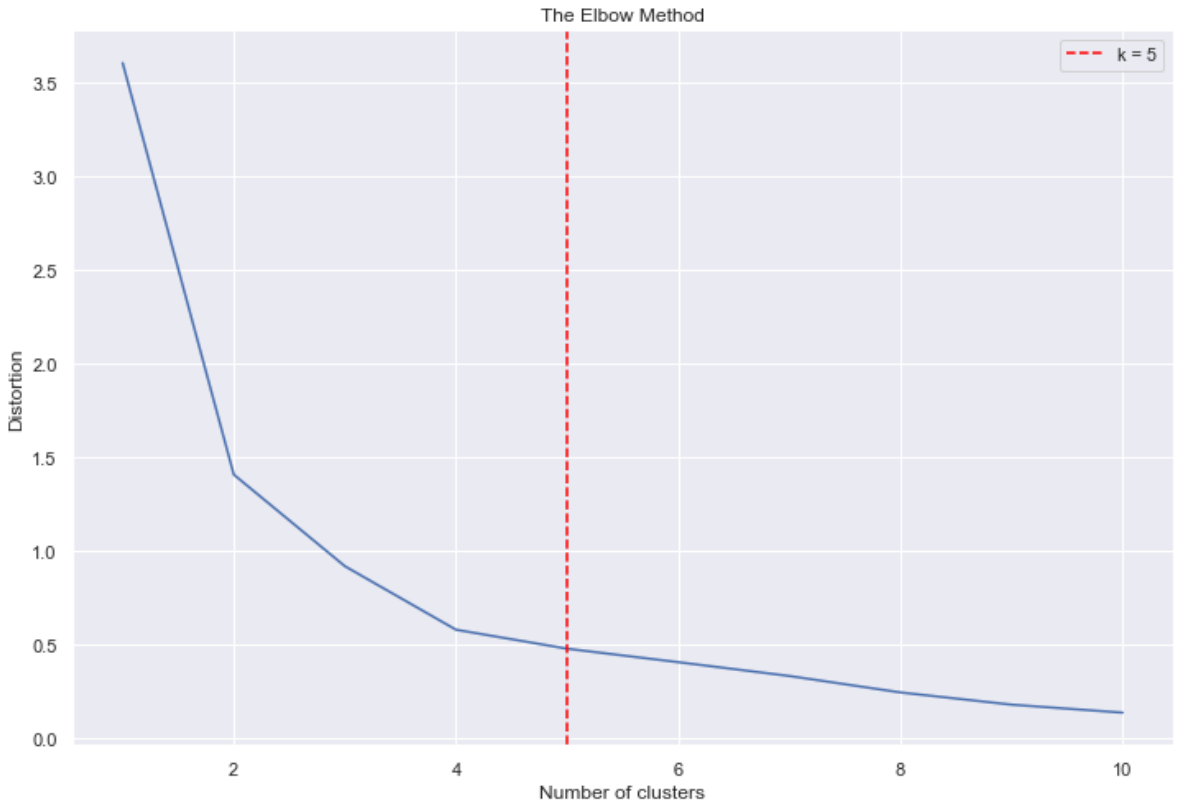
There are several algorithms that can be used to cluster data.

K-Means aims to partition the observations into a predefined number of clusters (k) in which each point belongs to the cluster with the nearest mean. It starts by randomly selecting k centroids and assigning the points to the closest cluster, then it updates each centroid with the mean of all points in the cluster. This algorithm is convenient when you need the get a precise number of groups, and it’s more appropriate for a small number of even clusters.

Here, in order to define the right k, I shall use the Elbow Method: plotting the variance as a function of the number of clusters and picking the k that flats the curve.

In [25]:

```
X = df_final[["Latitude","Longitude"]]
max_k = 10
## iterations
distortions = []
for i in range(1, max_k+1):
    if len(X) >= i:
        model = cluster.KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10, rand
        model.fit(X)
        distortions.append(model.inertia_)
## best k: the lowest derivative
k = [i*100 for i in np.diff(distortions,2)].index(min([i*100 for i
     in np.diff(distortions,2)]))
## plot
fig, ax = plt.subplots()
ax.plot(range(1, len(distortions)+1), distortions)
ax.axvline(k, ls='--', color="red", label="k = "+str(k))
ax.set(title='The Elbow Method', xlabel='Number of clusters',
       ylabel="Distortion")
ax.legend()
ax.grid(True)
plt.show()
```



K = 5 seems to be the right number of clusters to use. Re-model using k = 5.

In [26]:

```
k = 5
model = cluster.KMeans(n_clusters=k, init='k-means++')
X = df_final[["Latitude","Longitude"]]
## clustering
df_X = X.copy()
df_X["cluster"] = model.fit_predict(X)
## find real centroids
closest, distances = scipy.cluster.vq.vq(model.cluster_centers_,
                     df_X.drop("cluster", axis=1).values)
df_X["centroids"] = 0
for i in closest:
    df_X["centroids"].iloc[i] = 1
## add clustering info to the original dataset
df_final[["cluster","centroids"]] = df_X[["cluster","centroids"]]
df_final.sample(5)
```

```
C:\Users\serge\anaconda3\lib\site-packages\pandas\core\indexing.py:670: Sett
ingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)
  iloc._setitem_with_indexer(indexer, value)
```

Out[26]:

	Borough Name	Latitude	Longitude	Venue	Venue Latitude	Venue Longitude	Distance from Borough city centre	Population	cars
35	Bromley	51.402805	0.014814	Esso	51.388977	-0.025777	3212	332336	
12	Barnet	51.653090	-0.200226	Hand Car Wash	51.613247	-0.250251	5623	395896	
7	Barking and Dagenham	51.554117	0.150504	BP	51.573640	0.085850	4974	212906	
77	Hackney	51.543240	-0.049362	Albanian Car Wash	51.573754	0.017208	5724	281120	
31	Brent	51.563826	-0.275760	BP	51.521460	-0.354700	7218	329771	

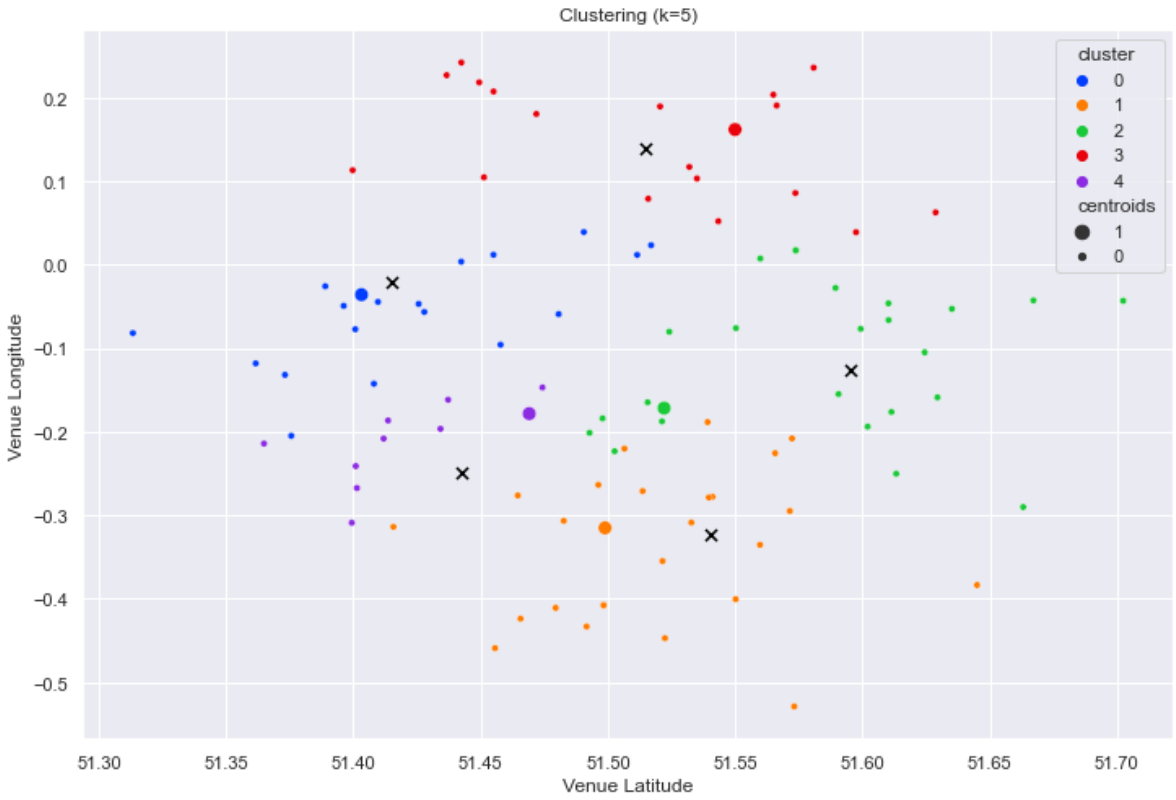
Plot a scatter plot with all of existing Car Washes allocated to individual clusters with their centroid points.

In [27]:

```
## plot
fig, ax = plt.subplots()
sns.scatterplot(x="Venue Latitude", y="Venue Longitude", data=df_final,
                palette=sns.color_palette("bright",k),
                hue='cluster', size="centroids", size_order=[1,0],
                legend="brief", ax=ax).set_title('Clustering (k='+str(k)+')')
th_centroids = model.cluster_centers_
ax.scatter(th_centroids[:,0], th_centroids[:,1], s=50, c='black',
           marker="x")
```

Out[27]:

```
<matplotlib.collections.PathCollection at 0x1e9c49f9d90>
```

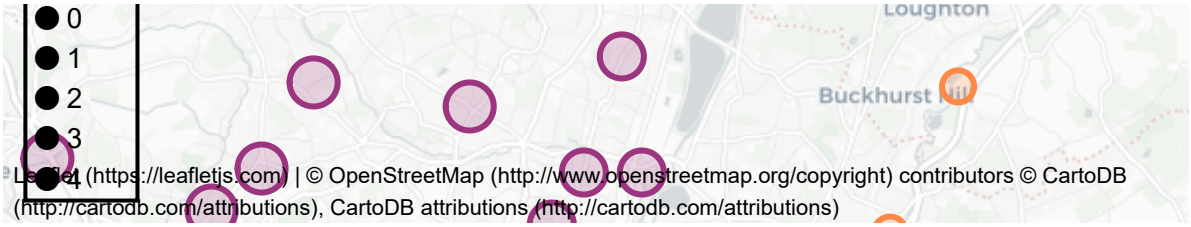


Finally, plot a map with existing clustered Car Washes with a bubble function. The bubble size represents volume of traffic in a particular area.

In [28]:

```
x, y = "Venue Latitude", "Venue Longitude"
color = "cluster"
size = "cars_and_taxis"
popup = "Venue"
marker = "centroids"
data = df_final.copy()
## create color column
lst_elements = sorted(list(df_final[color].unique()))
lst_colors = ['#%06X' % np.random.randint(0, 0xFFFFFF) for i in
              range(len(lst_elements))]
data["color"] = data[color].apply(lambda x:
                lst_colors[lst_elements.index(x)])
## create size column (scaled)
scaler = preprocessing.MinMaxScaler(feature_range=(3,15))
data["size"] = scaler.fit_transform(
               data[size].values.reshape(-1,1)).reshape(-1)
## initialize the map with the starting location
map_ = folium.Map(location=[lat, lng], tiles="cartodbpositron",
                  zoom_start=11)
## add points
data.apply(lambda row: folium.CircleMarker(
           location=[row[x],row[y]], popup=row[popup],
           color=row["color"], fill=True,
           radius=row["size"]).add_to(map_), axis=1)
## add html legend
legend_html = """<div style="position:fixed; bottom:10px; left:10px; border:2px solid black
for i in lst_elements:
     legend_html = legend_html+""" <i class="fa fa-circle
     fa-1x" style="color:"""+lst_colors[lst_elements.index(i)]+""""">
     </i> """+str(i)+"""<br>"""
legend_html = legend_html+"""</div>"""
map_.get_root().html.add_child(folium.Element(legend_html))
## add centroids marker
lst_elements = sorted(list(df_final[marker].unique()))
data[data[marker]==1].apply(lambda row:
           folium.Marker(location=[row[x],row[y]],
           popup=row[marker], draggable=False,
           icon=folium.Icon(color="black")).add_to(map_), axis=1)
## plot the map
map_
```

Out[28]:



It seems that the ML algorithm has split the existing facilities in the following manner:

- Cluster 1: South London
- Cluster 2: West London
- Cluster 3: North London
- Cluster 4: East London
- Cluster 5: South-West London

Analyse each cluster.

In [29]:

```
cluster_1 = df_final.loc[df_final['cluster'] == 0]
cluster_1
```

Out[29]:

	Borough Name	Latitude	Longitude	Venue	Venue Latitude	Venue Longitude	Distance from Borough city centre	Population	ca
34	Bromley	51.402805	0.014814	BP	51.403212	-0.035880	3520	332336	
35	Bromley	51.402805	0.014814	Esso	51.388977	-0.025777	3212	332336	
36	Bromley	51.402805	0.014814	Shell	51.409696	-0.044399	4182	332336	
37	Bromley	51.402805	0.014814	Leo Hand Car Wash	51.442428	0.003712	4477	332336	
38	Bromley	51.402805	0.014814	IMO Car Wash	51.396218	-0.049229	4507	332336	
39	Bromley	51.402805	0.014814	BP	51.427894	-0.056504	5684	332336	
40	Bromley	51.402805	0.014814	Sydenham Hand CarWash	51.425696	-0.046902	4985	332336	
41	Bromley	51.402805	0.014814	BP	51.400804	-0.077238	6396	332336	
42	Bromley	51.402805	0.014814	Hand Car Wash Valet Centre	51.455074	0.011995	5821	332336	
52	Croydon	51.371305	-0.101957	100% car wash	51.361666	-0.118163	1555	386710	
53	Croydon	51.371305	-0.101957	BP	51.373168	-0.131866	2088	386710	
54	Croydon	51.371305	-0.101957	BP	51.313436	-0.081789	6592	386710	
55	Croydon	51.371305	-0.101957	Shell	51.408091	-0.142457	4968	386710	
56	Croydon	51.371305	-0.101957	Waves Hand Car Wash	51.375608	-0.204790	7161	386710	
69	Greenwich	51.482084	-0.004542	Charlton Car Wash	51.490540	0.039198	3174	287942	
70	Greenwich	51.482084	-0.004542	BP	51.480684	-0.059145	3788	287942	
71	Greenwich	51.482084	-0.004542	Silvertown CarWash	51.511523	0.011848	3468	287942	
72	Greenwich	51.482084	-0.004542	BP	51.516999	0.023454	4343	287942	
73	Greenwich	51.482084	-0.004542	Herne Hill Hand Car Wash	51.457858	-0.095731	6874	287942	

In [30]:

```
c1 = cluster_1.drop_duplicates(['Borough Name','Population', 'cars_and_taxis'])
total_pop_c1 = c1['Population'].sum()
print('Total Population in Cluster 1:', f"{total_pop_c1:,}")
total_vehicles_c1 = c1['cars_and_taxis'].sum()
print('Total Cars in Cluster 1:', f"{total_vehicles_c1:,}", 'million of vehicles per year i
print('Total Number of Car washes in Cluster 1:', cluster_1.shape[0])
```

```
Total Population in Cluster 1: 1,006,988
Total Cars in Cluster 1: 2,140.82 million of vehicles per year in 2019
Total Number of Car washes in Cluster 1: 19
```

In [31]:

```
cluster_2 = df_final.loc[df_final['cluster'] == 1]
cluster_2
```

Out[31]:

	Borough Name	Latitude	Longitude	Venue	Venue Latitude	Venue Longitude	Distance from Borough city centre	Population	ca
23	Brent	51.563826	-0.275760	Hand Wash Car	51.571516	-0.294850	1574	329771	
24	Brent	51.563826	-0.275760	Shell	51.565697	-0.225631	3475	329771	
25	Brent	51.563826	-0.275760	Rainbow Carwash	51.541113	-0.277896	2532	329771	
26	Brent	51.563826	-0.275760	Over The Rainbow Hand Car Wash and Valet Centre	51.539679	-0.278554	2694	329771	
27	Brent	51.563826	-0.275760	Vanguard Hand Car Wash	51.532776	-0.308695	4140	329771	
28	Brent	51.563826	-0.275760	Skyline Automotive	51.559772	-0.335296	4144	329771	
29	Brent	51.563826	-0.275760	Mermaid Hand Car Wash	51.513649	-0.270903	5595	329771	
30	Brent	51.563826	-0.275760	All Car Glass	51.572329	-0.208004	4782	329771	
31	Brent	51.563826	-0.275760	BP	51.521460	-0.354700	7218	329771	
32	Brent	51.563826	-0.275760	The American Car Wash	51.539175	-0.188431	6638	329771	
33	Brent	51.563826	-0.275760	H2O Car Wash & Valeting	51.506499	-0.220130	7454	329771	
57	Ealing	51.512655	-0.305195	Northfields Hand Car Wash & Valeting Service	51.498872	-0.315074	1680	341806	
58	Ealing	51.512655	-0.305195	BP	51.496262	-0.263516	3416	341806	
59	Ealing	51.512655	-0.305195	Ceramic Pro	51.482704	-0.306619	3335	341806	
60	Ealing	51.512655	-0.305195	Richmond Hand Car Wash	51.464615	-0.276245	5711	341806	
61	Ealing	51.512655	-0.305195	Waves Hand Car Wash	51.498320	-0.407678	7278	341806	

	Borough Name	Latitude	Longitude	Venue	Venue Latitude	Venue Longitude	Distance from Borough city centre	Population	ca
62	Ealing	51.512655	-0.305195	Waves Hand Car Wash	51.498306	-0.407683	7278	341806	
82	Harrow	51.596827	-0.337316	IMO Car Wash	51.645031	-0.383499	6243	251160	
83	Harrow	51.596827	-0.337316	Ruislip Auto Care	51.550179	-0.400491	6787	251160	
84	Hillingdon	51.542519	-0.448335	Station Car Wash	51.522425	-0.447150	2238	306870	
85	Hillingdon	51.542519	-0.448335	Airport Car Wash	51.491665	-0.433134	5758	306870	
86	Hillingdon	51.542519	-0.448335	Top Car Valet (TCV)	51.573199	-0.528915	6539	306870	
87	Hillingdon	51.542519	-0.448335	Cranford Hand Car Wash	51.479508	-0.410816	7480	306870	
88	Hounslow	51.468613	-0.361347	BP	51.465683	-0.423782	4342	271523	
89	Hounslow	51.468613	-0.361347	Hampton Wick Car Wash	51.415760	-0.313788	6745	271523	
90	Hounslow	51.468613	-0.361347	Vamp Truck & Bus Wash Ltd	51.455674	-0.459106	6931	271523	

In [32]:

```
c2 = cluster_2.drop_duplicates(['Borough Name','Population', 'cars_and_taxis'])
total_pop_c2 = c2['Population'].sum()
print('Total Population in Cluster 2:', f"{total_pop_c2:,}")
total_vehicles_c2 = c2['cars_and_taxis'].sum()
print('Total Cars in Cluster 2:', f"{total_vehicles_c2:,}", 'million of vehicles per year i
print('Total Number of Car washes in Cluster 2:', cluster_2.shape[0])
```

```
Total Population in Cluster 2: 1,501,130
Total Cars in Cluster 2: 3,946.6800000000003 million of vehicles per year in
2019
Total Number of Car washes in Cluster 2: 26
```

In [33]:

```
cluster_3 = df_final.loc[df_final['cluster'] == 2]
cluster_3
```

Out[33]:

	Borough Name	Latitude	Longitude	Venue	Venue Latitude	Venue Longitude	Distance from Borough city centre	Population	car
10	Barnet	51.653090	-0.200226	BP	51.629465	-0.158883	3882	395896	
11	Barnet	51.653090	-0.200226	hand car wash north finchley	51.611376	-0.176303	4928	395896	
12	Barnet	51.653090	-0.200226	Hand Car Wash	51.613247	-0.250251	5623	395896	
13	Barnet	51.653090	-0.200226	Hand Car Wash	51.602029	-0.193927	5700	395896	
14	Barnet	51.653090	-0.200226	Astoria Pro Valeting	51.663135	-0.290071	6304	395896	
15	Barnet	51.653090	-0.200226	Bourne Hill Car Wash	51.624454	-0.104866	7318	395896	
43	Camden	51.542305	-0.139560	Church Street Hand Car Wash	51.522077	-0.171609	3161	270029	
44	Camden	51.542305	-0.139560	Lellers Hand Car Wash	51.515613	-0.164762	3445	270029	
45	Camden	51.542305	-0.139560	The American Car Wash Company	51.524152	-0.080164	4582	270029	
46	Camden	51.542305	-0.139560	Tino Mobile Car Wash	51.521244	-0.187515	4064	270029	
47	Camden	51.542305	-0.139560	Starhand Car Wash	51.550317	-0.075813	4502	270029	
48	Camden	51.542305	-0.139560	Ocean Car Wash	51.590638	-0.154903	5484	270029	
49	Camden	51.542305	-0.139560	Belgravia Auto Valet Limited	51.497938	-0.184012	5820	270029	
50	Camden	51.542305	-0.139560	Waves Hand Car Wash	51.492825	-0.201205	6969	270029	
51	Camden	51.542305	-0.139560	5 Star Car Wash	51.502649	-0.223355	7291	270029	
63	Enfield	51.652085	-0.081018	BP	51.667164	-0.042631	3137	333794	
64	Enfield	51.652085	-0.081018	BP	51.635125	-0.052770	2715	333794	

	Borough Name	Latitude	Longitude	Venue	Venue Latitude	Venue Longitude	Distance from Borough city centre	Population	cars
65	Enfield	51.652085	-0.081018	BP	51.610260	-0.066170	4767	333794	
66	Enfield	51.652085	-0.081018	Shell	51.599288	-0.076798	5884	333794	
67	Enfield	51.652085	-0.081018	Crystal Hand Car Wash - Ikea	51.610198	-0.046220	5246	333794	
68	Enfield	51.652085	-0.081018	American KO Car wash	51.702440	-0.043130	6185	333794	
74	Hackney	51.543240	-0.049362	Jet service station hand car wash	51.559904	0.007467	4349	281120	
75	Hackney	51.543240	-0.049362	BP	51.589455	-0.027751	5357	281120	
76	Hackney	51.543240	-0.049362	GreenMan Car Wash	51.573861	0.017347	5738	281120	
77	Hackney	51.543240	-0.049362	Albanian Car Wash	51.573754	0.017208	5724	281120	

In [34]:

```
c3 = cluster_3.drop_duplicates(['Borough Name','Population', 'cars_and_taxis'])
total_pop_c3 = c3['Population'].sum()
print('Total Population in Cluster 3:', f"{total_pop_c3:,}")
total_vehicles_c3 = c3['cars_and_taxis'].sum()
print('Total Cars in Cluster 3:', f"{total_vehicles_c3:,}", 'million of vehicles per year i
print('Total Number of Car washes in Cluster 3:', cluster_3.shape[0])
```

```
Total Population in Cluster 3: 1,280,839
Total Cars in Cluster 3: 2,437.33 million of vehicles per year in 2019
Total Number of Car washes in Cluster 3: 25
```

In [35]:

```
cluster_4 = df_final.loc[df_final['cluster'] == 3]
cluster_4
```

Out[35]:

	Borough Name	Latitude	Longitude	Venue	Venue Latitude	Venue Longitude	Distance from Borough city centre	Population	c
0	Barking and Dagenham	51.554117	0.150504	BP	51.549951	0.161963	918	212906	
1	Barking and Dagenham	51.554117	0.150504	BP	51.565010	0.203570	3867	212906	
2	Barking and Dagenham	51.554117	0.150504	Waves Hand Car Wash	51.566331	0.190599	3089	212906	
3	Barking and Dagenham	51.554117	0.150504	Ship And Shovel Carwash	51.532009	0.117131	3375	212906	
4	Barking and Dagenham	51.554117	0.150504	IMO Car Wash	51.535007	0.103300	3899	212906	
5	Barking and Dagenham	51.554117	0.150504	IMO Car Wash	51.520519	0.189557	4615	212906	
6	Barking and Dagenham	51.554117	0.150504	BP	51.580810	0.236020	6621	212906	
7	Barking and Dagenham	51.554117	0.150504	BP	51.573640	0.085850	4974	212906	
8	Barking and Dagenham	51.554117	0.150504	ARC Car Wash	51.515858	0.078961	6533	212906	
9	Barking and Dagenham	51.554117	0.150504	Shell	51.543413	0.052078	6916	212906	
16	Bexley	51.441679	0.150488	Car Wash at The Co-operative	51.451353	0.104628	3359	248287	
17	Bexley	51.441679	0.150488	Magnificent Hand Car Wash	51.471912	0.180476	3956	248287	
18	Bexley	51.441679	0.150488	IMO Car Wash	51.455107	0.207403	4221	248287	
19	Bexley	51.441679	0.150488	Hand Car Wash	51.449471	0.218372	4789	248287	
20	Bexley	51.441679	0.150488	Supashine Dartford	51.436686	0.226948	5334	248287	
21	Bexley	51.441679	0.150488	Olley's Posh Wosh	51.399753	0.113250	5335	248287	

	Borough Name	Latitude	Longitude	Venue	Venue Latitude	Venue Longitude	Distance from Borough city centre	Population	c
22	Bexley	51.441679	0.150488	Shell	51.442457	0.242187	6363	248287	
97	Redbridge	51.576320	0.045410	Chigwell Hand Car Wash	51.597480	0.039001	2396	305222	
98	Redbridge	51.576320	0.045410	SupaShine Chigwell	51.628731	0.062752	5956	305222	

In [36]:

```
c4 = cluster_4.drop_duplicates(['Borough Name','Population', 'cars_and_taxis'])
total_pop_c4 = c4['Population'].sum()
print('Total Population in Cluster 4:', f"{total_pop_c4:,}")
total_vehicles_c4 = c4['cars_and_taxis'].sum()
print('Total Cars in Cluster 4:', "{:,.2f}".format(total_vehicles_c4), 'million of vehicles
print('Total Number of Car washes in Cluster 4:', cluster_4.shape[0])
```

```
Total Population in Cluster 4: 766,415
Total Cars in Cluster 4: 1,697.21 million of vehicles per year in 2019
Total Number of Car washes in Cluster 4: 19
```

In [37]:

```
cluster_5 = df_final.loc[df_final['cluster'] == 4]
cluster_5
```

Out[37]:

	Borough Name	Latitude	Longitude	Venue	Venue Latitude	Venue Longitude	Distance from Borough city centre	Populatic
78	Hammersmith and Fulham	51.492038	-0.223640	Chelsea Coachworks Ltd	51.469103	-0.178297	4049	1851
79	Hammersmith and Fulham	51.492038	-0.223640	Julian Hand Car Wash	51.474326	-0.147029	5665	1851
80	Hammersmith and Fulham	51.492038	-0.223640	Mint Green Car Wash	51.434252	-0.196447	6703	1851
81	Hammersmith and Fulham	51.492038	-0.223640	BP	51.437259	-0.161596	7463	1851
91	Kingston upon Thames	51.409627	-0.306262	BP	51.401451	-0.267320	2853	1775(
92	Kingston upon Thames	51.409627	-0.306262	Kingston Hand Carwash	51.399402	-0.308994	1153	1775(
93	Kingston upon Thames	51.409627	-0.306262	Waves Hand Car Wash	51.401001	-0.241093	4626	1775(
94	Kingston upon Thames	51.409627	-0.306262	TheEcoSmart - Car Valeting East London	51.411908	-0.208150	6816	1775(
95	Merton	51.410870	-0.188097	IMO Car Wash	51.413672	-0.186490	331	2065
96	Merton	51.410870	-0.188097	BP	51.364920	-0.214090	5424	2065

In [38]:

```
c5 = cluster_5.drop_duplicates(['Borough Name','Population', 'cars_and_taxis'])
total_pop_c5 = c5['Population'].sum()
print('Total Population in Cluster 5:', f"{total_pop_c5:,}")
total_vehicles_c5 = c5['cars_and_taxis'].sum()
print('Total Cars in Cluster 5:', "{:,.2f}".format(total_vehicles_c5), 'million of vehicles
print('Total Number of Car washes in Cluster 5:', cluster_5.shape[0])
```

```
Total Population in Cluster 5: 569,198
Total Cars in Cluster 5: 1,163.60 million of vehicles per year in 2019
Total Number of Car washes in Cluster 5: 10
```

Results

We can now proceed and analyse each cluster by dividing population size and volume of traffic by number of car washes.
We are interested in lowest number of these.

In [134]:

```
#define clusters
clusters = [cluster_1, cluster_2, cluster_3, cluster_4, cluster_5]
#define population
totals_pop = [total_pop_c1, total_pop_c2, total_pop_c3, total_pop_c4, total_pop_c5]#define
#define vehicle volume
totals_vehicle = [total_vehicles_c1,total_vehicles_c2,total_vehicles_c3,total_vehicles_c4,t

#create results table
results = pd.DataFrame(index=[1,2,3,4,5], columns=['Cluster','No. of Car Washes','Car Wash
results['Cluster'] = ['Cluster %s' %i for i in range(1, len(results) + 1)]
results['No. of Car Washes'] = [cluster_1.shape[0],cluster_2.shape[0],cluster_3.shape[0],cl
#fill in car wash per population
appended_data = []
for i, x in zip(clusters,totals_pop):
 data = i.shape[0]/x
 appended_data.append(data)
results['Car Wash per Population'] = np.array(appended_data)

#fill in car wash per population
appended_data_1 = []
for i, x in zip(clusters,totals_vehicle):
 data1 = i.shape[0]/x
 appended_data_1.append(data1)
results['Car Wash no. per Traffic'] = np.array(appended_data_1)
results

# results['Car Wash per Population'] = [cluster_1.shape[0]/total_pop_c1,cluster_2.shape[0]/
# results['Car Wash per no. of Traffic'] = [cluster_1.shape[0]/total_vehicles_c1,cluster_2.
```

Out[134]:

	Cluster	No. of Car Washes	Car Wash per Population	Car Wash no. per Traffic
1	Cluster 1	19	0.000019	0.008875
2	Cluster 2	26	0.000017	0.006588
3	Cluster 3	25	0.000020	0.010257
4	Cluster 4	19	0.000025	0.011195
5	Cluster 5	10	0.000018	0.008594

Find the lowest numbers per cluster

In [138]:

```
results.loc[results['Car Wash per Population'].idxmin()]
```

Out[138]:

```
Cluster                       Cluster 2
No. of Car Washes                    26
Car Wash per Population     1.73203e-05
Car Wash no. per Traffic     0.00658782
Name: 2, dtype: object
```

In [137]:

```
results.loc[results['Car Wash no. per Traffic'].idxmin()]
```

Out[137]:

```
Cluster                       Cluster 2
No. of Car Washes                    26
Car Wash per Population     1.73203e-05
Car Wash no. per Traffic     0.00658782
Name: 2, dtype: object
```

Discussion

Cluster no.2 has the lowest number of car wash facilities per population and traffic volume, despite having the highest number of Car Washes. Therefore, for a potential investor starting a new venture this area would be highly desirable. The Boroughs include:

- Brent
- Harrow
- Ealing
- Hillingdon

Indeed, this area benefits from dense population and high amount of vehicle traffic, for example due to Heathrow Airport, therefore it is an attractive location. Further analysis could benefit incorporating a cost consideration i.e. the optimum location from a financial point of view.

Conclusion

This report evaluated the best location to open a new car wash in London using Data Science techniques. Publicly available data was extracted on locations of London boroughs, existing car wash locations and supporting information such as population size and vehicle volume. This data was plotted and visualized and K-Means clustering was used to partition the observations into an optimum number of clusters. The analysis concluded that cluster no.4 which is roughly the area of West London has less competition compared to other areas. Four boroughs were identified that would be the most beneficial.

References

1) Wikipedia (2021), List of London Boroughs, Available at: https://en.wikipedia.org/wiki/List_of_London_boroughs (https://en.wikipedia.org/wiki/List_of_London_boroughs) (Accessed 31/01/2021)

2) Departmment for Transport (2021), Road Traffic Statistics, Available at: https://roadtraffic.dft.gov.uk/downloads (https://roadtraffic.dft.gov.uk/downloads) (Accessed 31/01/2021)

In [ ]: