Capstone Project - Final Assessment

Applied Data Science Capstone - [Coursera (https://www.coursera.org/learn/applied-data-science-capstone/home/welcome)](https://www.coursera.org/learn/applied-data-science-capstone/home/welcome)

Business problem

In this project we will try to find the the right location for a coffee shop in London (understanding location as Borough or cluster of Boroughs). The criteria to classify and rank the different Boroughs is agreed with the client.

Demographic preferences.

1. Growing population
2. Working population
3. Mid-high income
4. Low crime rate

Analytics approach

We are going to search for different sources of data that combined can help us make a better decision. The data we are going to analyze can be grouped into:

1. Pure demographics data. Looking at areas where the population is growing.
2. Profiling data including unemployment rates, average gross income and average house price and crime indicators.
3. Business metrics, number of businesses, two-year business survival rate.
4. Competition and other business in the area information.

Data Sources

London Boroughs Population - UK National Statistics (https://www.citypopulation.de/php/uk-greaterlondon.php)
London Borough Profiles - London Data Store (https://data.london.gov.uk/dataset/london-borough-profiles)
London Boroughs GeoJson file - Carto.com (https://joshuaboyd1.carto.com/tables/london_boroughs_proper/public)
FourSquare API - Google Cloud (https://foursquare.com/developers/explore#req=users%2Fself)
Postcode Lookup API - Postcodes.io (https://postcodes.io/)

1. Getting / Cleaning the Data

1.1 List of boroughs and population trends from wikipedia

We start scraping wikipedia to get the full list and correct names of all the Boroughs. We also include the population in 2011 and 2017.

In [6]:
```
df_web.head()
```

Out[6]:

	pop_11	pop_17	pop_inc
name			
Barking and Dagenham	187029	210711	0.13
Barnet	357538	387803	0.08
Bexley	232774	246124	0.06
Brent	312245	329102	0.05
Bromley	310554	329391	0.06

1.2 London Borough Profiling data

The London Borough Profiles help paint a general picture of an area by presenting a range of headline indicator data covering demographic, economic, social and environmental datasets for each borough, alongside relevant comparator areas. We are taking only the variables we consider relevant for this project.

In [9]:
```
df_pro.head()
```

Out[9]:

	inner_outer	percent_pop_work_age	unemployment_rate	gross_annual_pay	active_businesses	two_year_survival_rate	crime_
name							
City of London	Inner	73.1	6.2	34161.16129	26130	64.3	84.868
Barking and Dagenham	Outer	63.1	11.0	27886.00000	6560	73.0	83.400
Barnet	Outer	64.9	8.5	33443.00000	26190	73.8	62.700
Bexley	Outer	62.9	7.6	34350.00000	9075	73.5	51.800
Brent	Outer	67.8	7.5	29812.00000	15745	74.4	78.800

1.3 GeoJson data

GeoJSON is a file format for representing geodata as JSON. In this format, every Feature element has a geometry (in this case, the geometry of the different boroughs) and additional properties.

In [12]:
```
df_geo.head()
```

Out[12]:

	name	cartodb_id	created_at	updated_at	geometry
0	Barking and Dagenham	1	2015-07-01T09:57:45	2015-07-01T09:57:45	(POLYGON ((0.148209 51.599635, 0.148199 51.599...
1	Barnet	2	2015-07-01T09:57:45	2015-07-01T09:57:45	(POLYGON ((-0.183361 51.668682, -0.183383 51.6...
2	Bexley	3	2015-07-01T09:57:45	2015-07-01T09:57:45	(POLYGON ((0.158044 51.509044, 0.156309 51.509...
3	Brent	4	2015-07-01T09:57:45	2015-07-01T09:57:45	(POLYGON ((-0.212138 51.555582, -0.212689 51.5...
4	Bromley	5	2015-07-01T09:57:45	2015-07-01T09:57:45	(POLYGON ((0.076463 51.430995, 0.075932 51.431...

1.4 Competition and other businesses

We are using the Foursquare API to get information about the surrounding areas.

In [14]:
```
location['lat'] = location.geometry.centroid.y
location['lng'] = location.geometry.centroid.x
location = location.drop('geometry', axis=1)
location.head()
```

Out[14]:

	name	lat	lng
0	Barking and Dagenham	51.544128	0.134830
1	Barnet	51.616216	-0.208416
2	Bexley	51.461051	0.144006
3	Brent	51.558737	-0.266241
4	Bromley	51.372166	0.053151

In [17]:
```python
boroughs_venues = pd.read_csv('data/venues_from_foursquare.csv', index_col=0)
boroughs_venues.head()
```

Out[17]:

	borough	borough_lat	borough_lng	venue_name	venue_lat	venue_lng	venue_cat
0	Barking and Dagenham	51.544128	0.13483	Capital Karts	51.531792	0.118739	Go Kart Track
1	Barking and Dagenham	51.544128	0.13483	Barking Park	51.545217	0.086134	Park
2	Barking and Dagenham	51.544128	0.13483	Central Park	51.559560	0.161981	Park
3	Barking and Dagenham	51.544128	0.13483	Mayesbrook Park	51.549842	0.108544	Park
4	Barking and Dagenham	51.544128	0.13483	Harrow Lodge Park	51.555648	0.197926	Park

In [18]:

```python
print(boroughs_venues['venue_cat'].value_counts())
boroughs_venues.head()
```

```
Park                           303
Pub                            209
Hotel                          171
Coffee Shop                    162
Café                           115
Pizza Place                     97
Indian Restaurant               80
Italian Restaurant              68
Garden                          58
Cocktail Bar                    53
Bakery                          52
Theater                         49
Movie Theater                   48
Scenic Lookout                  45
Art Gallery                     44
Gym / Fitness Center            42
Bookstore                       42
Restaurant                      38
Bar                             37
Supermarket                     33
Department Store                32
Portuguese Restaurant           32
Steakhouse                      32
Grocery Store                   31
Japanese Restaurant             31
Market                          31
Gastropub                       31
History Museum                  30
Turkish Restaurant              29
Historic Site                   27
Plaza                           27
Art Museum                      26
Wine Bar                        24
```

Garden Center	23
Music Venue	22
Asian Restaurant	22
Brewery	22
Ice Cream Shop	21
Thai Restaurant	21
Mediterranean Restaurant	20
Burger Joint	20
Sushi Restaurant	20
Breakfast Spot	19
Street Food Gathering	18
Trail	18
Performing Arts Venue	18
Museum	17
Seafood Restaurant	16
French Restaurant	16
Nature Preserve	16
Clothing Store	16
Botanical Garden	16
Climbing Gym	15
Indie Movie Theater	15
Greek Restaurant	15
Boutique	15
Chinese Restaurant	14
Farmers Market	14
Wine Shop	13
Flea Market	13
Tennis Stadium	13
Beer Store	13
Farm	13
Fish & Chips Shop	12
Canal	12
Multiplex	12
Korean Restaurant	12
Playground	12
Mini Golf	11
Beer Bar	11
Social Club	11

Science Museum	10
Sandwich Place	10
Cheese Shop	10
Golf Course	10
Record Shop	10
Airport Lounge	9
Forest	9
Spa	9
Dessert Shop	9
Bistro	9
Bike Shop	9
Chocolate Shop	9
BBQ Joint	9
Hotel Bar	8
Gourmet Shop	8
Liquor Store	8
Athletics & Sports	8
Pedestrian Plaza	8
Pool	8
Lake	8
Middle Eastern Restaurant	8
Jazz Club	7
Go Kart Track	7
Event Space	7
Fast Food Restaurant	7
Stadium	7
Golf Driving Range	7
Yoga Studio	7
Latin American Restaurant	6
American Restaurant	6
Warehouse Store	6
Whisky Bar	6
Deli / Bodega	6
Gym	6
Cricket Ground	6
Furniture / Home Store	6
English Restaurant	6
Concert Hall	6

Vegetarian / Vegan Restaurant	6
Tapas Restaurant	6
Castle	6
Butcher	6
Shopping Plaza	5
Fruit & Vegetable Store	5
Argentinian Restaurant	5
Train Station	5
Tea Room	5
Outdoor Sculpture	5
Salon / Barbershop	5
Ethiopian Restaurant	5
Falafel Restaurant	5
Waterfront	5
Indie Theater	4
Pet Store	4
Zoo Exhibit	4
Donut Shop	4
Opera House	4
Gym Pool	4
Roof Deck	4
Food & Drink Shop	4
Tennis Court	4
Comic Shop	4
Caribbean Restaurant	4
Shopping Mall	4
Monument / Landmark	4
Rugby Stadium	4
Sporting Goods Shop	4
Lounge	4
Food Court	4
Dim Sum Restaurant	3
Field	3
Diner	3
Lebanese Restaurant	3
Electronics Store	3
Beer Garden	3
Nightclub	3

Sculpture Garden	3
Creperie	3
Track	3
Toy / Game Store	3
Fish Market	3
Irish Pub	3
Soccer Stadium	3
Soccer Field	3
Modern European Restaurant	3
German Restaurant	3
Brasserie	2
Hobby Shop	2
Reservoir	2
Campground	2
Noodle House	2
Canal Lock	2
Gaming Cafe	2
Planetarium	2
African Restaurant	2
Palace	2
Pakistani Restaurant	2
Champagne Bar	2
Food	2
Brazilian Restaurant	2
Recording Studio	2
Vietnamese Restaurant	2
Cave	2
Food Truck	2
Fried Chicken Joint	2
Nail Salon	1
Airport Service	1
Bagel Shop	1
Arts & Crafts Store	1
Other Great Outdoors	1
Eastern European Restaurant	1
Harbor / Marina	1
Observatory	1
Men's Store	1

```
Bubble Tea Shop                     1
Persian Restaurant                  1
Road                                1
Racetrack                           1
Organic Grocery                     1
Theme Park Ride / Attraction        1
Gift Shop                           1
Rental Car Location                 1
Pharmacy                            1
Military Base                       1
Water Park                          1
Hockey Rink                         1
Airport                             1
Molecular Gastronomy Restaurant     1
Malay Restaurant                    1
Cupcake Shop                        1
Burrito Place                       1
Optical Shop                        1
Lingerie Store                      1
Skating Rink                        1
Souvenir Shop                       1
Mexican Restaurant                  1
Australian Restaurant               1
Aquarium                            1
Sports Club                         1
Juice Bar                           1
Film Studio                         1
Spanish Restaurant                  1
Windmill                            1
Name: venue_cat, dtype: int64
```

Out[18]:

	borough	borough_lat	borough_lng	venue_name	venue_lat	venue_lng	venue_cat
0	Barking and Dagenham	51.544128	0.13483	Capital Karts	51.531792	0.118739	Go Kart Track
1	Barking and Dagenham	51.544128	0.13483	Barking Park	51.545217	0.086134	Park
2	Barking and Dagenham	51.544128	0.13483	Central Park	51.559560	0.161981	Park
3	Barking and Dagenham	51.544128	0.13483	Mayesbrook Park	51.549842	0.108544	Park
4	Barking and Dagenham	51.544128	0.13483	Harrow Lodge Park	51.555648	0.197926	Park

In [32]:

```python
venues_grouped = boroughs_venues_filtered.groupby('borough')['venue_cat'].value_counts()
venues_grouped
```

Out[32]:

```
borough               venue_cat
Barking and Dagenham  Park           17
                      Pub            11
                      Coffee Shop     6
                      Hotel           5
                      Café            1
Barnet                Park           10
                      Pub            10
                      Café            6
                      Coffee Shop     5
Bexley                Park           12
                      Hotel           5
                      Pub             5
                      Coffee Shop     3
                      Café            1
Brent                 Park           10
                      Pub             6
                      Hotel           4
                      Coffee Shop     2
Bromley               Pub            14
                      Park           11
                      Coffee Shop    10
                      Café            1
Camden                Hotel          11
                      Park            7
                      Coffee Shop     2
                      Café            1
                      Pub             1
City of London        Hotel           9
                      Coffee Shop     4
                      Pub             2
                      Park            1
Croydon               Park           10
```

	Pub	10
	Café	6
	Coffee Shop	6
	Hotel	1
Ealing	Coffee Shop	9
	Pub	8
	Park	7
	Café	5
	Hotel	2
Enfield	Café	10
	Park	9
	Coffee Shop	8
	Pub	2
Greenwich	Pub	12
	Park	11
	Coffee Shop	7
	Café	4
	Hotel	2
Hackney	Hotel	6
	Pub	5
	Coffee Shop	4
	Park	3
	Café	1
Hammersmith and Fulham	Hotel	16
	Park	7
	Café	3
	Coffee Shop	1
	Pub	1
Haringey	Park	9
	Hotel	6
	Coffee Shop	5
	Pub	5
	Café	1
Harrow	Pub	9
	Park	8
	Coffee Shop	5
	Hotel	3
	Café	2

Havering	Park	12
	Coffee Shop	10
	Pub	7
	Café	2
	Hotel	1
Hillingdon	Park	8
	Pub	8
	Coffee Shop	7
	Hotel	4
	Café	1
Hounslow	Park	10
	Pub	10
	Café	5
	Coffee Shop	5
	Hotel	5
Islington	Hotel	8
	Park	6
	Coffee Shop	3
	Pub	3
	Café	1
Kensington and Chelsea	Hotel	15
	Park	9
	Café	3
	Coffee Shop	1
Kingston upon Thames	Park	13
	Pub	9
	Café	8
	Coffee Shop	2
	Hotel	2
Lambeth	Hotel	12
	Park	6
	Café	4
	Coffee Shop	1
Lewisham	Hotel	6
	Park	6
	Coffee Shop	4
	Café	2
	Pub	2

Merton	Park	16
	Pub	13
	Coffee Shop	9
	Café	8
Newham	Pub	9
	Coffee Shop	7
	Park	6
	Café	4
	Hotel	1
Redbridge	Park	15
	Pub	9
	Café	7
	Coffee Shop	4
Richmond upon Thames	Park	13
	Café	9
	Coffee Shop	7
	Pub	7
	Hotel	2
Southwark	Hotel	8
	Park	6
	Coffee Shop	4
	Pub	2
	Café	1
Sutton	Pub	15
	Park	12
	Café	6
	Coffee Shop	6
	Hotel	1
Tower Hamlets	Hotel	6
	Coffee Shop	5
	Pub	4
	Park	3
	Café	1
Waltham Forest	Park	12
	Pub	9
	Coffee Shop	8
	Café	4
Wandsworth	Hotel	15

```
                        Park            12
                        Café             4
                        Pub              1
Westminster             Hotel           15
                        Park             6
                        Café             3
                        Coffee Shop      2
Name: venue_cat, dtype: int64
```

```
In [35]: # we merge the cafe and coffee shop columns into one
venues_grouped_unst['Cafes'] = venues_grouped_unst['Coffee Shop'] + venues_grouped_unst[
'Café']
venues_grouped_unst = venues_grouped_unst.drop(['Coffee Shop', 'Café'], axis=1)
venues_grouped_unst.head()
```

Out[35]:

venue_cat	Hotel	Park	Pub	Cafes
borough				
Barking and Dagenham	5.0	17.0	11.0	7.0
Barnet	0.0	10.0	10.0	11.0
Bexley	5.0	12.0	5.0	4.0
Brent	4.0	10.0	6.0	2.0
Bromley	0.0	11.0	14.0	11.0

1.5 Merging the Data

We are going to combine all dataframes into one. We have df_web, df_pro, df_geo and venues_grouped_unst

In [37]:
```
df = df_geo.merge(right=df_concat, left_on=df_geo.name, right_on=df_concat.index)
df = df.drop('key_0', axis=1)
print(type(df))
df.head()
```

```
<class 'geopandas.geodataframe.GeoDataFrame'>
```

Out[37]:

	name	cartodb_id	created_at	updated_at	geometry	pop_11	pop_17	pop_inc	inner_outer	percent_pop_work_age	un
0	Barking and Dagenham	1	2015-07-01T09:57:45	2015-07-01T09:57:45	(POLYGON ((0.148209 51.599635, 0.148199 51.599...	187029	210711	0.13	Outer	63.1	11
1	Barnet	2	2015-07-01T09:57:45	2015-07-01T09:57:45	(POLYGON ((-0.183361 51.668682, -0.183383 51.6...	357538	387803	0.08	Outer	64.9	8.5
2	Bexley	3	2015-07-01T09:57:45	2015-07-01T09:57:45	(POLYGON ((0.158044 51.509044, 0.156309 51.509...	232774	246124	0.06	Outer	62.9	7.6
3	Brent	4	2015-07-01T09:57:45	2015-07-01T09:57:45	(POLYGON ((-0.212138 51.555582, -0.212689 51.5...	312245	329102	0.05	Outer	67.8	7.5
4	Bromley	5	2015-07-01T09:57:45	2015-07-01T09:57:45	(POLYGON ((0.076463 51.430995, 0.075932 51.431...	310554	329391	0.06	Outer	62.6	5.3

2. Data Exploration

Let's look at the data we have gathered to try to identify good locations for a new business opportunity.

In [25]:
```
# we summarise the different variables
df.describe().T
```

Out[25]:

	count	mean	std	min	25%	50%	75%	max
cartodb_id	33.0	17.000000	9.669540	1.00	9.00	17.00	25.0	33.0
pop_11	33.0	248618.393939	69871.558116	7412.00	206285.00	255483.00	304481.0	364815.0
pop_17	33.0	267424.272727	75224.420982	7654.00	235000.00	275505.00	323257.0	387803.0
pop_inc	33.0	0.073636	0.045471	-0.02	0.05	0.06	0.1	0.2
percent_pop_work_age	33.0	68.254545	3.911768	62.30	64.90	67.70	72.1	75.3
unemployment_rate	33.0	6.081818	1.853789	3.80	4.60	5.70	7.6	11.0
gross_annual_pay	33.0	34161.161290	3610.623761	27886.00	32056.00	33443.00	36429.0	42141.0
active_businesses	33.0	16403.333333	8838.768355	6560.00	11055.00	14350.00	18390.0	55385.0
two_year_survival_rate	33.0	73.769697	3.444514	63.80	73.00	74.40	75.8	78.8
crime_rate_per_thousand	33.0	84.868750	30.639073	50.40	64.10	78.00	99.6	212.4
median_house_price	33.0	465467.969697	204356.260649	243500.00	345000.00	410000.00	485000.0	1200000.0
Café	33.0	3.484848	2.762712	0.00	1.00	3.00	5.0	10.0
Coffee Shop	33.0	4.909091	2.742759	0.00	3.00	5.00	7.0	10.0
Hotel	33.0	5.181818	4.996590	0.00	1.00	4.00	8.0	16.0
Park	33.0	9.181818	3.711928	1.00	6.00	9.00	12.0	17.0
Pub	33.0	6.333333	4.392228	0.00	2.00	7.00	9.0	15.0

In [26]:
```
# we deal with null values
df.isnull().sum()
```

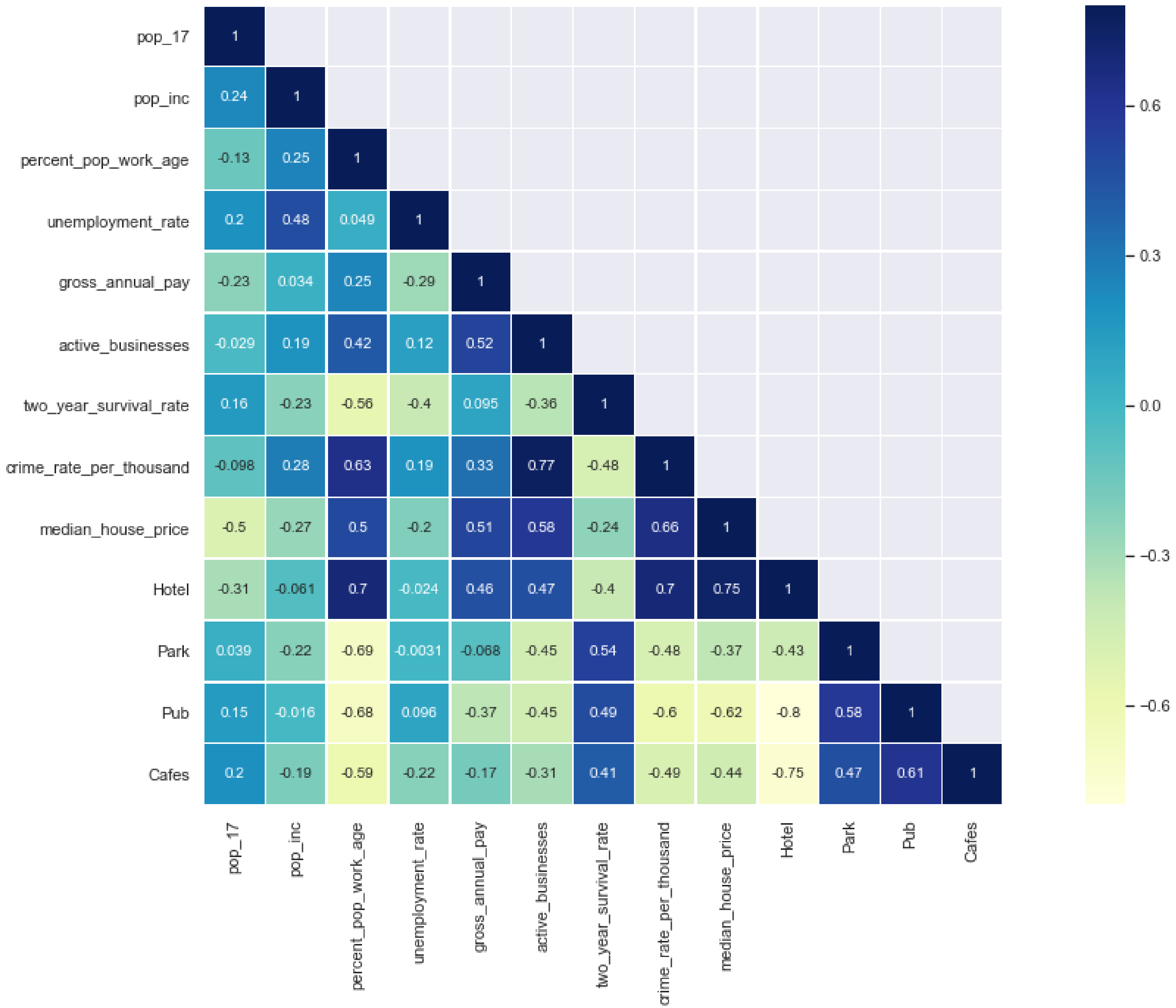
Out[26]:
```
name                        0
cartodb_id                  0
created_at                  0
updated_at                  0
geometry                    0
pop_11                      0
pop_17                      0
pop_inc                     0
inner_outer                 0
percent_pop_work_age        0
unemployment_rate           0
gross_annual_pay            0
active_businesses           0
two_year_survival_rate      0
crime_rate_per_thousand     0
median_house_price          0
Café                        0
Coffee Shop                 0
Hotel                       0
Park                        0
Pub                         0
dtype: int64
```

In [38]:

```python
# we subset all the numeric variables to perform further analysis
X = df[['pop_17', 'pop_inc', 'inner_outer', 'percent_pop_work_age',
       'unemployment_rate', 'gross_annual_pay',
       'active_businesses', 'two_year_survival_rate',
       'crime_rate_per_thousand', 'median_house_price',
      'Hotel', 'Park', 'Pub', 'Cafes']]
```

In [39]:
```
# we create a correlation matrix to identify existing relationships in the variables
corrMatt = X.corr()
mask = np.array(corrMatt)
mask[np.tril_indices_from(mask)] = False
fig,ax= plt.subplots()
fig.set_size_inches(20,10)
sns.heatmap(corrMatt, mask=mask,vmax=.8, cmap='YlGnBu', linewidths=.5, square=True,annot
=True)
```

Out[39]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x92b790>
```


pop_17
pop_inc
percent_pop_work_age
unemployment_rate
gross_annual_pay
active_businesses
two_year_survival_rate
crime_rate_per_thousand
median_house_price
Hotel
Park
Pub
Cafes
1
0.24 1
-0.13 0.25 1
0.2 0.48 0.049 1
-0.23 0.034 0.25 -0.29 1
-0.029 0.19 0.42 0.12 0.52 1
0.16 -0.23 -0.56 -0.4 0.095 -0.36 1
-0.098 0.28 0.63 0.19 0.33 0.77 -0.48 1
-0.5 -0.27 0.5 -0.2 0.51 0.58 -0.24 0.66 1
-0.31 -0.061 0.7 -0.024 0.46 0.47 -0.4 0.7 0.75 1
0.039 -0.22 -0.69 -0.0031 -0.068 -0.45 0.54 -0.48 -0.37 -0.43 1
0.15 -0.016 -0.68 0.096 -0.37 -0.45 0.49 -0.6 -0.62 -0.8 0.58 1
0.2 -0.19 -0.59 -0.22 -0.17 -0.31 0.41 -0.49 -0.44 -0.75 0.47 0.61 1
0.6
0.3
0.0
−0.3
−0.6

3. Clustering

3.1 Running the model. KMeans

In [41]:

```
# we convert categorical column with get_dummies
X = pd.get_dummies(X, drop_first=True)
X.head()
```

Out[41]:

	pop_17	pop_inc	percent_pop_work_age	unemployment_rate	gross_annual_pay	active_businesses	two_year_survival_rate	crime_rate
0	210711	0.13	63.1	11.0	27886.0	6560	73.0	83.4
1	387803	0.08	64.9	8.5	33443.0	26190	73.8	62.7
2	246124	0.06	62.9	7.6	34350.0	9075	73.5	51.8
3	329102	0.05	67.8	7.5	29812.0	15745	74.4	78.8
4	329391	0.06	62.6	5.3	37682.0	15695	78.6	64.1

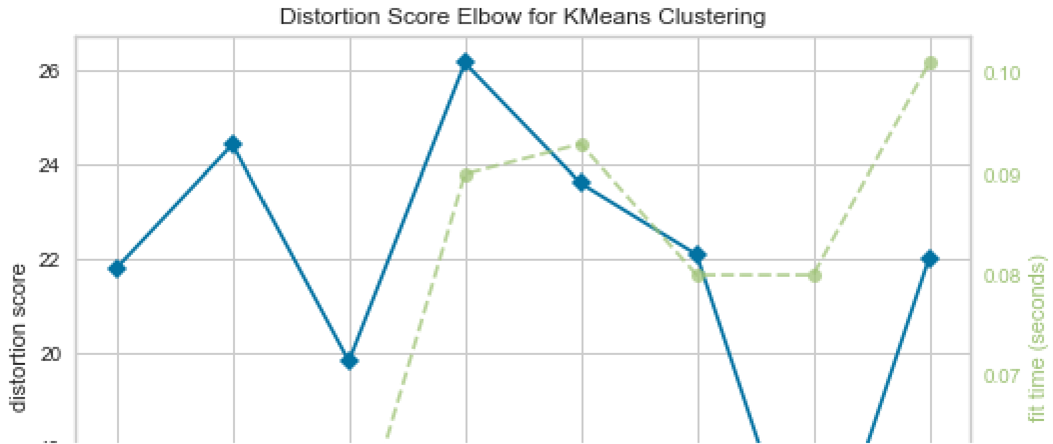
In [ ]:

```
clt_labels = model.labels_
clt_labels
```

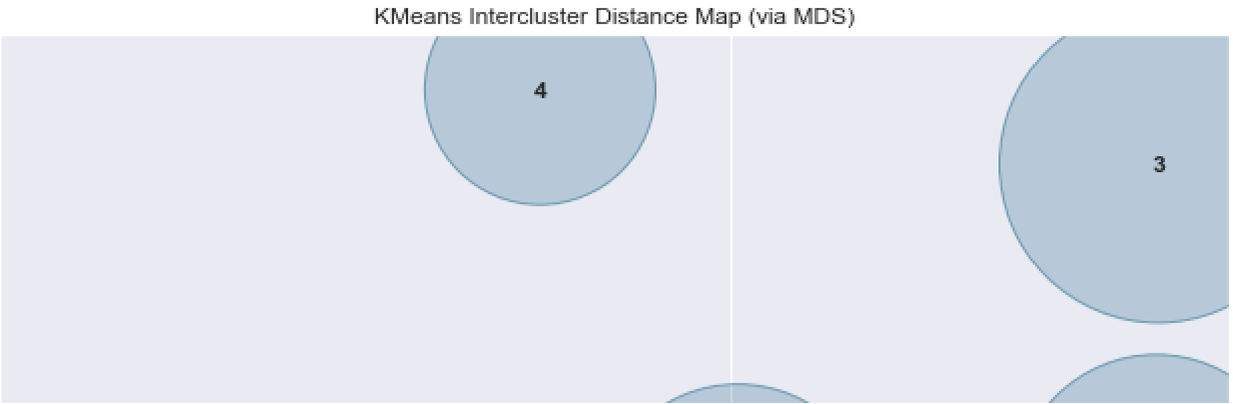
Clusters: We choose 6 clusters for the model to minimize the distortion score

In [45]:

```
from yellowbrick.cluster import KElbowVisualizer

# Instantiate the visualizer
visualizer = KElbowVisualizer(model, k=(4,12))

visualizer.fit(X_std)    # Fit the data to the visualizer
visualizer.poof()    # Draw/show/poof the data
```



In [80]:
```python
from yellowbrick.cluster import InterclusterDistance

visualizer = InterclusterDistance(KMeans(5))

visualizer.fit(X_std) # Fit the training data to the visualizer
visualizer.poof() # Draw/show/poof the data
```



3.2 Comparing the clusters

We can start comparing the different clusters using the criteria agreed at the begining of the project.

In [52]:
```
print('Cluster 0 is formed by', len(df_labeled[df_labeled['cluster_label'] == 0]), 'Boro
ughs:', df_labeled[df_labeled['cluster_label'] == 0]['name'].str.cat(sep=', '))
print('Cluster 1 is formed by', len(df_labeled[df_labeled['cluster_label'] == 1]), 'Boro
ughs:', df_labeled[df_labeled['cluster_label'] == 1]['name'].str.cat(sep=', '))
print('Cluster 2 is formed by', len(df_labeled[df_labeled['cluster_label'] == 2]), 'Boro
ughs:', df_labeled[df_labeled['cluster_label'] == 2]['name'].str.cat(sep=', '))
print('Cluster 3 is formed by', len(df_labeled[df_labeled['cluster_label'] == 3]), 'Boro
ugh:', df_labeled[df_labeled['cluster_label'] == 3]['name'].str.cat(sep=', '))
print('Cluster 4 is formed by', len(df_labeled[df_labeled['cluster_label'] == 4]), 'Boro
ughs:', df_labeled[df_labeled['cluster_label'] == 4]['name'].str.cat(sep=', '))
print('Cluster 5 is formed by', len(df_labeled[df_labeled['cluster_label'] == 5]), 'Boro
ugh:', df_labeled[df_labeled['cluster_label'] == 5]['name'].str.cat(sep=', '))
```

```
Cluster 0 is formed by 5 Boroughs: Bromley, Kingston upon Thames, Merton, Richmond up
on Thames, Sutton
Cluster 1 is formed by 15 Boroughs: Barking and Dagenham, Barnet, Bexley, Brent, Croy
don, Ealing, Enfield, Greenwich, Harrow, Havering, Hillingdon, Hounslow, Newham, Redb

ridge, Waltham Forest
Cluster 2 is formed by 3 Boroughs: Hammersmith and Fulham, Kensington and Chelsea, Wa
ndsworth
Cluster 3 is formed by 1 Borough: City of London
Cluster 4 is formed by 8 Boroughs: Camden, Hackney, Haringey, Islington, Lambeth, Lew
isham, Southwark, Tower Hamlets
Cluster 5 is formed by 1 Borough: Westminster
```

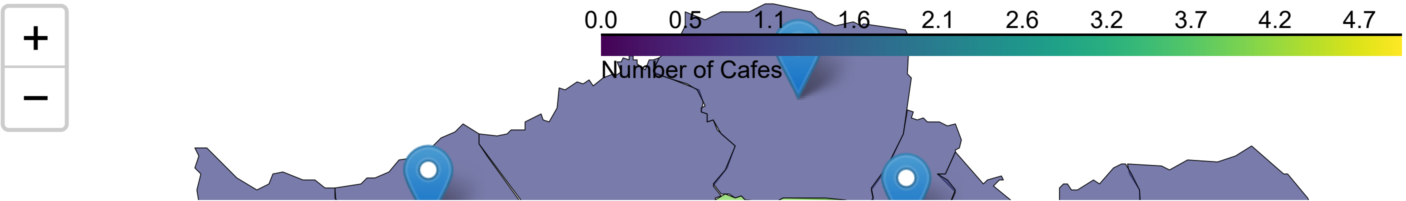
Clusters: These are the resulting clusters with the boroughs that form them

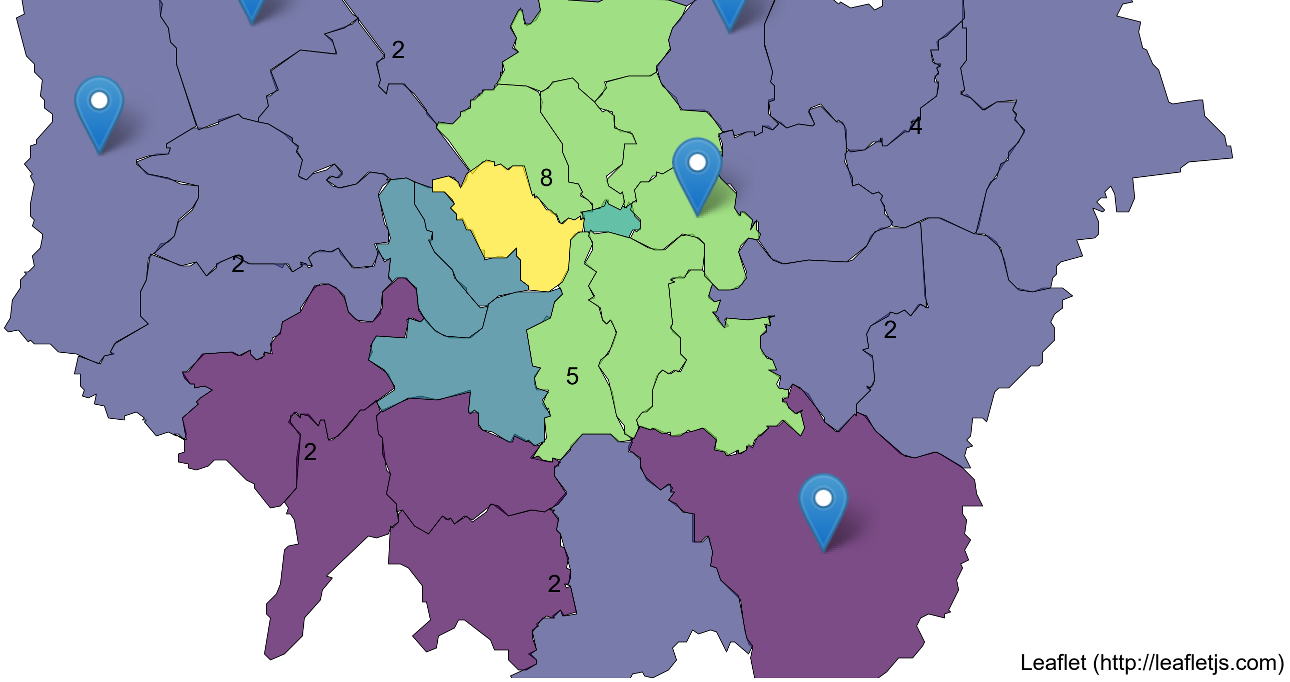
We can quickly see the boroughs on a map

In [110]:

```
lat = 51.510067
lng = -0.133869
zoom_start = 10


m = folium.Map(location=[lat, lng], zoom_start=zoom_start) # generate map centred around the Conrad Hotel
ft = 'cluster_label'
cmap = cm.linear.viridis.scale(df[ft].min(), df[ft].max())
folium.GeoJson(df, style_function=lambda feature: {
             'fillColor': cmap(feature['properties'][ft]),
             'fillOpacity' : 0.7,
             'weight' : 0.4,
            'color' : 'black'


}).add_to(m)

marker = MarkerCluster().add_to(m)
for ix, row in df.iterrows():
    text = "Borough: " + str(row['name']) + "<br>" + ft + ': '+ str(row[ft]) + "<br>" + "Population: " + str(row['pop_17'])
    popup = folium.Popup(IFrame(text, width=300, height=100))
    folium.Marker(location = [row.geometry.centroid.y ,row.geometry.centroid.x], popup=popup).add_to(marker)

cmap.caption = 'Number of Cafes'
m.add_child(cmap)

m
```

Out[110]:




2
4
8
2
2
5
2
2
Leaflet (http://leafletjs.com)

In [54]:
```
df_cluster
```

Out[54]:

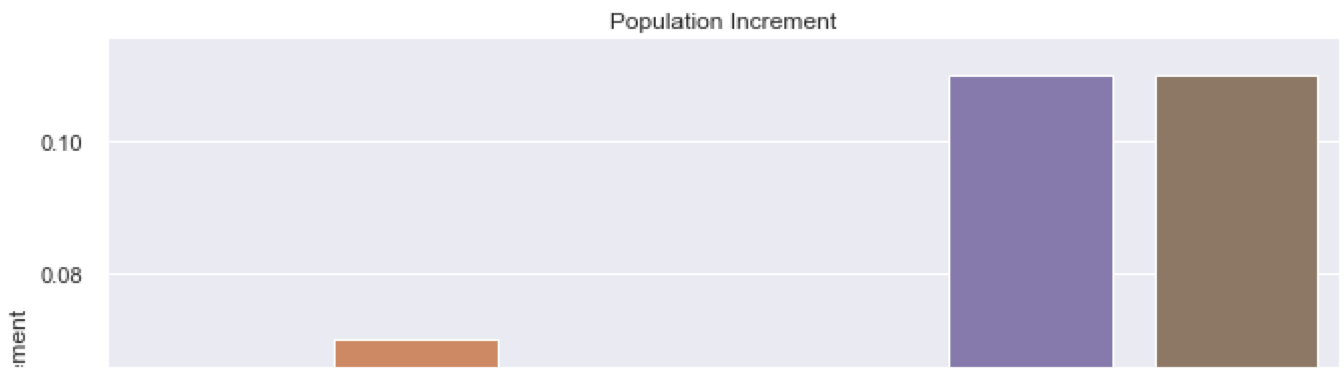
	cluster_label	cartodb_id	pop_11	pop_17	pop_inc	percent_pop_work_age	unemployment_rate	gross_annual_pay	active_busi
0	0	21.20	210036.60	221795.00	0.06	65.16	4.74	36884.60	11926.00
1	1	13.07	281572.13	301234.33	0.07	65.71	6.59	31795.27	13676.67
2	2	21.67	216135.33	220665.33	0.01	71.47	4.87	37755.39	15713.33
3	3	7.00	7412.00	7654.00	0.03	73.10	6.20	34161.16	26130.00
4	4	19.25	256905.25	285383.12	0.11	72.65	6.07	34549.75	18484.38
5	5	33.00	219582.00	244796.00	0.11	72.30	8.80	42141.00	55385.00

3.2.1 Growing population

Which cluster shows higher growth rates?

In [93]:

```
pop = sns.barplot(x='cluster_label', y='pop_inc', data=df_cluster)
pop.set_title('Population Increment')
pop.set_xlabel('Clusters')
pop.set_ylabel('Population Increment')
sns.despine(left=True)
```



3.2.2 Working population

Which cluster show lower unemployment rates?

In [98]:

```
pop = sns.barplot(x='cluster_label', y='unemployment_rate', data=df_cluster)
pop.set_title('Unemployment rate')
pop.set_xlabel('Clusters')
pop.set_ylabel('Unemployment rate')
sns.despine(left=True)
```

3.2.3 Mid-high income

Which cluster show higher gross annual income?

In [96]:
```
pop = sns.barplot(x='cluster_label', y='gross_annual_pay', data=df_cluster)
pop.set_title('Gross annual pay')
pop.set_xlabel('Clusters')
pop.set_ylabel('Gross annual pay')
sns.despine(left=True)
```

3.2.4 Crime rates

Which cluster show lower crime rates?

In [94]:

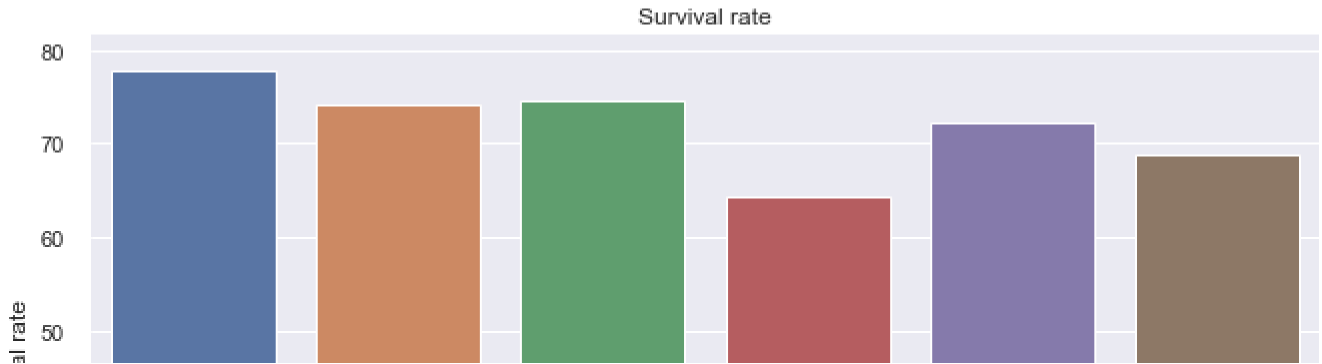
```
pop = sns.barplot(x='cluster_label', y='crime_rate_per_thousand', data=df_cluster)
pop.set_title('Crimes per thousand')
pop.set_xlabel('Clusters')
pop.set_ylabel('Crimes per thousand')
sns.despine(left=True)
```

3.2.5 Business survival rates?

Which cluster show higher survival rate for businesses after two years of activity?

In [95]:

```
pop = sns.barplot(x='cluster_label', y='two_year_survival_rate', data=df_cluster)
pop.set_title('Survival rate')
pop.set_xlabel('Clusters')
pop.set_ylabel('Two-year survival rate')
sns.despine(left=True)
```



3.3 What cluster fits the requirements?

Result Once analysed we would recommend opening the business in cluster 4

4. Building Visualizations

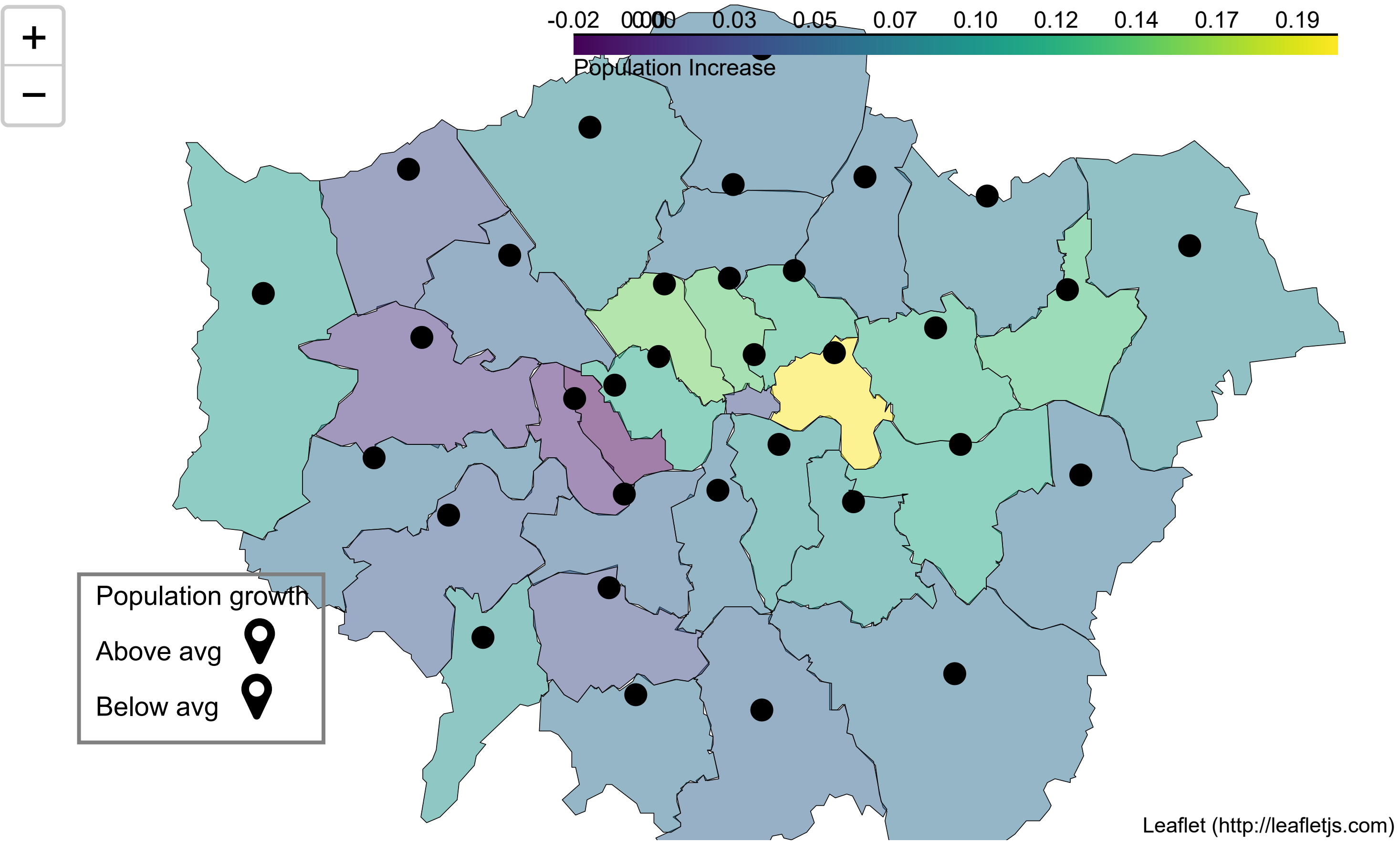
4.1 What boroughs are growing in population?

In [100]:

```
lat = 51.510067
lng = -0.133869
zoom_start = 10
avg_inc = np.mean(df.pop_inc)

m = folium.Map(location=[lat, lng], zoom_start=zoom_start) # generate map centred around
the Conrad Hotel
ft = 'pop_inc'
cmap = cm.linear.viridis.scale(df[ft].min(), df[ft].max())
folium.GeoJson(df, style_function=lambda feature: {
             'fillColor': cmap(feature['properties'][ft]),
             'fillOpacity' : 0.5,
             'weight' : 0.4,
            'color' : 'black'


}).add_to(m)

for row in df.itertuples():
    m.add_child(folium.Marker(location=[row.geometry.centroid.y, row.geometry.centroid.x
],
           popup=row.name, icon=folium.Icon(color='green' if row.pop_inc > avg_inc else
'purple', prefix='fa', icon='circle')))

cmap.caption = 'Population Increase'
m.add_child(cmap)

legend_html =   '''
                <div style="position: fixed;
                            bottom: 50px; left: 50px; width: 130px; height: 90px;
                            border:2px solid grey; z-index:9999; font-size:14px; backgro
und-color:white;
                            ">  Population growth <br>
                                Above avg   <i class="fa fa-map-marker fa-2x"
 style="color:green"></i><br>
                                Below avg   <i class="fa fa-map-marker fa-2x"
```

```
 style="color:purple"></i>
                </div>
                '''
m.get_root().html.add_child(folium.Element(legend_html))

m
```

Out[100]:

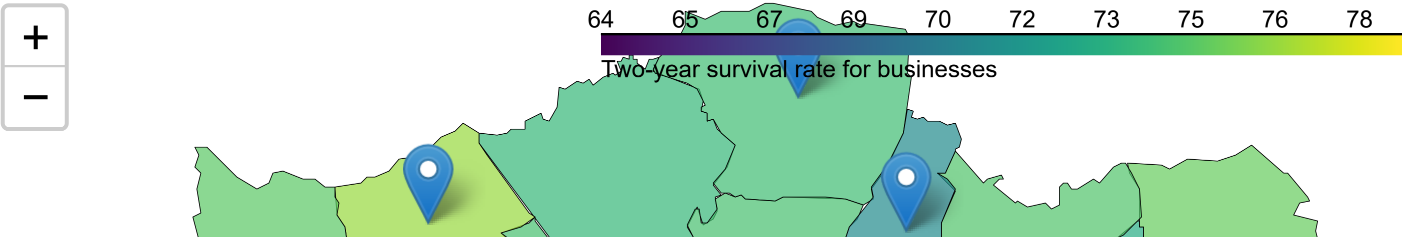


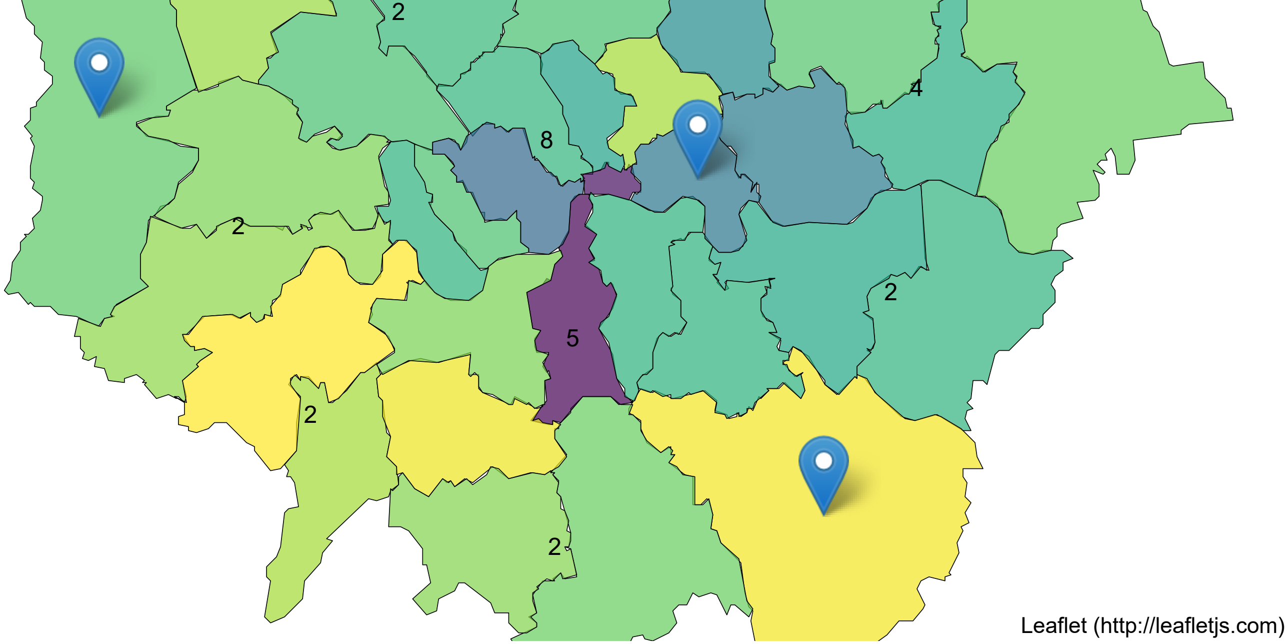
4.2 What boroughs show higher business survival rates?

In [55]:

```python
lat = 51.510067
lng = -0.133869
zoom_start = 10


m = folium.Map(location=[lat, lng], zoom_start=zoom_start) # generate map centred around the Conrad Hotel
ft = 'two_year_survival_rate'
cmap = cm.linear.viridis.scale(df[ft].min(), df[ft].max())
folium.GeoJson(df, style_function=lambda feature: {
             'fillColor': cmap(feature['properties'][ft]),
             'fillOpacity' : 0.7,
             'weight' : 0.4,
            'color' : 'black'



}).add_to(m)

marker = MarkerCluster().add_to(m)
for ix, row in df.iterrows():
    text = "Borough: " + str(row['name']) + "<br>" + ft + ': '+ str(row[ft]) + "<br>" + "Population: " + str(row['pop_17'])
    popup = folium.Popup(IFrame(text, width=300, height=100))
    folium.Marker(location = [row.geometry.centroid.y ,row.geometry.centroid.x], popup=popup).add_to(marker)

cmap.caption = 'Two-year survival rate for businesses'
m.add_child(cmap)
m
```

Out[55]:



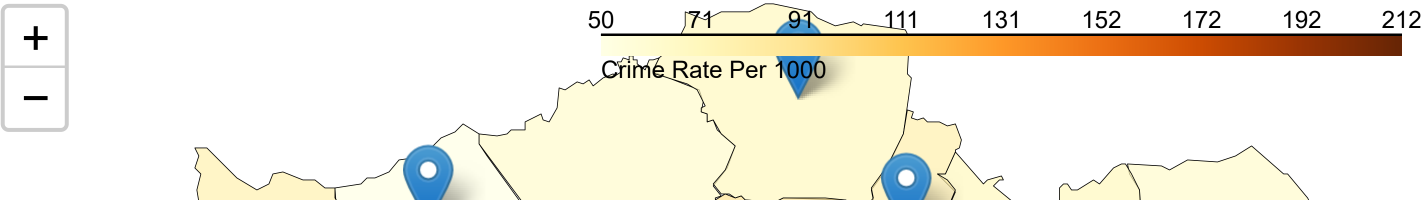


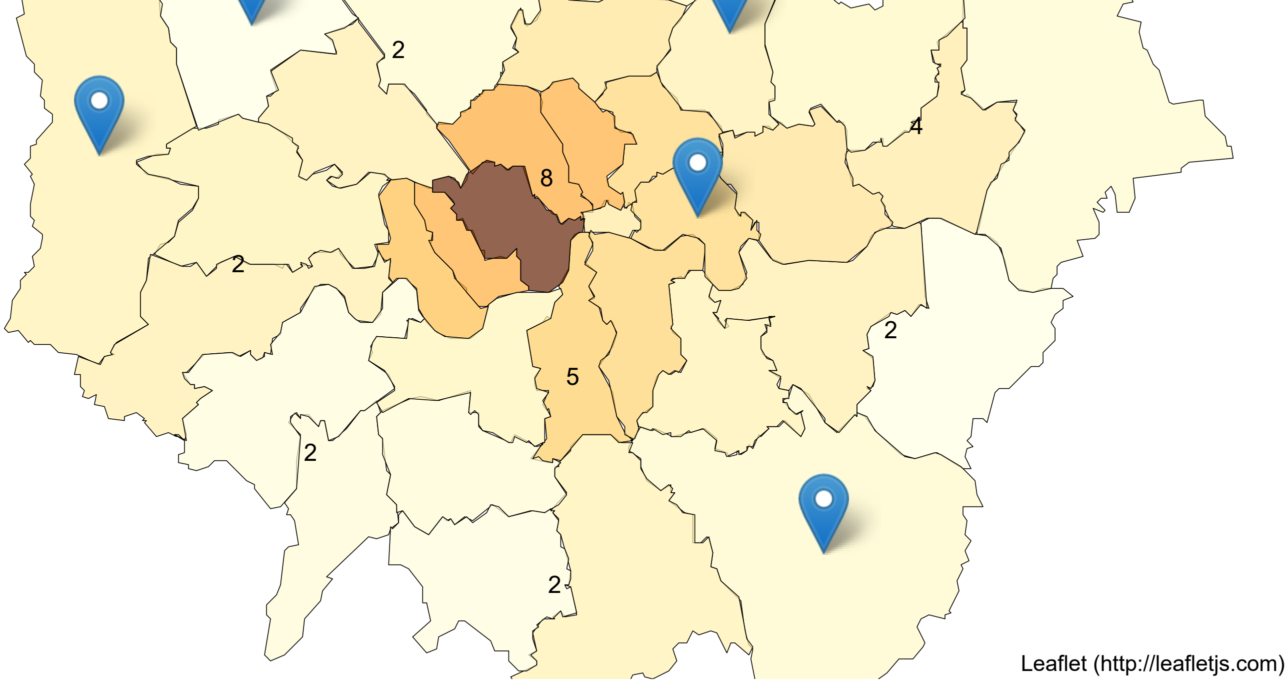
4.3 What boroughs show higher crime rates?

In [56]:

```
lat = 51.510067
lng = -0.133869
zoom_start = 10


m = folium.Map(location=[lat, lng], zoom_start=zoom_start) # generate map centred around
the Conrad Hotel
ft = 'crime_rate_per_thousand'
cmap = cm.linear.YlOrBr_09.scale(df[ft].min(), df[ft].max())
folium.GeoJson(df, style_function=lambda feature: {
             'fillColor': cmap(feature['properties'][ft]),
             'fillOpacity' : 0.7,
             'weight' : 0.4,
            'color' : 'black'


}).add_to(m)

marker = MarkerCluster().add_to(m)
for ix, row in df.iterrows():
    text = "Borough: " + str(row['name']) + "<br>" + ft + ': '+ str(row[ft]) + "<br>" +
"Population: " + str(row['pop_17'])
    popup = folium.Popup(IFrame(text, width=300, height=100))
    folium.Marker(location = [row.geometry.centroid.y ,row.geometry.centroid.x], popup=p
opup).add_to(marker)

cmap.caption = 'Crime Rate Per 1000'
m.add_child(cmap)

m
```

Out[56]:



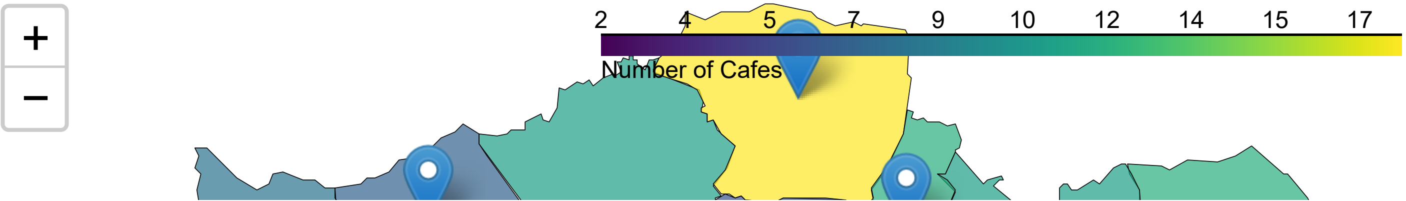


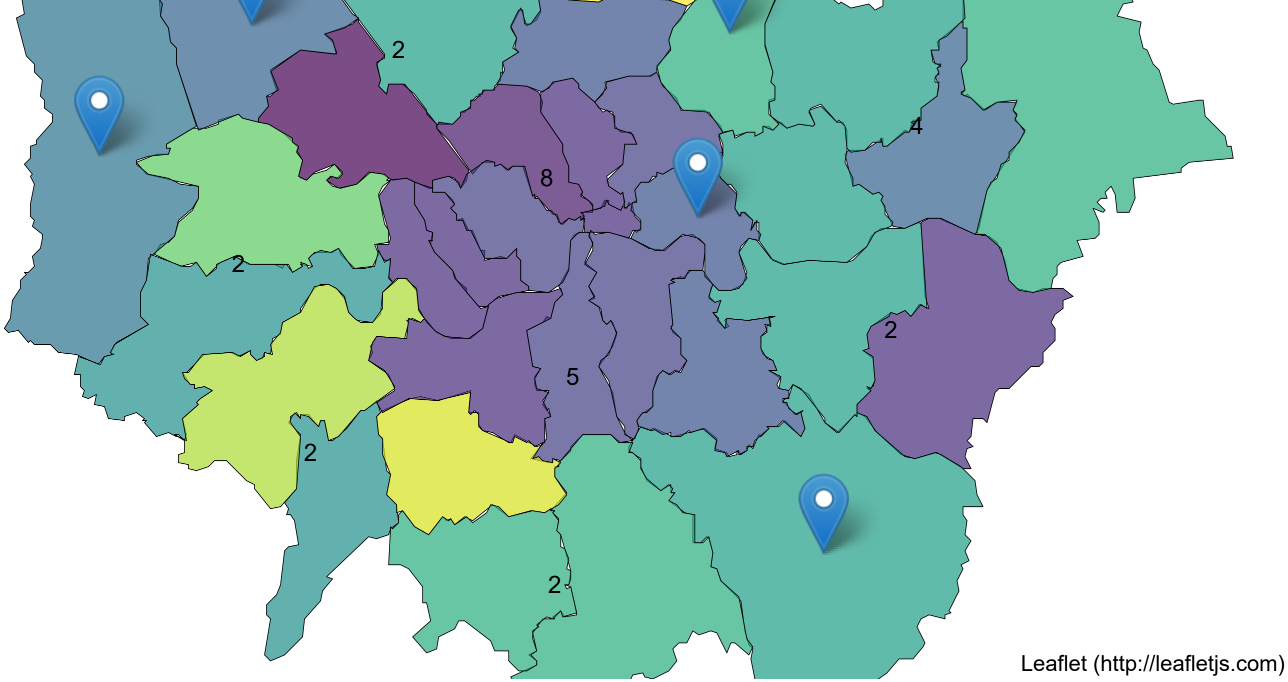
4.4 What boroughs have more cafes?

In [57]:

```
lat = 51.510067
lng = -0.133869
zoom_start = 10


m = folium.Map(location=[lat, lng], zoom_start=zoom_start) # generate map centred around the Conrad Hotel
ft = 'Cafes'
cmap = cm.linear.viridis.scale(df[ft].min(), df[ft].max())
folium.GeoJson(df, style_function=lambda feature: {
             'fillColor': cmap(feature['properties'][ft]),
             'fillOpacity' : 0.7,
             'weight' : 0.4,
            'color' : 'black'


}).add_to(m)

marker = MarkerCluster().add_to(m)
for ix, row in df.iterrows():
    text = "Borough: " + str(row['name']) + "<br>" + ft + ': '+ str(row[ft]) + "<br>" + "Population: " + str(row['pop_17'])
    popup = folium.Popup(IFrame(text, width=300, height=100))
    folium.Marker(location = [row.geometry.centroid.y ,row.geometry.centroid.x], popup=popup).add_to(marker)

cmap.caption = 'Number of Cafes'
m.add_child(cmap)

m
```

Out[57]:



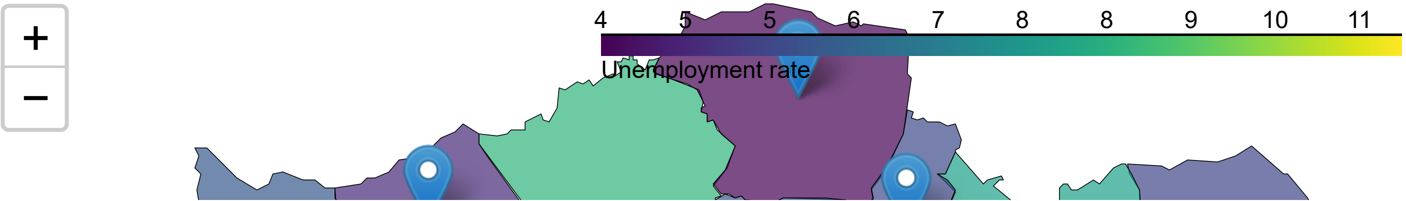


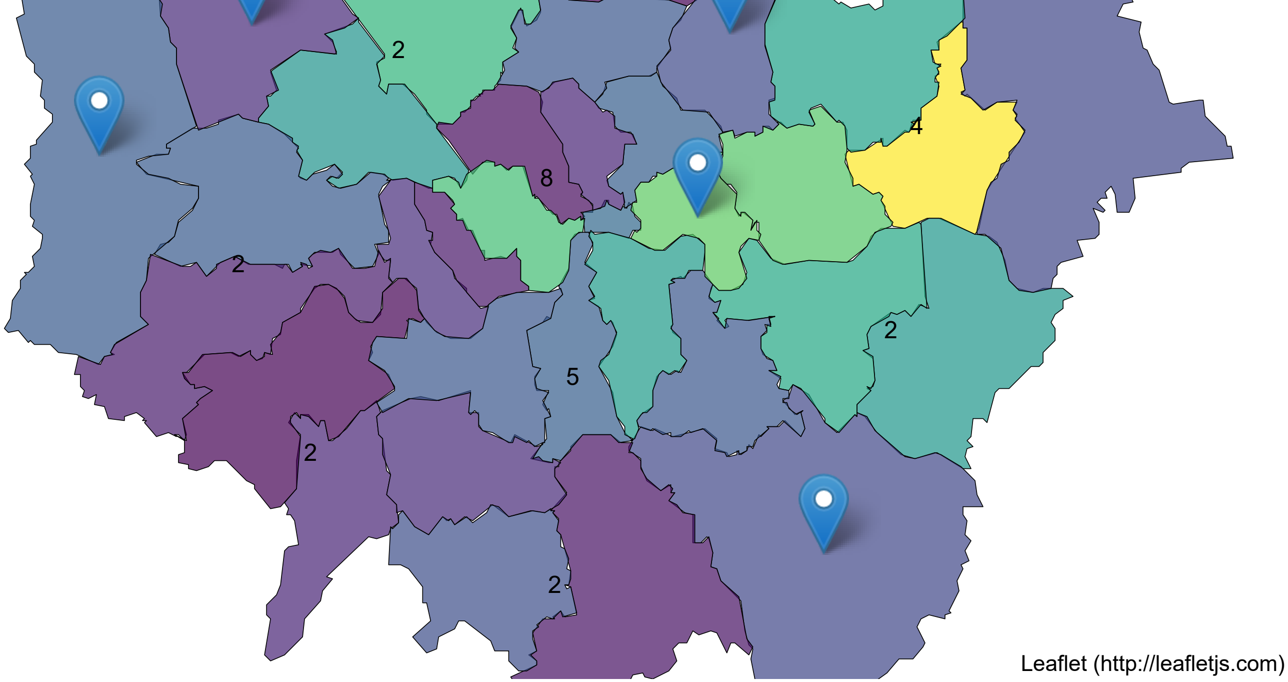
4.5 What boroughs show lower unemployment rate?

In [58]:

```
lat = 51.510067
lng = -0.133869
zoom_start = 10


m = folium.Map(location=[lat, lng], zoom_start=zoom_start) # generate map centred around the Conrad Hotel
ft = 'unemployment_rate'
cmap = cm.linear.viridis.scale(df[ft].min(), df[ft].max())
folium.GeoJson(df, style_function=lambda feature: {
             'fillColor': cmap(feature['properties'][ft]),
             'fillOpacity' : 0.7,
             'weight' : 0.4,
            'color' : 'black'


}).add_to(m)

marker = MarkerCluster().add_to(m)
for ix, row in df.iterrows():
    text = "Borough: " + str(row['name']) + "<br>" + ft + ': '+ str(row[ft]) + "<br>" + "Population: " + str(row['pop_17'])
    popup = folium.Popup(IFrame(text, width=300, height=100))
    folium.Marker(location = [row.geometry.centroid.y ,row.geometry.centroid.x], popup=popup).add_to(marker)

cmap.caption = 'Unemployment rate'
m.add_child(cmap)

m
```

Out[58]:

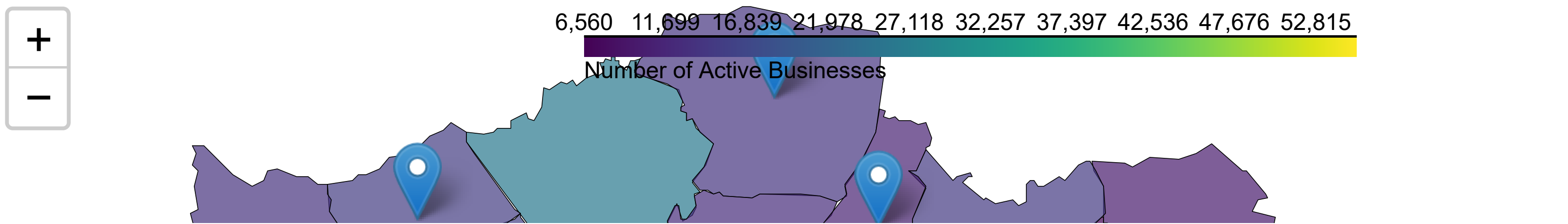


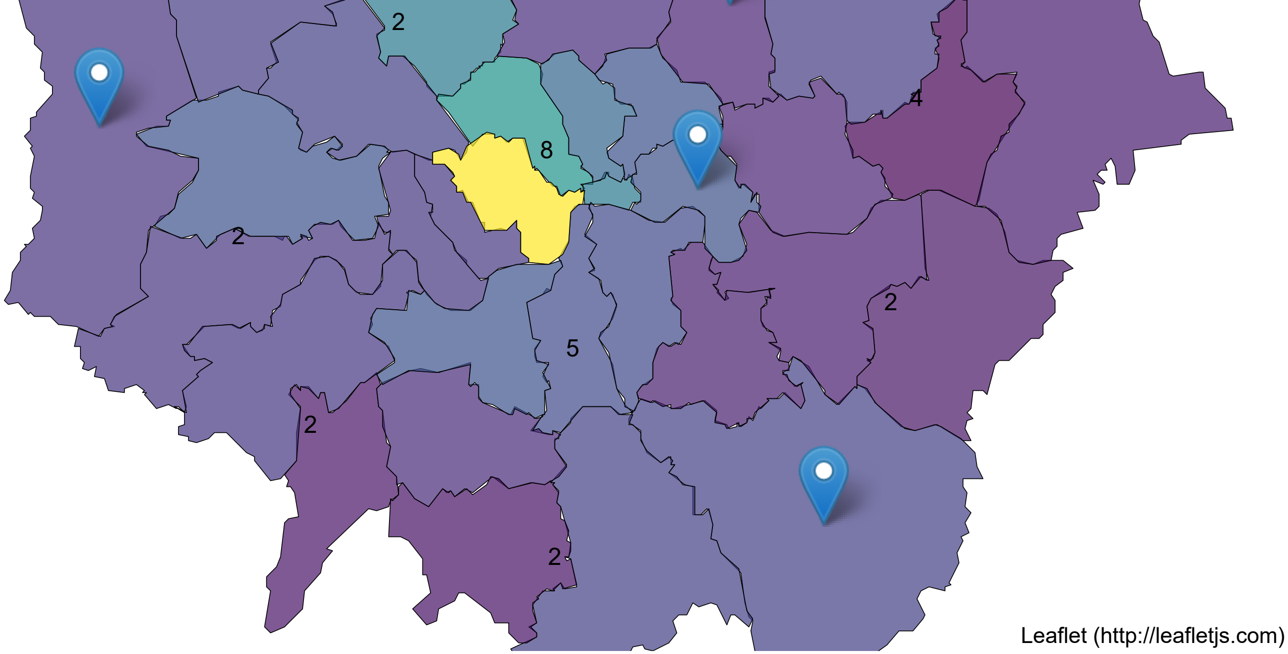

2
4
8
2
2
5
2
2
Leaflet (http://leafletjs.com)

In [59]:

```python
lat = 51.510067
lng = -0.133869
zoom_start = 10


m = folium.Map(location=[lat, lng], zoom_start=zoom_start) # generate map centred around the Conrad Hotel
ft = 'active_businesses'
cmap = cm.linear.viridis.scale(df[ft].min(), df[ft].max())
folium.GeoJson(df, style_function=lambda feature: {
             'fillColor': cmap(feature['properties'][ft]),
             'fillOpacity' : 0.7,
             'weight' : 0.4,
            'color' : 'black'



}).add_to(m)

marker = MarkerCluster().add_to(m)
for ix, row in df.iterrows():
    text = "Borough: " + str(row['name']) + "<br>" + ft + ': '+ str(row[ft]) + "<br>" + "Population: " + str(row['pop_17'])
    popup = folium.Popup(IFrame(text, width=300, height=100))
    folium.Marker(location = [row.geometry.centroid.y ,row.geometry.centroid.x], popup=popup).add_to(marker)

cmap.caption = 'Number of Active Businesses'
m.add_child(cmap)

m
```

Out[59]:




2
8
4
2
2
5
2
2
2
Leaflet (http://leafletjs.com)