Capstone Project: The Battle of Neighbourhoods

By: Milena Shehu

Table of Contents

1. Introduction

1.1 Description of the Problem

The population of London has grown considerably over the last decades. London is very diverse. It represents what is called the reflection of the old British Empire. In London, you can get fresh from food supplies from Africa. One begins to wonder the efficiency of the supply mechanism.

1.2 Discussion of the Background

Let's suppose that a successful restaurant chain in Africa is looking to expand operation into Europe through London. They want to create a high-end restaurant that comes with organic mix and healthy. Their target is not only West Africans, but they are pro-organic and healthy eating. To them every meal counts and counts as a royal when you eat.

1.3 Target Audience

Considering the diversity of London, there is a high multicultural sense. As such, in the search for an high-end African-inclined restaurant, there is a high shortage.

2. Data

2.1 Description of Data

This project will rely on public data from Wikipedia and Foursquare.

Dataset 1:

The London Area consists of 32 Boroughs and the "City of London". Our data will be from the link - Greater London Area <https://en.wikipedia.org/wiki/List_of_areas_of_London (https://en.wikipedia.org/wiki/List_of_areas_of_London) >

The web scrapped of the Wikipedia page for the Greater London Area data is provided below:

In [4]:
```
# library for BeautifulSoup
from bs4 import BeautifulSoup

# library to handle data in a vectorized manner
import numpy as np

# library for data analsysis
import pandas as pd
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)

# library to handle JSON files
import json
print('numpy, pandas, ..., imported...')

!pip -q install geopy
# conda install -c conda-forge geopy --yes # uncomment this line if you have
n't completed the Foursquare API lab
print('geopy installed...')
# convert an address into latitude and longitude values
from geopy.geocoders import Nominatim
print('Nominatim imported...')

# library to handle requests
import requests
print('requests imported...')

# tranform JSON file into a pandas dataframe
from pandas.io.json import json_normalize
print('json_normalize imported...')

# Matplotlib and associated plotting modules
import matplotlib.cm as cm
import matplotlib.colors as colors
print('matplotlib imported...')

# import k-means from clustering stage
from sklearn.cluster import KMeans
print('Kmeans imported...')

# install the Geocoder
!pip -q install geocoder
import geocoder

# import time
import time

# !conda install -c conda-forge folium=0.5.0 --yes # uncomment this line if yo
u haven't completed the Foursquare API lab
!pip -q install folium
print('folium installed...')
import folium # map rendering library
print('folium imported...')
print('...Done')
```

```
numpy, pandas, ..., imported...
geopy installed...
Nominatim imported...
requests imported...
json_normalize imported...
matplotlib imported...
Kmeans imported...
folium installed...
folium imported...
...Done
```

In [5]:
```
wikipedia_link = 'https://en.wikipedia.org/wiki/List_of_areas_of_London'
headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:64.0) Gecko/20100101 Firefox/64.0'}
wikipedia_page = requests.get(wikipedia_link, headers = headers)
wikipedia_page
```

Out[5]:
```
<Response [200]>
```

In [6]:
```
# Cleans html file
soup = BeautifulSoup(wikipedia_page.content, 'html.parser')
# This extracts the "tbody" within the table where class is "wikitable sortable"
table = soup.find('table', {'class':'wikitable sortable'}).tbody
```

In [7]:
```
# Extracts all "tr" (table rows) within the table above
rows = table.find_all('tr')
```

In [8]:
```
# Extracts the column headers, removes and replaces possible '\n' with space for the "th" tag
columns = [i.text.replace('\n', '')
           for i in rows[0].find_all('th')]
```

In [9]:
```
# Converts columns to pd dataframe
df = pd.DataFrame(columns = columns)
df
```

Out[9]:

Location	London borough	Post town	Postcode district	Dial code	OS grid ref

In [10]:
```
# Extracts every row with corresponding columns
# Then appends the values to the create pd dataframe "df"
# Please note that the first row (row[0]) is skipped because it is already the header
for i in range(1, len(rows)):
    tds = rows[i].find_all('td')


    if len(tds) == 7:
        values = [tds[0].text, tds[1].text, tds[2].text.replace('\n', ''.replace('\xa0','')), tds[3].text, tds[4].text.replace('\n', ''.replace('\xa0','')), tds[5].text.replace('\n', ''.replace('\xa0','')), tds[6].text.replace('\n', ''.replace('\xa0',''))]
    else:
        values = [td.text.replace('\n', '').replace('\xa0','') for td in tds]

        df = df.append(pd.Series(values, index = columns), ignore_index = True)

        df
```

In [12]:
```
df.head(5)
```

Out[12]:

	Location	London borough	Post town	Postcode district	Dial code	OS grid ref
0	Abbey Wood	Bexley, Greenwich [7]	LONDON	SE2	020	TQ465785
1	Acton	Ealing, Hammersmith and Fulham[8]	LONDON	W3, W4	020	TQ205805
2	Addington	Croydon[8]	CROYDON	CR0	020	TQ375645
3	Addiscombe	Croydon[8]	CROYDON	CR0	020	TQ345665
4	Albany Park	Bexley	BEXLEY, SIDCUP	DA5, DA14	020	TQ478728

In [13]:
```
df = df.rename(index=str, columns = {'Location': 'Location', 'London\xa0borough': 'Borough', 'Post town': 'Post-town', 'Postcode\xa0district': 'Postcode', 'Dial\xa0code': 'Dial-code', 'OS grid ref': 'OSGridRef'})
```

In [14]:
```
df.head(5)
```

Out[14]:

	Location	Borough	Post-town	Postcode	Dial-code	OSGridRef
0	Abbey Wood	Bexley, Greenwich [7]	LONDON	SE2	020	TQ465785
1	Acton	Ealing, Hammersmith and Fulham[8]	LONDON	W3, W4	020	TQ205805
2	Addington	Croydon[8]	CROYDON	CR0	020	TQ375645
3	Addiscombe	Croydon[8]	CROYDON	CR0	020	TQ345665
4	Albany Park	Bexley	BEXLEY, SIDCUP	DA5, DA14	020	TQ478728

Looking the data, under the Borough, there are borough names with []. These are references extracted from the wiki page. So remove these, the following was done:

In [15]:
```
df['Borough'] = df['Borough'].map(lambda x: x.rstrip(']').rstrip('0123456789')
.rstrip('['))
```

In [16]:
```
df.shape
```

Out[16]:
```
(533, 6)
```

In [17]:
```
df.head(5)
```

Out[17]:

	Location	Borough	Post-town	Postcode	Dial-code	OSGridRef
0	Abbey Wood	Bexley, Greenwich	LONDON	SE2	020	TQ465785
1	Acton	Ealing, Hammersmith and Fulham	LONDON	W3, W4	020	TQ205805
2	Addington	Croydon	CROYDON	CR0	020	TQ375645
3	Addiscombe	Croydon	CROYDON	CR0	020	TQ345665
4	Albany Park	Bexley	BEXLEY, SIDCUP	DA5, DA14	020	TQ478728

Assumption 1: Where the Postcode are more than one, (for example, in Acton, there are 2 postcodes - W3 and W4), the postcodes are spread to multi-rows and assigned the same values from the other columns.

In [18]:
```
df0 = df.drop('Postcode', axis=1).join(df['Postcode'].str.split(',', expand=Tr
ue).stack().reset_index(level=1, drop=True).rename('Postcode'))
```

In [19]:
```
df0.head(5)
```

Out[19]:

	Location	Borough	Post-town	Dial-code	OSGridRef	Postcode
0	Abbey Wood	Bexley, Greenwich	LONDON	020	TQ465785	SE2
1	Acton	Ealing, Hammersmith and Fulham	LONDON	020	TQ205805	W3
1	Acton	Ealing, Hammersmith and Fulham	LONDON	020	TQ205805	W4
10	Angel	Islington	LONDON	020	TQ345665	EC1
10	Angel	Islington	LONDON	020	TQ345665	N1

In [20]:
```
df0.shape
```

Out[20]:
```
(637, 6)
```

Assumption 2: From the data, only the 'Location', 'Borough', 'Postcode', 'Post-town' will be used for this project. So they are extracted into a new data frame.

In [21]:
```
df1 = df0[['Location', 'Borough', 'Postcode', 'Post-town']].reset_index(drop=True)
```

In [22]:
```
df1.head(5)
```

Out[22]:

	Location	Borough	Postcode	Post-town
0	Abbey Wood	Bexley, Greenwich	SE2	LONDON
1	Acton	Ealing, Hammersmith and Fulham	W3	LONDON
2	Acton	Ealing, Hammersmith and Fulham	W4	LONDON
3	Angel	Islington	EC1	LONDON
4	Angel	Islington	N1	LONDON

In [23]:
```
df1.shape
```

Out[23]:
```
(637, 4)
```

Assumption 3: Now, only the Boroughs with London Post-town will be used for our search of location. Therefore, all the non-post-town are dropped.

In [24]:
```
df2 = df1
df21 = df2[df2['Post-town'].str.contains('LONDON')]
```

In [25]:
```
df21.head(5)
```

Out[25]:

	Location	Borough	Postcode	Post-town
0	Abbey Wood	Bexley, Greenwich	SE2	LONDON
1	Acton	Ealing, Hammersmith and Fulham	W3	LONDON
2	Acton	Ealing, Hammersmith and Fulham	W4	LONDON
3	Angel	Islington	EC1	LONDON
4	Angel	Islington	N1	LONDON

In [26]:
```
df21.shape
```

Out[26]:
```
(381, 4)
```

From assumption 3, there are now 380 instances, which is a drop from 638 because of the drop of non-London post-towns.

In [27]:
```
# Re-assigns the df21 to new dataframe without the Post-town
df3 = df21[['Location', 'Borough', 'Postcode']].reset_index(drop=True)
```

In [28]:
```
df3.head(10)
```

Out[28]:

	Location	Borough	Postcode
0	Abbey Wood	Bexley, Greenwich	SE2
1	Acton	Ealing, Hammersmith and Fulham	W3
2	Acton	Ealing, Hammersmith and Fulham	W4
3	Angel	Islington	EC1
4	Angel	Islington	N1
5	Church End	Brent	NW10
6	Church End	Barnet	N3
7	Clapham	Lambeth, Wandsworth	SW4
8	Clerkenwell	Islington	EC1
9	Colindale	Barnet	NW9

In [29]:
```
df_london = df3
df_london.to_csv('LondonLocations.csv', index = False)
```

Assumption 4: Due to its more diverse outlook, proximity to afro-caribbean markets and accessible facilities, only the South East areas of London will be considered for our analysis. The South East areas has postcodes starting with SE.

So, first, we remove the whitesapaces at the start of some of the postcodes and then drop the other non-SE postcodes.

In [30]:
```
df_london.head(5)
```

Out[30]:

	Location	Borough	Postcode
0	Abbey Wood	Bexley, Greenwich	SE2
1	Acton	Ealing, Hammersmith and Fulham	W3
2	Acton	Ealing, Hammersmith and Fulham	W4
3	Angel	Islington	EC1
4	Angel	Islington	N1

In [31]:
```
df_london.Postcode = df_london.Postcode.str.strip()
```

In [32]:
```
df_london.head(5)
```

Out[32]:

	Location	Borough	Postcode
0	Abbey Wood	Bexley, Greenwich	SE2
1	Acton	Ealing, Hammersmith and Fulham	W3
2	Acton	Ealing, Hammersmith and Fulham	W4
3	Angel	Islington	EC1
4	Angel	Islington	N1

In [86]:
```
# New dataframe for South East London postcodes - df_se
df_se = df_london[df_london['Postcode'].str.startswith(('SE'))].reset_index(dr
op=True)
```

In [87]:
```
df_se.head(10)
```

Out[87]:

	Location	Borough	Postcode
0	Abbey Wood	Bexley, Greenwich	SE2
1	Crofton Park	Lewisham	SE4
2	Crossness	Bexley	SE2
3	Crystal Palace	Bromley	SE19
4	Crystal Palace	Bromley	SE20
5	Crystal Palace	Bromley	SE26
6	Denmark Hill	Southwark	SE5
7	Deptford	Lewisham	SE8
8	Dulwich	Southwark	SE21
9	East Dulwich	Southwark	SE22

Now,data is ready. df_se is the data we will focus on.

Assumption 5: This assumption will focus on the demography of London where there are predominantly more multicultural groups. The top 5 Black Africans are shown below:

In [88]:
```
demograph_link = 'https://en.wikipedia.org/wiki/Demography_of_London'
headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:64.0) G
ecko/20100101 Firefox/64.0'}
demograph_page = requests.get(demograph_link, headers = headers)
soup1 = BeautifulSoup(demograph_page.content, 'html.parser')
table1 = soup1.find('table', {'class':'wikitable sortable'}).tbody
rows1 = table1.find_all('tr')
columns1 = [i.text.replace('\n', '')
           for i in rows1[0].find_all('th')]
```

In [89]:
```
columns1
```

Out[89]:
```
['Local authority', 'White', 'Mixed', 'Asian', 'Black', 'Other']
```

In [90]:
```
demo_london = pd.DataFrame(columns = columns1)
```

In [91]:
```
demo_london.head(5)
```

Out[91]:

	Local authority	White	Mixed	Asian	Black	Other

In [92]:
```
for j in range(1, len(rows1)):
    tds1 = rows1[j].find_all('td')
    if len(tds1) == 7:
        values1 = [tds1[0].text, tds1[1].text, tds1[2].text.replace('\n', ''.r
eplace('\xa0','')), tds1[3].text, tds1[4].text.replace('\n', ''.replace('\xa0'
,'')), tds1[5].text.replace('\n', ''.replace('\xa0',''))]
    else:
        values1 = [td1.text.replace('\n', '').replace('\xa0','') for td1 in td
s1]

        demo_london = demo_london.append(pd.Series(values1, index = columns1),
ignore_index = True)

        demo_london
```

In [93]:
```
demo_london['Black'] = demo_london['Black'].astype('float')
```

In [94]:
```
demo_london_sorted = demo_london.sort_values(by='Black', ascending = False)
```

In [95]:
```
demo_london_sorted.head(5)
```

Out[95]:

	Local authority	White	Mixed	Asian	Black	Other
22	Lewisham	53.5	7.4	9.3	27.2	2.6
27	Southwark	54.3	6.2	9.4	26.9	3.3
21	Lambeth	57.1	7.6	6.9	25.9	2.4
11	Hackney	54.7	6.4	10.5	23.1	5.3
7	Croydon	55.1	6.6	16.4	20.2	1.8

Assumption 6: Our next assumption will be based on the top 5 areas will significantly high "Black", "Mixed" and other races. These leaves us with Lewisham, Southwark, Lambeth, Hackney and Croydon.

In [96]:
```
df_se
```

Out[96]:

	Location	Borough	Postcode
0	Abbey Wood	Bexley, Greenwich	SE2
1	Crofton Park	Lewisham	SE4
2	Crossness	Bexley	SE2
3	Crystal Palace	Bromley	SE19
4	Crystal Palace	Bromley	SE20
5	Crystal Palace	Bromley	SE26
6	Denmark Hill	Southwark	SE5
7	Deptford	Lewisham	SE8
8	Dulwich	Southwark	SE21
9	East Dulwich	Southwark	SE22
10	Elephant and Castle	Southwark	SE1
11	Elephant and Castle	Southwark	SE11
12	Elephant and Castle	Southwark	SE17
13	Eltham	Greenwich	SE9
14	Falconwood	Bexley, Greenwich	SE9
15	Bankside	Southwark	SE1
16	Forest Hill	Lewisham	SE23
17	Gipsy Hill	Lambeth	SE19
18	Gipsy Hill	Lambeth	SE27
19	Greenwich	Greenwich	SE10
20	Grove Park	Lewisham	SE12
21	Herne Hill	Lambeth	SE24
22	Hither Green	Lewisham	SE13
23	Honor Oak	Lewisham	SE23
24	Horn Park	Greenwich, Lewisham	SE12
25	Kennington	Lambeth, Southwark	SE11
26	Kidbrooke	Greenwich	SE3
27	Ladywell	Lewisham	SE4
28	Ladywell	Lewisham	SE13
29	Lambeth	Lambeth	SE1
30	Lee	Lewisham	SE12
31	Lewisham	Lewisham	SE13
32	Beckenham	Bromley	SE20
33	Longlands	Bexley	SE9
34	Maze Hill	Greenwich	SE10

	Location	Borough	Postcode
35	Middle Park	Greenwich	SE9
36	Mottingham	Bromley	SE9
37	New Cross	Lewisham	SE14
38	New Eltham	Greenwich	SE9
39	Newington	Southwark	SE1
40	Newington	Southwark	SE17
41	Nunhead	Southwark	SE15
42	Oval	Lambeth	SE11
43	Bellingham	Lewisham	SE6
44	Peckham	Southwark	SE15
45	Penge	Bromley	SE20
46	Plumstead	Greenwich	SE18
47	Rotherhithe	Southwark	SE16
48	Selhurst	Croydon	SE25
49	Shooter's Hill	Greenwich	SE18
50	Bermondsey	Southwark	SE1
51	South Norwood	Croydon	SE25
52	Southend	Lewisham	SE6
53	St Johns	Lewisham	SE4
54	Bexleyheath (also Bexley New Town)	Bexley	SE2
55	Surrey Quays	Southwark	SE16
56	Sydenham (also Lower Sydenham, Upper Sydenham)	Lewisham, Bromley	SE26
57	Sydenham Hill	Lewisham, Southwark	SE21
58	Sydenham Hill	Lewisham, Southwark	SE26
59	Thamesmead	Bexley, Greenwich	SE28
60	Thamesmead	Bexley, Greenwich	SE2
61	Tulse Hill	Lambeth	SE24
62	Tulse Hill	Lambeth	SE27
63	Upper Norwood	Croydon	SE19
64	Walworth	Southwark	SE17
65	Well Hall	Greenwich	SE9
66	Blackheath	Lewisham	SE3
67	West Heath	Bexley	SE2
68	Blackheath Royal Standard	Greenwich	SE3
69	Blackheath Royal Standard	Greenwich	SE12
70	West Norwood	Lambeth	SE27

	Location	Borough	Postcode
71	Westcombe Park	Greenwich	SE3
72	Woolwich	Greenwich	SE18
73	Brixton	Lambeth	SE5
74	Brockley	Lewisham	SE4
75	Camberwell	Southwark	SE5
76	Catford	Lewisham	SE6
77	Charlton	Greenwich	SE7
78	Anerley	Bromley	SE20
79	Chinbrook	Lewisham	SE12

In [44]:
```
df_se_top = df_se[df_se['Borough'].isin(['Lewisham', 'Southwark', 'Lambeth',
'Hackney', 'Croydon'])].reset_index(drop=True)
```

In [45]:
```
df_se_top.head(5)
```

Out[45]:

	Location	Borough	Postcode
0	Crofton Park	Lewisham	SE4
1	Denmark Hill	Southwark	SE5
2	Deptford	Lewisham	SE8
3	Dulwich	Southwark	SE21
4	East Dulwich	Southwark	SE22

In [47]:
```
df_se.shape
```

Out[47]: (80, 3)

We have our working dataframe to be df_se_top to work with.

Dataset 2:

In obtaining the location data of the locations, the Geocoder package is used with the arcgis_geocoder to obtain the latitude and longitude of the needed locations.

In [48]:
```
# Geocoder starts here
# Defining a function to use --> get_latlng()'''
def get_latlng(arcgis_geocoder):

    # Initialize the Location (lat. and long.) to "None"
    lat_lng_coords = None

    # While loop helps to create a continous run until all the location coordi
nates are geocoded
    while(lat_lng_coords is None):
        g = geocoder.arcgis('{}, London, United Kingdom'.format(arcgis_geocode
r))
        lat_lng_coords = g.latlng
    return lat_lng_coords
# Geocoder ends here
```

Testing the function above for a sample postcode - SE2.

In [49]:
```
sample = get_latlng('SE2')
sample
```

Out[49]:
```
[51.492450000000076, 0.12127000000003818]
```

And reverse geocoding this, using the geocodefarm geocoder, gives the following:

In [50]:
```
gg = geocoder.geocodefarm(sample, method = 'reverse')
gg
```

Out[50]:
```
<[OK] Geocodefarm - Reverse [Harrow Manor Way, London, SE2 9SW, United Kingdo
m]>
```

So, we are certain that the geocoder works fine. So we proceed to applying it to our dataframe df_se_top.

In [51]:
```
start = time.time()

postal_codes = df_se_top['Postcode']
coordinates = [get_latlng(postal_code) for postal_code in postal_codes.tolist
()]

end = time.time()
print("Time of execution: ", end - start, "seconds")
```

```
Time of execution:  23.727686405181885 seconds
```

Then we proceed to store the location data - latitude and longitude as follows. The obtained coordinates are then joined to df_se_topto create new data frame.

In [52]:
```
df_se_loc = df_se_top

# The obtained coordinates (latitude and longitude) are joined with the datafr
ame as shown
df_se_coordinates = pd.DataFrame(coordinates, columns = ['Latitude', 'Longitud
e'])
df_se_loc['Latitude'] = df_se_coordinates['Latitude']
df_se_loc['Longitude'] = df_se_coordinates['Longitude']
```

In [53]:
```
df_se_loc.head(5)
```

Out[53]:

	Location	Borough	Postcode	Latitude	Longitude
0	Crofton Park	Lewisham	SE4	51.46268	-0.03558
1	Denmark Hill	Southwark	SE5	51.47480	-0.09313
2	Deptford	Lewisham	SE8	51.48114	-0.02467
3	Dulwich	Southwark	SE21	51.44100	-0.08897
4	East Dulwich	Southwark	SE22	51.45256	-0.07076

In [54]:
```
df_se_loc.to_csv('SELondonLocationsCoordinates.csv', index = False)
```

In [55]:
```
df_se_loc.shape
```

Out[55]:
```
(46, 5)
```

Dataset 3:

Single Neighbourhood — An initial exploration of a single Neighbourhood within the London area was done to examine the Foursquare workability. The Lewisham Borough postcode SE13 and Location - Lewisham is used for this.

In [56]:
```
# Resets the current index to a new
se_df = df_se_loc.reset_index().drop('index', axis = 1)
se_df.loc[se_df['Location'] == 'Lewisham']
```

Out[56]:

	Location	Borough	Postcode	Latitude	Longitude
20	Lewisham	Lewisham	SE13	51.46196	-0.00754

Now, let’s use the Lewisham with the index location 20 as shown below:

In [57]:
```
lewisham_lat = se_df.loc[20, 'Latitude']
lewisham_long = se_df.loc[20, 'Longitude']
lewisham_loc = se_df.loc[20, 'Location']
lewisham_postcode = se_df.loc[20, 'Postcode']
print('The latitude and longitude values of {} with postcode {}, are {}, {}.'.
format(lewisham_loc, lewisham_postcode, lewisham_lat, lewisham_long))
```

```
The latitude and longitude values of Lewisham with postcode SE13, are 51.4619
6000000003, -0.007539999999949032.
```

Let's explore the top 100 venues that are within a 2000 metres radius of Lewisham. And then, let's create the GET request URL, and then the url is named. Since there is a limit to Foursquare usage → https://developer.foursquare.com/docs/api/troubleshooting/rate-limits (https://developer.foursquare.com/docs/api/troubleshooting/rate-limits)

3. Methodology

3.1 Data Exploration

An initial exploration of a single Neighbourhood within the London area was done to examine the Foursquare workability. The Lewisham Borough postcode SE13 and Location - Lewisham is used for this.

In [58]:
```
# Resets the current index to a new
se_df = df_se_loc.reset_index().drop('index', axis = 1)
```

In [60]:
```
se_df
```

Out[60]:

	Location	Borough	Postcode	Latitude	Longitude
0	Crofton Park	Lewisham	SE4	51.46268	-0.03558
1	Denmark Hill	Southwark	SE5	51.47480	-0.09313
2	Deptford	Lewisham	SE8	51.48114	-0.02467
3	Dulwich	Southwark	SE21	51.44100	-0.08897
4	East Dulwich	Southwark	SE22	51.45256	-0.07076
5	Elephant and Castle	Southwark	SE1	51.49960	-0.09613
6	Elephant and Castle	Southwark	SE11	51.49084	-0.11108
7	Elephant and Castle	Southwark	SE17	51.48764	-0.09542
8	Bankside	Southwark	SE1	51.49960	-0.09613
9	Forest Hill	Lewisham	SE23	51.44122	-0.04764
10	Gipsy Hill	Lambeth	SE19	51.41990	-0.08808
11	Gipsy Hill	Lambeth	SE27	51.43407	-0.10375
12	Grove Park	Lewisham	SE12	51.44759	0.01350
13	Herne Hill	Lambeth	SE24	51.45529	-0.09928
14	Hither Green	Lewisham	SE13	51.46196	-0.00754
15	Honor Oak	Lewisham	SE23	51.44122	-0.04764
16	Ladywell	Lewisham	SE4	51.46268	-0.03558
17	Ladywell	Lewisham	SE13	51.46196	-0.00754
18	Lambeth	Lambeth	SE1	51.49960	-0.09613
19	Lee	Lewisham	SE12	51.44759	0.01350
20	Lewisham	Lewisham	SE13	51.46196	-0.00754
21	New Cross	Lewisham	SE14	51.47489	-0.04038
22	Newington	Southwark	SE1	51.49960	-0.09613
23	Newington	Southwark	SE17	51.48764	-0.09542
24	Nunhead	Southwark	SE15	51.47218	-0.06779
25	Oval	Lambeth	SE11	51.49084	-0.11108
26	Bellingham	Lewisham	SE6	51.43722	-0.01868
27	Peckham	Southwark	SE15	51.47218	-0.06779
28	Rotherhithe	Southwark	SE16	51.49574	-0.05157
29	Selhurst	Croydon	SE25	51.39925	-0.07414
30	Bermondsey	Southwark	SE1	51.49960	-0.09613
31	South Norwood	Croydon	SE25	51.39925	-0.07414
32	Southend	Lewisham	SE6	51.43722	-0.01868
33	St Johns	Lewisham	SE4	51.46268	-0.03558
34	Surrey Quays	Southwark	SE16	51.49574	-0.05157

	Location	Borough	Postcode	Latitude	Longitude
35	Tulse Hill	Lambeth	SE24	51.45529	-0.09928
36	Tulse Hill	Lambeth	SE27	51.43407	-0.10375
37	Upper Norwood	Croydon	SE19	51.41990	-0.08808
38	Walworth	Southwark	SE17	51.48764	-0.09542
39	Blackheath	Lewisham	SE3	51.47138	0.02338
40	West Norwood	Lambeth	SE27	51.43407	-0.10375
41	Brixton	Lambeth	SE5	51.47480	-0.09313
42	Brockley	Lewisham	SE4	51.46268	-0.03558
43	Camberwell	Southwark	SE5	51.47480	-0.09313
44	Catford	Lewisham	SE6	51.43722	-0.01868
45	Chinbrook	Lewisham	SE12	51.44759	0.01350

In [61]:
```
se_df.loc[se_df['Location'] == 'Lewisham']
```

Out[61]:

	Location	Borough	Postcode	Latitude	Longitude
20	Lewisham	Lewisham	SE13	51.46196	-0.00754

In [62]:
```
se_df.loc[20, 'Location']
```

Out[62]:
```
'Lewisham'
```

In [63]:
```
lewisham_lat = se_df.loc[20, 'Latitude']
lewisham_long = se_df.loc[20, 'Longitude']
lewisham_loc = se_df.loc[20, 'Location']
lewisham_postcode = se_df.loc[20, 'Postcode']

print('The latitude and longitude values of {} with postcode {}, are {}, {}.'.
format(lewisham_loc,

lewisham_postcode,

lewisham_lat,

lewisham_long))
```

```
The latitude and longitude values of Lewisham with postcode SE13, are 51.4619
6000000003, -0.007539999999949032.
```

In [64]:
```
# Credentials are provided already for this part
LIMIT = 100 # limit of number of venues returned by Foursquare API
radius = 2000 # define radius
url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secre
t={}&v={}&ll={},{}&radius={}&limit={}'.format(
    '5VE0TJS5ZJC2YCCBP5R5UE5F0ZVFBHKHM2PK1DPDNSGKAQK3',
    '2R0VDFNSWDVUWEBXPR0YMJSYJYEF1WLJKWPGZ5TIZF41QRTU',
    '20180604',
    lewisham_lat,
    lewisham_long,
    radius,
    LIMIT)
# displays URL
url
```

Out[64]:
```
'https://api.foursquare.com/v2/venues/explore?&client_id=5VE0TJS5ZJC2YCCBP5R5
UE5F0ZVFBHKHM2PK1DPDNSGKAQK3&client_secret=2R0VDFNSWDVUWEBXPR0YMJSYJYEF1WLJKW
PGZ5TIZF41QRTU&v=20180604&ll=51.46196000000003,-0.007539999999949032&radius=2
000&limit=100'
```

In [ ]:
```
results = requests.get(url).json()
results
```

In [66]:
```
# function that extracts the category of the venue
def get_category_type(row):
    try:
        categories_list = row['categories']
    except:
        categories_list = row['venue.categories']

    if len(categories_list) == 0:
        return None
    else:
        return categories_list[0]['name']
```

The result will be structured pandas dataframe as shown below:

In [67]:
```
venues = results['response']['groups'][0]['items']

nearby_venues = json_normalize(venues) # flatten JSON

# filter columns
filtered_columns = ['venue.name', 'venue.categories', 'venue.location.lat', 'v
enue.location.lng']
nearby_venues =nearby_venues.loc[:, filtered_columns]

# filter the category for each row
nearby_venues['venue.categories'] = nearby_venues.apply(get_category_type, axi
s=1)

# clean columns
nearby_venues.columns = [col.split(".")[-1] for col in nearby_venues.columns]
```

```
/home/jupyterlab/conda/envs/python/lib/python3.6/site-packages/ipykernel_laun
cher.py:3: FutureWarning: pandas.io.json.json_normalize is deprecated, use pa
ndas.json_normalize instead
  This is separate from the ipykernel package so we can avoid doing imports u
ntil
```

In [68]:
```
nearby_venues_lewisham_unique = nearby_venues['categories'].value_counts().to_
frame(name='Count')
```

In [69]:
```
nearby_venues_lewisham_unique.head(5)
```

Out[69]:

	Count
Pub	13
Café	8
Gastropub	6
Park	5
Garden	4

Even though there are restaurants are the Lewisham area, they are not even in the top 5 venues. It should be noted that since we are limited by data availability, our perspectives will be on what we have.

In [70]:
```
print('{} venues were returned by Foursquare.'.format(nearby_venues.shape[0]))
```

```
100 venues were returned by Foursquare.
```

3.1.2 Multiple Neighbourhoods

Now let's explore (Multiple) Neighborhoods in the South East London area.

To do this, the function getNearbyVenues is used and it's created to repeat the same process for all neighborhoods.

In [71]:
```
def getNearbyVenues(names, latitudes, longitudes, radius=2000):

    venues_list=[]
    for name, lat, lng in zip(names, latitudes, longitudes):
        print(name)

        # create the API request URL
        url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&clie
nt_secret={}&v={}&ll={},{}&radius={}&limit={}'.format(
            '5VE0TJS5ZJC2YCCBP5R5UE5F0ZVFBHKHM2PK1DPDNSGKAQK3',
            '2R0VDFNSWDVUWEBXPR0YMJSYJYEF1WLJKWPGZ5TIZF41QRTU',
            '20180604',
            lat,
            lng,
            radius,
            LIMIT)

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

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

    nearby_venues = pd.DataFrame([item for venue_list in venues_list for item
in venue_list])
    nearby_venues.columns = ['Neighbourhood',
                  'Neighbourhood Latitude',
                  'Neighbourhood Longitude',
                  'Venue',
                  'Venue Latitude',
                  'Venue Longitude',
                  'Venue Category']

    return(nearby_venues)
```

The created function - getNearbyVenues is then used on each neighbourhoods. And creates a new dataframe called london_venues.

In [72]:
```
se_venues = getNearbyVenues(names=se_df['Location'],
                                    latitudes=se_df['Latitude'],
                                    longitudes=se_df['Longitude']
                                   )
```

```
Crofton Park
Denmark Hill
Deptford
Dulwich
East Dulwich
Elephant and Castle
Elephant and Castle
Elephant and Castle
Bankside
Forest Hill
Gipsy Hill
Gipsy Hill
Grove Park
Herne Hill
Hither Green
Honor Oak
Ladywell
Ladywell
Lambeth
Lee
Lewisham
New Cross
Newington
Newington
Nunhead
Oval
Bellingham
Peckham
Rotherhithe
Selhurst
Bermondsey
South Norwood
Southend
St Johns
Surrey Quays
Tulse Hill
Tulse Hill
Upper Norwood
Walworth
Blackheath
West Norwood
Brixton
Brockley
Camberwell
Catford
Chinbrook
```

In [74]:
```
len(se_venues)
```

Out[74]: 4237

In [75]:
```
se_venues['Neighbourhood'].value_counts()
se_venues.to_csv('se_venues.csv')
```

In [76]:
```
se_venues.head(5)
```

Out[76]:

	Neighbourhood	Neighbourhood Latitude	Neighbourhood Longitude	Venue	Venue Latitude	Venue Longitude	Venu Catego
0	Crofton Park	51.46268	-0.03558	The Orchard	51.463678	-0.035699	Gastropu
1	Crofton Park	51.46268	-0.03558	Brockley's Rock	51.459457	-0.033868	Fish Chi Sho
2	Crofton Park	51.46268	-0.03558	Browns Of Brockley	51.464513	-0.037346	Coffe Sho
3	Crofton Park	51.46268	-0.03558	Waterintobeer	51.463712	-0.038826	Beer Sto
4	Crofton Park	51.46268	-0.03558	Saka Maka	51.464826	-0.036437	India Restaura

The number of venues returned for each neighbourhoods is then explored as follows:

In [77]:
```
se_venues.groupby('Neighbourhood').count()
```

Out[77]:

Neighbourhood	Neighbourhood Latitude	Neighbourhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
Bankside	100	100	100	100	100	100
Bellingham	70	70	70	70	70	70
Bermondsey	100	100	100	100	100	100
Blackheath	89	89	89	89	89	89
Brixton	100	100	100	100	100	100
Brockley	100	100	100	100	100	100
Camberwell	100	100	100	100	100	100
Catford	70	70	70	70	70	70
Chinbrook	54	54	54	54	54	54
Crofton Park	100	100	100	100	100	100
Denmark Hill	100	100	100	100	100	100
Deptford	100	100	100	100	100	100
Dulwich	100	100	100	100	100	100
East Dulwich	80	80	80	80	80	80
Elephant and Castle	300	300	300	300	300	300
Forest Hill	100	100	100	100	100	100
Gipsy Hill	200	200	200	200	200	200
Grove Park	54	54	54	54	54	54
Herne Hill	100	100	100	100	100	100
Hither Green	100	100	100	100	100	100
Honor Oak	100	100	100	100	100	100
Ladywell	200	200	200	200	200	200
Lambeth	100	100	100	100	100	100
Lee	54	54	54	54	54	54
Lewisham	100	100	100	100	100	100
New Cross	100	100	100	100	100	100
Newington	200	200	200	200	200	200
Nunhead	100	100	100	100	100	100
Oval	100	100	100	100	100	100
Peckham	100	100	100	100	100	100
Rotherhithe	100	100	100	100	100	100
Selhurst	48	48	48	48	48	48
South Norwood	48	48	48	48	48	48

Neighbourhood	Neighbourhood Latitude	Neighbourhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
Southend	70	70	70	70	70	70
St Johns	100	100	100	100	100	100
Surrey Quays	100	100	100	100	100	100
Tulse Hill	200	200	200	200	200	200
Upper Norwood	100	100	100	100	100	100
Walworth	100	100	100	100	100	100
West Norwood	100	100	100	100	100	100

In [78]:
```
print('There are {} uniques categories.'.format(len(se_venues['Venue Category'].unique())))
```

```
There are 198 uniques categories.
```

In [79]:
```
se_venue_unique_count = se_venues['Venue Category'].value_counts().to_frame(name='Count')
```

In [80]:
```
se_venue_unique_count.head(5)
```

Out[80]:

	Count
Pub	433
Coffee Shop	301
Café	266
Park	198
Grocery Store	148

In [81]:
```
se_venue_unique_count.describe()
```

Out[81]:

	Count
count	198.000000
mean	21.398990
std	47.518069
min	1.000000
25%	4.000000
50%	8.000000
75%	19.000000
max	433.000000

For this section, the neighbourhoods in South East London will be clustered based on the processed data obtained above.

3.2 Clustering

Map Visualization

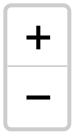
In [82]:
```
address = 'London, United Kingdom'

geolocator = Nominatim(user_agent="ln_explorer")
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude
print('The geograpical coordinate of London are {}, {}.'.format(latitude, long
itude))
```

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

In [99]:
```
# df_london_coordinates
map_london = folium.Map(location = [latitude, longitude], zoom_start = 12)
map_london
```

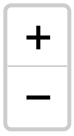
Out[99]:



Leaflet (http://leafletjs.com)

In [84]:
```
# Adding markers to map
for lat, lng, borough, loc in zip(se_df['Latitude'],
                                  se_df['Longitude'],
                                  se_df['Borough'],
                                  se_df['Location']):
    label = '{} - {}'.format(loc, 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).add_to(map_london)

display(map_london)
```



Leaflet (http://leafletjs.com)

In [85]:
```
type(se_df)
```

Out[85]:
```
pandas.core.frame.DataFrame
```

In [97]:
```
# one hot encoding
se_onehot = pd.get_dummies(se_venues[['Venue Category']], prefix = "", prefix_
sep = "")
```

In [98]:
```
# add neighborhood column back to dataframe
se_onehot['Neighbourhood'] = se_venues['Neighbourhood']
```

In [103]:
```
# move neighborhood column to the first column
fixed_columns = [se_onehot.columns[-1]] + list(se_onehot.columns[:-1])
se_onehot = se_onehot[fixed_columns]
```

In [ ]:
```
se_onehot.loc[se_onehot['African Restaurant'] != 0]
```

In [ ]:
```
se_onehot.loc[se_onehot['Neighbourhood'] == 'Lewisham']
```

In [106]:
```
se_onehot.to_csv('selondon_onehot.csv', index = False)
```

In [107]:
```
se_onehot.shape
```

Out[107]:
```
(4237, 199)
```

Regrouping and Category Statistics

In [109]:
```
se_grouped = se_onehot.groupby('Neighbourhood').mean().reset_index()
se_grouped.head()
```

Out[109]:

	Neighbourhood	Zoo Exhibit	African Restaurant	American Restaurant	Antique Shop	Aquarium	Argentinian Restaurant	Art Gallery	M
0	Bankside	0.000000	0.00	0.000000	0.0	0.0	0.010000	0.01	
1	Bellingham	0.000000	0.00	0.000000	0.0	0.0	0.000000	0.00	
2	Bermondsey	0.000000	0.00	0.000000	0.0	0.0	0.010000	0.01	
3	Blackheath	0.011236	0.00	0.011236	0.0	0.0	0.011236	0.00	
4	Brixton	0.000000	0.01	0.000000	0.0	0.0	0.000000	0.01	

In [110]:
```
print("Before One-hot encoding:", se_df.shape)
print("After One-hot encoding:", se_grouped.shape)
```

```
Before One-hot encoding: (46, 5)
After One-hot encoding: (40, 199)
```

In [111]:
```
se_grouped.to_csv('london_grouped.csv', index = False)
```

In [112]:
```
num_top_venues = 5 # Top common venues needed

for hood in se_grouped['Neighbourhood']:
    print("----"+hood+"----")
    temp = se_grouped[se_grouped['Neighbourhood'] == hood].T.reset_index()
    temp.columns = ['venue', 'freq']
    temp = temp.iloc[1:]
    temp['freq'] = temp['freq'].astype(float)
    temp = temp.round({'freq': 2})
    print(temp.sort_values('freq', ascending = False).reset_index(drop = True)
.head(num_top_venues))
    print('\n')
```

```
----Bankside----
                venue  freq
0         Coffee Shop  0.08
1                 Pub  0.06
2               Hotel  0.06
3  Italian Restaurant  0.05
4             Theater  0.04


----Bellingham----
           venue  freq
0  Grocery Store  0.13
1           Park  0.09
2    Supermarket  0.09
3           Café  0.06
4            Pub  0.06


----Bermondsey----
                venue  freq
0         Coffee Shop  0.08
1                 Pub  0.06
2               Hotel  0.06
3  Italian Restaurant  0.05
4             Theater  0.04


----Blackheath----
           venue  freq
0            Pub  0.15
1  Grocery Store  0.07
2           Park  0.06
3    Coffee Shop  0.06
4    Supermarket  0.04


----Brixton----
                       venue  freq
0                       Café  0.08
1                Coffee Shop  0.06
2                       Park  0.06
3                        Pub  0.05
4  Middle Eastern Restaurant  0.04


----Brockley----
         venue  freq
0  Coffee Shop  0.13
1          Pub  0.12
2         Café  0.07
3         Park  0.06
4    Gastropub  0.04


----Camberwell----
                       venue  freq
0                       Café  0.08
```

```
1               Coffee Shop  0.06
2                      Park  0.06
3                       Pub  0.05
4  Middle Eastern Restaurant  0.04


----Catford----
           venue  freq
0  Grocery Store  0.13
1           Park  0.09
2    Supermarket  0.09
3           Café  0.06
4            Pub  0.06


----Chinbrook----
                venue  freq
0                 Pub  0.13
1       Grocery Store  0.13
2                Park  0.09
3                Café  0.07
4  Italian Restaurant  0.06


----Crofton Park----
         venue  freq
0  Coffee Shop  0.13
1          Pub  0.12
2         Café  0.07
3         Park  0.06
4    Gastropub  0.04


----Denmark Hill----
                       venue  freq
0                       Café  0.08
1               Coffee Shop  0.06
2                      Park  0.06
3                       Pub  0.05
4  Middle Eastern Restaurant  0.04


----Deptford----
         venue  freq
0          Pub  0.13
1  Coffee Shop  0.06
2         Café  0.06
3          Bar  0.05
4         Park  0.03


----Dulwich----
         venue  freq
0          Pub  0.14
1         Café  0.08
2       Bakery  0.06
3  Coffee Shop  0.06
```

```
4         Park  0.06


----East Dulwich----
                venue  freq
0                 Pub  0.12
1               Café  0.09
2         Pizza Place  0.08
3         Coffee Shop  0.05
4  Italian Restaurant  0.05


----Elephant and Castle----
                venue  freq
0                 Pub  0.08
1               Café  0.08
2         Coffee Shop  0.07
3               Hotel  0.05
4  Italian Restaurant  0.03


----Forest Hill----
         venue  freq
0          Pub  0.15
1  Coffee Shop  0.07
2         Café  0.06
3         Park  0.05
4  Supermarket  0.05


----Gipsy Hill----
           venue  freq
0            Pub  0.12
1    Coffee Shop  0.08
2  Grocery Store  0.06
3           Café  0.05
4           Park  0.05


----Grove Park----
                venue  freq
0                 Pub  0.13
1       Grocery Store  0.13
2                Park  0.09
3               Café  0.07
4  Italian Restaurant  0.06


----Herne Hill----
         venue  freq
0  Coffee Shop  0.10
1         Café  0.06
2          Pub  0.06
3       Market  0.04
4  Pizza Place  0.04
```

```
----Hither Green----
       venue  freq
0        Pub  0.13
1       Café  0.08
2  Gastropub  0.06
3       Park  0.05
4     Garden  0.04


----Honor Oak----
         venue  freq
0          Pub  0.15
1  Coffee Shop  0.07
2         Café  0.06
3         Park  0.05
4  Supermarket  0.05


----Ladywell----
         venue  freq
0          Pub  0.12
1  Coffee Shop  0.08
2         Café  0.08
3         Park  0.06
4    Gastropub  0.05


----Lambeth----
                venue  freq
0         Coffee Shop  0.08
1                 Pub  0.06
2               Hotel  0.06
3  Italian Restaurant  0.05
4             Theater  0.04


----Lee----
                venue  freq
0                 Pub  0.13
1       Grocery Store  0.13
2                Park  0.09
3                Café  0.07
4  Italian Restaurant  0.06


----Lewisham----
       venue  freq
0        Pub  0.13
1       Café  0.08
2  Gastropub  0.06
3       Park  0.05
4     Garden  0.04


----New Cross----
                venue  freq
0                 Pub  0.12
```

```
1         Coffee Shop  0.08
2                Café  0.07
3  Italian Restaurant  0.04
4                 Bar  0.04


----Newington----
         venue  freq
0  Coffee Shop  0.08
1          Pub  0.08
2         Café  0.05
3      Theater  0.04
4        Hotel  0.04


----Nunhead----
         venue  freq
0          Pub  0.10
1  Pizza Place  0.06
2         Café  0.06
3         Park  0.05
4  Coffee Shop  0.05


----Oval----
         venue  freq
0         Café  0.13
1          Pub  0.08
2        Hotel  0.07
3  Coffee Shop  0.05
4         Park  0.05


----Peckham----
         venue  freq
0          Pub  0.10
1  Pizza Place  0.06
2         Café  0.06
3         Park  0.05
4  Coffee Shop  0.05


----Rotherhithe----
         venue  freq
0          Pub  0.10
1      Brewery  0.08
2  Coffee Shop  0.06
3         Park  0.06
4          Bar  0.05


----Selhurst----
           venue  freq
0            Pub  0.12
1  Grocery Store  0.10
2           Café  0.08
3    Coffee Shop  0.06
```

```
4    Supermarket  0.06


----South Norwood----
           venue  freq
0            Pub  0.12
1  Grocery Store  0.10
2           Café  0.08
3    Coffee Shop  0.06
4    Supermarket  0.06


----Southend----
           venue  freq
0  Grocery Store  0.13
1           Park  0.09
2    Supermarket  0.09
3           Café  0.06
4            Pub  0.06


----St Johns----
         venue  freq
0  Coffee Shop  0.13
1          Pub  0.12
2         Café  0.07
3         Park  0.06
4    Gastropub  0.04


----Surrey Quays----
         venue  freq
0          Pub  0.10
1      Brewery  0.08
2  Coffee Shop  0.06
3         Park  0.06
4          Bar  0.05


----Tulse Hill----
         venue  freq
0  Coffee Shop  0.10
1          Pub  0.10
2         Café  0.06
3      Brewery  0.04
4  Pizza Place  0.04


----Upper Norwood----
                venue  freq
0                 Pub  0.12
1         Coffee Shop  0.06
2                Park  0.06
3  Italian Restaurant  0.05
4       Grocery Store  0.04
```

```
----Walworth----
                venue  freq
0                 Pub  0.11
1         Coffee Shop  0.09
2                Café  0.09
3  Italian Restaurant  0.04
4         Pizza Place  0.03


----West Norwood----
           venue  freq
0            Pub  0.13
1  Grocery Store  0.09
2    Coffee Shop  0.09
3           Café  0.06
4         Bakery  0.04
```

Creating new dataframe:

In [113]:
```
def return_most_common_venues(row, num_top_venues):
    row_categories = row.iloc[1:]
    row_categories_sorted = row_categories.sort_values(ascending = False)

    return row_categories_sorted.index.values[0:num_top_venues]
```

In [114]:
```
num_top_venues = 10

indicators = ['st', 'nd', 'rd']

# create columns according to number of top venues
columns = ['Neighbourhood']
for ind in np.arange(num_top_venues):
    try:
        columns.append('{}{} Most Common Venue'.format(ind+1, indicators[ind
]))
    except:
        columns.append('{}th Most Common Venue'.format(ind+1))

# create a new dataframe
neighbourhoods_venues_sorted = pd.DataFrame(columns=columns)
neighbourhoods_venues_sorted['Neighbourhood'] = se_grouped['Neighbourhood']

for ind in np.arange(se_grouped.shape[0]):
    neighbourhoods_venues_sorted.iloc[ind, 1:] = return_most_common_venues(se_
grouped.iloc[ind, :], num_top_venues)
```

In [115]:
```
neighbourhoods_venues_sorted.to_csv('neighbourhoods_venues_sorted.csv', index
= False)
```

In [116]:
```
se_grouped_clustering = se_grouped.drop('Neighbourhood', 1)
```

Clustering of Neighbourhoods

In [117]:
```
# set number of clusters
kclusters = 5

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

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

Out[117]: array([2, 1, 2, 3, 0, 0, 0, 1, 1, 0], dtype=int32)

In [118]:
```
kmeans.labels_[0:10]
```

Out[118]: array([2, 1, 2, 3, 0, 0, 0, 1, 1, 0], dtype=int32)

In [119]:
```
# add clustering labels
neighbourhoods_venues_sorted.insert(0, 'Cluster Labels', kmeans.labels_)
```

In [120]:
```
se_merged = se_df
```

In [121]:
```
# match/merge SE London data with latitude/longitude for each neighborhood
se_merged_latlong = se_merged.join(neighbourhoods_venues_sorted.set_index('Nei
ghbourhood'), on = 'Location')
```

In [ ]:
```
se_merged_latlong.head(3)
```

In [123]:
```
se_clusters = se_merged_latlong
```

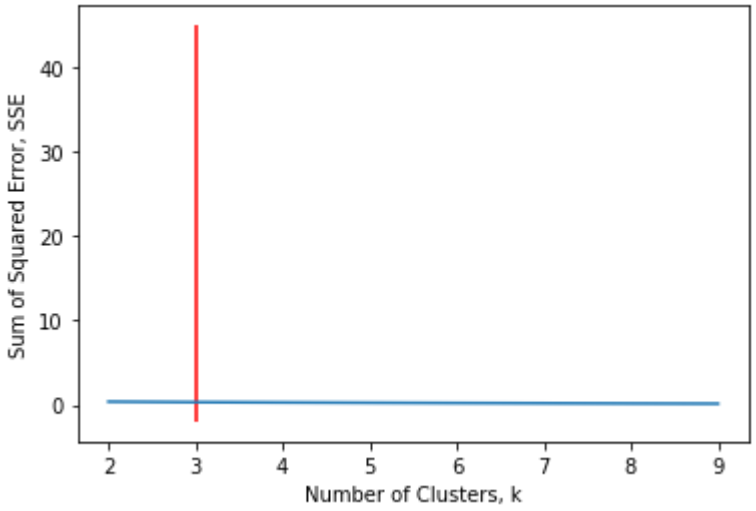
1. Elbow Method

In [125]:
```
%matplotlib inline
import matplotlib
import numpy as np
```

In [126]:
```
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# SSE is initialize with empty values
# n_clusters is the "k"
sse = {}
for n_cluster1 in range(2, 10):
    kmeans1 = KMeans(n_clusters = n_cluster1, max_iter = 500).fit(se_grouped_c
lustering)
    se_grouped_clustering["clusters"] = kmeans1.labels_

    # The inertia is the sum of distances of samples to their closest cluster
 centre
    sse[n_cluster1] = kmeans1.inertia_
plt.figure()
plt.plot(list(sse.keys()), list(sse.values()))
plt.xlabel("Number of Clusters, k")
plt.ylabel("Sum of Squared Error, SSE")
# vertical line
plt.vlines(3, ymin = -2, ymax = 45, colors = 'red')
plt.show()
```

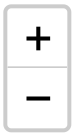


In [131]:
```
# create map
map_clusters = folium.Map(location=[latitude, longitude], zoom_start=11)

# set color scheme for the clusters
x = np.arange(kclusters)
ys = [i + x + (i*x)**2 for i in range(kclusters)]
colors_array = cm.rainbow(np.linspace(0, 1, len(ys)))
rainbow = [colors.rgb2hex(i) for i in colors_array]

# add markers to the map
markers_colors = []
for lat, lon, poi, cluster in zip(se_clusters['Latitude'], se_clusters['Longit
ude'], se_clusters['Location'], se_clusters['Cluster Labels']):
    label = folium.Popup(str(poi) + ' Cluster ' + str(cluster), parse_html=Tru
e)
    folium.CircleMarker(
        [lat, lon],
        radius=20,
        popup=label,
        color=rainbow[cluster-1],
        fill=True,
        fill_color=rainbow[cluster-1],
        fill_opacity=0.7).add_to(map_clusters)

display(map_clusters)
```



Cluster 1

In [129]:
```
se_clusters.loc[se_clusters['Cluster Labels'] == 1, se_clusters.columns[[1] + list(range(5, se_clusters.shape[1]))]]
```

Out[129]:

	Borough	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th I Com Ve
12	Lewisham	1	Grocery Store	Pub	Park	Café	Gym / Fitness Center	Italian Restaurant	Superma
19	Lewisham	1	Grocery Store	Pub	Park	Café	Gym / Fitness Center	Italian Restaurant	Superma
26	Lewisham	1	Grocery Store	Park	Supermarket	Pub	Café	Coffee Shop	Fast I Restau
32	Lewisham	1	Grocery Store	Park	Supermarket	Pub	Café	Coffee Shop	Fast I Restau
44	Lewisham	1	Grocery Store	Park	Supermarket	Pub	Café	Coffee Shop	Fast I Restau
45	Lewisham	1	Grocery Store	Pub	Park	Café	Gym / Fitness Center	Italian Restaurant	Superma

Cluster 2

In [130]:
```
se_clusters.loc[se_clusters['Cluster Labels'] == 2, se_clusters.columns[[1] + list(range(5, se_clusters.shape[1]))]]
```

Out[130]:

	Borough	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Mos Commo Venu
5	Southwark	2	Pub	Café	Coffee Shop	Hotel	Italian Restaurant	Theater	Par
6	Southwark	2	Pub	Café	Coffee Shop	Hotel	Italian Restaurant	Theater	Par
7	Southwark	2	Pub	Café	Coffee Shop	Hotel	Italian Restaurant	Theater	Par
8	Southwark	2	Coffee Shop	Hotel	Pub	Italian Restaurant	Theater	Seafood Restaurant	A Museur
18	Lambeth	2	Coffee Shop	Hotel	Pub	Italian Restaurant	Theater	Seafood Restaurant	A Museur
22	Southwark	2	Coffee Shop	Pub	Café	Italian Restaurant	Hotel	Theater	Pizz Plac
23	Southwark	2	Coffee Shop	Pub	Café	Italian Restaurant	Hotel	Theater	Pizz Plac
30	Southwark	2	Coffee Shop	Hotel	Pub	Italian Restaurant	Theater	Seafood Restaurant	A Museur

4. Result

The following are the highlights of the 5 clusters above:

- Pubs, Cafe, Coffee Shops are popular in the South East London.
- As for restaurants, the Italian Restaurants are very popular in the South East London area. Especially in Southwark and Lambeth areas.
- With the Lewisham area being the most condensed area of Africans in the South East Area, it is surprising to see how in the top 10 venues, you can barely see restaurants in the top 5 venues.
- Although, the Clusters have variations, a very visible presence is the predominance of pubs.

5. Discussion and Conclusion

It is very important to note that Clusters 1 and 2 (shown above) are the most viable clusters to create a brand African Restaurant. Their proximity to other amenities and accessibility to station are paramount. These 2 clusters do not have top restaurants that could rival their standards if they are created. And the proximity to resources needed is paramount as Lewisham and Lambeth are not far out from Peckham (under Southwark).

In conclusion, this project would have had better results if there were more data in terms of crime data within the area, traffic access and allowance of more venues exploration with the Foursquare (limited venues for free calls).

Also, getting the ratings and feedbacks of the current restaurants within the clusters would have helped in providing more insight into the best location.

In [ ]: