Introduction

The question I will address in this project is twofold:

1.) What area in/around London would be ideal for a new business venture? 2.) What type of business is most popular in this area?

The data I will use comes from foursquare API as well as referencing google searches / wikipedia.

Below I will import libraries and functions to use for the project

In [72]:
```
from geopy.geocoders import Nominatim
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
import requests
from sklearn.cluster import KMeans
import matplotlib.cm as cm
import matplotlib.colors as colors


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

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

LDF.head()
```

Out[72]:

	Borough	Inner	Status	Local authority	Political control	Headquarters	Area (sq mi)	Population (2013 est)[1]	Co-ordinates	Nr. in map
0	Barking and Dagenham [note 1]	NaN	NaN	Barking and Dagenham London Borough Council	Labour	Town Hall, 1 Town Square	13.93	194352	51°33′39″N 0°09′21″E / 51.5607°N 0.1557°E	25
1	Barnet	NaN	NaN	Barnet London Borough Council	Conservative	Barnet House, 2 Bristol Avenue, Colindale	33.49	369088	51°37′31″N 0°09′06″W / 51.6252°N 0.1517°W	31
2	Bexley	NaN	NaN	Bexley London Borough Council	Conservative	Civic Offices, 2 Watling Street	23.38	236687	51°27′18″N 0°09′02″E / 51.4549°N 0.1505°E	23
3	Brent	NaN	NaN	Brent London Borough Council	Labour	Brent Civic Centre, Engineers Way	16.70	317264	51°33′32″N 0°16′54″W / 51.5588°N 0.2817°W	12
4	Bromley	NaN	NaN	Bromley London Borough Council	Conservative	Civic Centre, Stockwell Close	57.97	317899	51°24′14″N 0°01′11″E / 51.4039°N 0.0198°E	20

Below I will clean the data pulled from the wikipedia

In [73]:
```
LF = LDF.drop(['Status','Local authority','Political control','Headquarters','Nr. in map'], axis=1)
LF['Inner'].replace(np.nan,'0', inplace=True)
LF['Borough'].replace('Barking and Dagenham [note 1]','Barking and Dagenham', inplace=True)
LF['Borough'].replace('Greenwich [note 2]','Greenwich', inplace=True)
LF['Borough'].replace('Hammersmith and Fulham [note 4]','Hammersmith and Fulham', inplace=True)
Inn = ['Camden','Greenwich','Hackney','Hammersmith and Fulham','Islington','Kensington and Chelsea','Lewisham','Lambeth','Southwark','Tower Hamlets','Wandsworth','Westminster']
LF.head()
LF['Inner'] = '0'
LF.head()
```

Out[73]:

	Borough	Inner	Area (sq mi)	Population (2013 est)[1]	Co-ordinates
0	Barking and Dagenham	0	13.93	194352	51°33′39″N 0°09′21″E / 51.5607°N 0.1557°E
1	Barnet	0	33.49	369088	51°37′31″N 0°09′06″W / 51.6252°N 0.1517°W
2	Bexley	0	23.38	236687	51°27′18″N 0°09′02″E / 51.4549°N 0.1505°E
3	Brent	0	16.70	317264	51°33′32″N 0°16′54″W / 51.5588°N 0.2817°W
4	Bromley	0	57.97	317899	51°24′14″N 0°01′11″E / 51.4039°N 0.0198°E

Now we need to get the coords of london and map everything

In [74]:
```
address = 'London'

geolocator = Nominatim(user_agent="London_explorer")
location = geolocator.geocode(address)
London_latitude = location.latitude
London_longitude = location.longitude

print('The geograpical coordinates of London are {}, {}.'.format(London_latitude, London_longitude))
```

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

In [75]:
```
!pip install folium
import folium

Fin_Brgh = folium.Map(location=[London_latitude, London_longitude], zoom_start=12)


for lat, lng, label in zip(fin['Latitude'], fin['Longitude'],
                           fin['Borough']):
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius=9,
        popup=label,
        color='Red',
        fill=True,
        fill_color='#Blue',
        fill_opacity=0.7).add_to(Fin_Brgh)
Fin_Brgh
```

```
Requirement already satisfied: folium in /opt/conda/envs/Python36/lib/python3.6/
site-packages (0.11.0)
Requirement already satisfied: jinja2>=2.9 in /opt/conda/envs/Python36/lib/pytho
n3.6/site-packages (from folium) (2.10)
Requirement already satisfied: requests in /opt/conda/envs/Python36/lib/python3.
6/site-packages (from folium) (2.21.0)
Requirement already satisfied: branca>=0.3.0 in /opt/conda/envs/Python36/lib/pyt
hon3.6/site-packages (from folium) (0.4.1)
Requirement already satisfied: numpy in /opt/conda/envs/Python36/lib/python3.6/s
ite-packages (from folium) (1.15.4)
Requirement already satisfied: MarkupSafe>=0.23 in /opt/conda/envs/Python36/lib/
python3.6/site-packages (from jinja2>=2.9->folium) (1.1.0)
Requirement already satisfied: urllib3<1.25,>=1.21.1 in /opt/conda/envs/Python36
/lib/python3.6/site-packages (from requests->folium) (1.24.1)
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/envs/Python36/li
b/python3.6/site-packages (from requests->folium) (2020.4.5.1)
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in /opt/conda/envs/Python36
/lib/python3.6/site-packages (from requests->folium) (3.0.4)
Requirement already satisfied: idna<2.9,>=2.5 in /opt/conda/envs/Python36/lib/py
thon3.6/site-packages (from requests->folium) (2.8)
```

Out[75]: Make this Notebook Trusted to load map: File -> Trust Notebook

Here is where we start getting the venue data from foursquare

In [76]:
```
CLIENT_ID = 'F250FC4MHPZTI5HSHTEP1BSKAFNCRD42ZSO5RVKUFLLNSLHU' #'your-client-ID' #
your Foursquare ID
CLIENT_SECRET = 'QRQSKBJMOCPSKTRGENBAAQQCGG0RWJGNFYT3NZBJRHL3GZDW' #'your-client-se
cret' # your Foursquare Secret
VERSION = '20200610' # Foursquare API version

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

```
My credentails:
My CLIENT_ID: F250FC4MHPZTI5HSHTEP1BSKAFNCRD42ZSO5RVKUFLLNSLHU
My CLIENT_SECRET:QRQSKBJMOCPSKTRGENBAAQQCGG0RWJGNFYT3NZBJRHL3GZDW
```

In [77]:
```
radius = 5000
LIMIT = 100

def getVenues(names, latitudes, longitudes, radius=5000):

    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={}&client_se
cret={}&v={}&ll={},{}&radius={}&limit={}'.format(
            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 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 ve
nue_list])
    nearby_venues.columns = ['Borough',
                  'Latitude',
                  'Longitude',
                  'Venue',
                  'Venue_Lat',
                  'Venue_Long',
                  'Venue_Category']

    return(nearby_venues)
```

In [78]:
```
Brgh_Venues = getVenues(names=fin['Borough'],
                        latitudes=fin['Latitude'],
                        longitudes=fin['Longitude'])
```

```
Barking and Dagenham
Bexley
Bromley
Enfield
Haringey
Havering
Merton
Redbridge
```

In [79]:
```
Brgh_Venues.groupby('Borough').count()
```

Out[79]:

Borough	Latitude	Longitude	Venue	Venue_Lat	Venue_Long	Venue_Category
Barking and Dagenham	99	99	99	99	99	99
Bexley	88	88	88	88	88	88
Bromley	100	100	100	100	100	100
Enfield	100	100	100	100	100	100
Haringey	100	100	100	100	100	100
Havering	100	100	100	100	100	100
Merton	100	100	100	100	100	100
Redbridge	100	100	100	100	100	100

In [80]:
```
London_Brgh_onehot = pd.get_dummies(Brgh_Venues[['Venue_Category']], prefix="", pre
fix_sep="")
mid =  Brgh_Venues['Borough']

London_Brgh_onehot.insert(0, 'Borough', mid)

London_Brgh_onehot.head()
```

Out[80]:

	Borough	ATM	American Restaurant	Argentinian Restaurant	Art Gallery	Art Museum	Arts & Crafts Store	Asian Restaurant	Athletics & Sports	Bagel Shop	...	Trail
0	Barking and Dagenham	0	0	0	0	0	0	0	0	0	...	0
1	Barking and Dagenham	0	0	0	0	0	0	0	0	0	...	0
2	Barking and Dagenham	0	0	0	0	0	0	0	0	0	...	0
3	Barking and Dagenham	0	0	0	0	0	0	0	0	0	...	0
4	Barking and Dagenham	0	0	0	0	0	0	0	0	0	...	0

5 rows × 170 columns

Now we need to streamline the data so that we see top venues per borough

In [81]:
```
Brgh_grouped = London_Brgh_onehot.groupby('Borough').mean().reset_index()
Brgh_grouped
```

Out[81]:

	Borough	ATM	American Restaurant	Argentinian Restaurant	Art Gallery	Art Museum	Arts & Crafts Store	Asian Restaurant	Athletics & Sports	Bagel Shop	...
0	Barking and Dagenham	0.000000	0.010101	0.00	0.00	0.000000	0.00	0.000000	0.00	0.000000	...
1	Bexley	0.011364	0.022727	0.00	0.00	0.011364	0.00	0.011364	0.00	0.011364	...
2	Bromley	0.000000	0.010000	0.00	0.00	0.000000	0.00	0.010000	0.01	0.000000	...
3	Enfield	0.000000	0.000000	0.00	0.00	0.000000	0.00	0.000000	0.01	0.000000	...
4	Haringey	0.000000	0.000000	0.00	0.00	0.000000	0.00	0.000000	0.00	0.000000	...
5	Havering	0.000000	0.000000	0.00	0.01	0.000000	0.01	0.000000	0.01	0.000000	...
6	Merton	0.000000	0.000000	0.01	0.00	0.000000	0.00	0.020000	0.00	0.000000	...
7	Redbridge	0.000000	0.000000	0.00	0.02	0.000000	0.00	0.000000	0.00	0.000000	...

8 rows × 170 columns

In [83]:
```
num_top_venues = 8

for brgh in Brgh_grouped['Borough']:
    print("_________"+brgh+"________")
    temp = Brgh_grouped[Brgh_grouped['Borough'] == brgh].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')
```

_____Barking and Dagenham_____

	venue	freq
0	Supermarket	0.10
1	Park	0.09
2	Grocery Store	0.08
3	Pub	0.07
4	Coffee Shop	0.06
5	Café	0.04
6	Italian Restaurant	0.04
7	Pizza Place	0.03

_____Bexley_____

	venue	freq
0	Coffee Shop	0.07
1	Chinese Restaurant	0.05
2	Pizza Place	0.05
3	Brewery	0.03
4	Ice Cream Shop	0.03
5	Bakery	0.03
6	Discount Store	0.03
7	Sandwich Place	0.02

_____Bromley_____

	venue	freq
0	Coffee Shop	0.10
1	Pub	0.08
2	Park	0.07
3	Grocery Store	0.06
4	Gym / Fitness Center	0.05
5	Pizza Place	0.05
6	Café	0.04
7	Supermarket	0.04

_____Enfield_____

	venue	freq
0	Coffee Shop	0.11
1	Turkish Restaurant	0.07
2	Pub	0.07
3	Café	0.05
4	Gym / Fitness Center	0.05
5	Park	0.05
6	Supermarket	0.05
7	Greek Restaurant	0.04

_____Haringey_____

	venue	freq
0	Pub	0.12
1	Park	0.09
2	Café	0.07
3	Turkish Restaurant	0.07
4	Coffee Shop	0.06
5	Pizza Place	0.04
6	Trail	0.04
7	Japanese Restaurant	0.03

_____Havering_____

	venue	freq
0	Hotel	0.07
1	Park	0.05

Looking at the top venues for the boroughs, we can look at business ideas. For example, in most boroughs, pubs, coffee shops and parks are popular...therefore, it might be a good idea to open one of these in one of the boroughs where these numbers are lacking. As an example...perhaps a good pub for tourists is what the Havering Borough is missing seeing how there is a frequented hotel there.

In [84]:
```
map_clusters = folium.Map(location=[London_latitude, London_longitude], zoom_start=
10)

# 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, rent, pop in zip(Borough_merged['Latitude'],
                                    Borough_merged['Longitude'],
                                    Borough_merged['Borough'],
                                    Borough_merged['Cluster Label'],
                                    Borough_merged['Max_Rent'],
                                    Borough_merged['Population']):
    label = folium.Popup(str(poi) + ' Cluster ' + str(cluster) + " " + "Rent " + st
r(rent) + " " + "Population " + str(pop), parse_html=True)
    folium.CircleMarker(
        [lat, lon],
        radius=25,
        popup=label,
        color=rainbow[cluster-1],
        fill=True,
        fill_color=rainbow[cluster-1],
        fill_opacity=0.7).add_to(map_clusters)

map_clusters
```

Out[84]: Make this Notebook Trusted to load map: File -> Trust Notebook

Results

As mentioned above, this data gives insight into what boroughs in the area of study might be in need of certain businesses. At the very least, it gives the prospective business owners areas to focus in on and explore further.

Conclusion

To conclude, I think that this cycle could continue...drilling down even further. For example, once the prospective business owner selects a borough, you could do this process again using data only from that borough. This could enable the owner to see what specific areas see the most foot traffic. You could also fuse this with data from other sources to paint an even clearer picture. For example, getting movie theater attendance data to determine what days an owner could offer incentives or specials for their business (Happy Hour etc).

In [ ]: