Capstone Project: The Battle of the Neighbourhoods (Week 2)

Corurse 9: Applied Data Science Capstone by IBM/Coursera

Table of contents

Introduction: Business Problem

In this project, we will try to build a model to find the optimal neighbourhood for openning a new business. As an example we will specify the business type to be an **Italian restaurant**.

Since London is huge city with a lot of restraunts, we will try to identify the optimal neighbourhood to open a new Italian restaurant. We will try to identify the optimal neighborhood based on:

- Less number of **high rating** Italian restraunts.
- Less number of **Italian restraunts**.
- Less number of restraunts.

Data

Based on teh description of the Business Problem, we will need to get data of:

- List of neighbourhoods in London with their latitudes, longitudes and area.
- Number of restraunts (any type) in each neighbourhood.
- Number of Italian restraunts in the neighbourhood.
- Rating of each Italian restraunt in the neighbourhood.

We will extract the required data as follows:

- We will get the list of neighbourhoods in London and their geo-location from Wikipedia page.
- We will get the restraunts information for each neighbourhood from **Foursquare API**.
- Also, for the map visualization we will get the location of London using **geopy API**.

Assumptions

First, lets import the required libraries

In [1]:
```
import numpy as np # library to handle data in a vectorized manner

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

import re #Library to read config file

import geocoder

import folium
from folium import plugins
from folium.plugins import HeatMap

#!conda install -c conda-forge geopy --yes
from geopy.geocoders import Nominatim

import geopy.distance

import configparser #Library to read config file

import json

import requests

import math

from sklearn.preprocessing import StandardScaler

from sklearn.cluster import KMeans
```

```
Intel(R) Data Analytics Acceleration Library (Intel(R) DAAL) solvers for sklearn enabled: https://intelpytho
n.github.io/daal4py/sklearn.html
```

Neighbourhood Data

We will retrieve the list of London neighbourhoods from Wikipedia page List of London boroughs which contain London neighbourhoods and their Geo-Location information and other irrelavant information which will be excluded.

In [2]:
```
# Funtion to calculate area in meter square from mile square
def convert_mi2_m2(row) :
    mi_2 = row["Area (sq mi)"]
    m_2 = mi_2 * 2590000
    return m_2

# function to calculate Circle radius in meteres from meter square area
def convert_area_radius(row) :
    area = row["Area"]
    radius = math.sqrt((area / math.pi))
    return radius

# read html tables from the Wikipedia page
dfs = pd.read_html('https://en.wikipedia.org/wiki/List_of_London_boroughs')
print('List of London boroughs loaded')

# we get 2 dataframes so we concat them into one dataframe
df = pd.concat([dfs[0], dfs[1]])

# Caclculate Area and radius in metric measurements
df["Area"] = df.apply(lambda row : convert_mi2_m2(row), axis=1)
df["Area-Radius"] = df.apply(lambda row : convert_area_radius(row), axis=1)

# extract only the columns we are interested in
london_df = df[['Borough', 'Co-ordinates', 'Area', 'Area-Radius']].reset_index(drop=True)
london_df.rename(columns={'Borough' : 'Neighbourhood'}, inplace = True)

# clean Borough from [note 1]
note_regex = re.compile(r"\[note \d]")
london_df['Neighbourhood'] = london_df['Neighbourhood'].str.replace(pat=note_regex, repl='', regex=True)

#Split the Co-ordinates column into 2 columns for DMS and Decimal geo-formats
london_df[['Co-ordinates_DMS', 'Co-ordinates_DEC']] = london_df['Co-ordinates'].str.split(pat=' / ', expand=T
rue)
london_df[['Latitude', 'Longitude']] = london_df['Co-ordinates_DEC'].str.split(expand=True)

# extract and clean Latitude and Longitude from Co-ordinates_DEC
def clean_dec(dec_str):

    sign = -1 if re.search('[swSW]', dec_str) else 1
    dec_str = re.sub(r'°.', '', dec_str)
    dec_str = re.sub(' ', '', dec_str)
```

```
    dec_str = re.sub(u'\ufeff', '', dec_str)

    return sign * (float(dec_str))

london_df['Latitude'] = london_df['Latitude'].apply(clean_dec)
london_df['Longitude'] = london_df['Longitude'].apply(clean_dec)

# extract only needed columns
london_df = london_df[['Neighbourhood', 'Latitude', 'Longitude', 'Area', 'Area-Radius']]

london_df.head()
```

```
List of London boroughs loaded
```

Out[2]:

	Neighbourhood	Latitude	Longitude	Area	Area-Radius
0	Barking and Dagenham	51.5607	0.1557	36078700.0	3388.835625
1	Barnet	51.6252	-0.1517	86739100.0	5254.513588
2	Bexley	51.4549	0.1505	60554200.0	4390.330342
3	Brent	51.5588	-0.2817	43253000.0	3710.506368
4	Bromley	51.4039	0.0198	150142300.0	6913.159800

Each neighborhood in London has different area. We already consider/assume the Longitude and Latitude we got from Wikipedia to be the center of each neighborhood.

As we are going to provide the **radius** in our Foursquare APIs calls, we associated each neighborhood to a radius value of a circle whose area equals the neighborhood area.

London City Location

Now, le's get the location of London using **geopy API**.

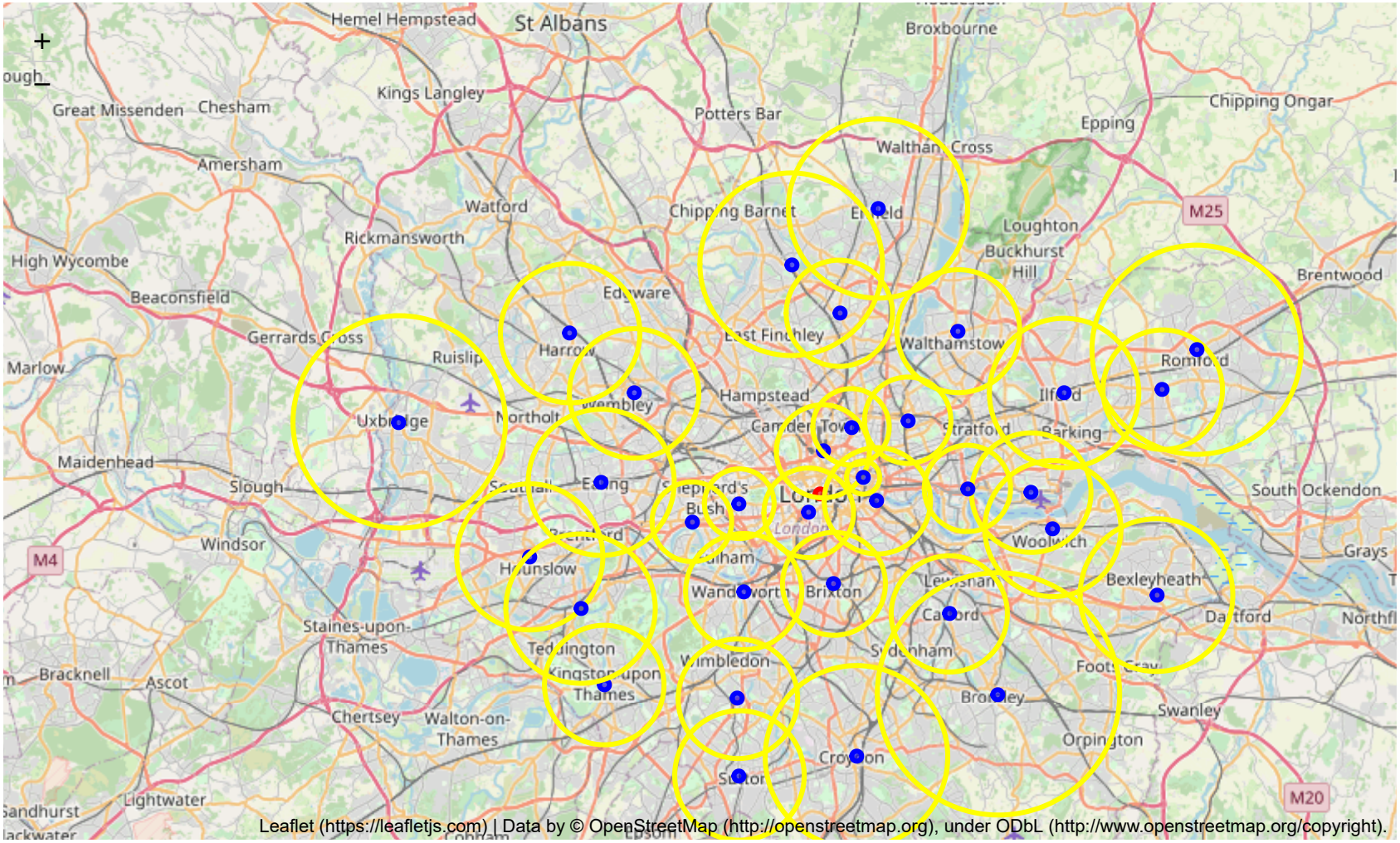
In [3]:
```
def get_geo_location(addr) :
    geolocator = Nominatim(user_agent="uk_explorer")
    location = geolocator.geocode(addr)
    latitude = location.latitude
    longitude = location.longitude
    #print('The geograpical coordinate of London City are {}, {}.'.format(latitude, longitude))
    return [latitude, longitude]

address = 'London, UK'
london_location = get_geo_location(address)
print('Coordinate of {}: {}'.format(address, london_location))
```

```
Coordinate of London, UK: [51.5073219, -0.1276474]
```

Now we can visualize the London maps and its neighborhood. Each neighborhood center will be surrounded with circls of radius proportional to its area.

In [4]:
```
map_london = folium.Map(location=london_location, zoom_start=10)


folium.CircleMarker(
    london_location,
    radius=3,
    color='red',
    popup=address,
    fill=True,
    fill_color='red',
    fill_opacity=0.6
).add_to(map_london)

for lat, lng, radius, label in zip(london_df.Latitude, london_df.Longitude, london_df['Area-Radius'] , london
_df.Neighbourhood) :
    folium.CircleMarker(
        [lat, lng],
        radius=3,
        color='blue',
        popup=label,
        fill = True,
        fill_color='blue',
        fill_opacity=0.6
    ).add_to(map_london)

    folium.Circle(
        [lat, lng],
        radius=radius,
        color='Yellow'
    ).add_to(map_london)

map_london
```

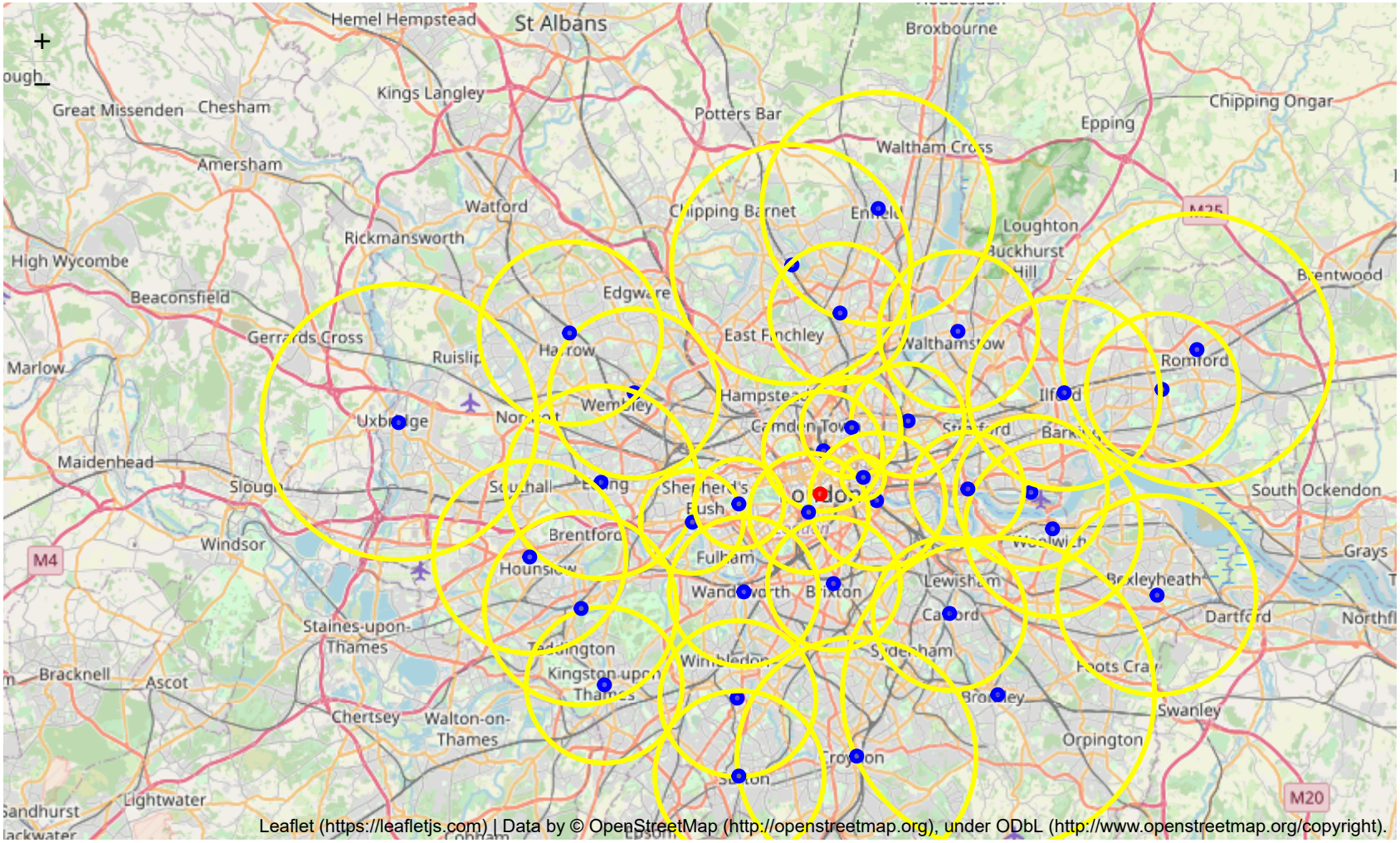
Out[4]:



The Circle coverage of London doesn't seem to be good enough. Let's increase the radius of all neighborhoods with 30% and check again

In [5]:
```
london_df['Area-Radius'] *= 1.3

map_london = folium.Map(location=london_location, zoom_start=10)


folium.CircleMarker(
    london_location,
    radius=3,
    color='red',
    popup=address,
    fill=True,
    fill_color='red',
    fill_opacity=0.6
).add_to(map_london)

for lat, lng, radius, label in zip(london_df.Latitude, london_df.Longitude, london_df['Area-Radius'] , london
_df.Neighbourhood) :
    folium.CircleMarker(
        [lat, lng],
        radius=3,
        color='blue',
        popup=label,
        fill = True,
        fill_color='blue',
        fill_opacity=0.6
    ).add_to(map_london)

    folium.Circle(
        [lat, lng],
        radius=radius,
        color='Yellow'
    ).add_to(map_london)

map_london
```

Out[5]:



Now we have better coverage to London. We can use this dataset for our analysis.

Restraunts Data

Now we need to build a data set of restaurants in each neighborhood. Our focus will be in restaurants in general and specially italian restaurants. For this purpose we will use **Foursquare APIs**.

Foursquare

First, let's prepare the basic **Foursquare** connection configurations.

In [6]:
```
# Read Foursquare authentication info from hidden config file
config = configparser.ConfigParser()
config.read('secrets.cfg')
CLIENT_ID = config['4square_personal']['CLIENT_ID']
CLIENT_SECRET = config['4square_personal']['CLIENT_SECRET']
VERSION = '20201103' # Foursquare API version
LIMIT = 100 # A default Foursquare API limit value

REQUEST_DEFAULT_PARAMS = dict(
    client_id = CLIENT_ID,
    client_secret = CLIENT_SECRET,
    v=VERSION
)
```

We will use the **explore** endpoint to query for restaurants in each neighborhood.
To limit the query result, we will specify parameter **section** as **food**. This should limit the result to food related venues. Also, we will utilize the **sortByPopularity** parameter to get results sorted based on Popularity.

Also, we will use the Venues Details endpoint to get the rating of the Italian restaurants.

In [7]:
```
def get_Restaurant_Rating(restaurant_id) :
    URL_4SQU_DETAIL = 'https://api.foursquare.com/v2/venues/{}'.format(restaurant_id)
    detail_params = REQUEST_DEFAULT_PARAMS.copy()
    try :
        rating = requests.get(url=URL_4SQU_DETAIL, params=detail_params).json()['response']['venue']['rating'
]
    except :
        rating = 0.0

    return rating

# validate whether the venue category is a competitor restaurant and if it is Italian restaurant
def validate_rest_category(rest_cat_name) :
    excluded_categories = ['Café', 'Pub', 'Food Court', 'Fast Food', 'Shop', 'Coffee', 'Bakery', 'Breakfast']
    italian_list = ['Italian Restaurant', 'Abruzzo Restaurant', 'Agriturismo', 'Aosta Restaurant', 'Basilicat
a Restaurant', 'Calabria Restaurant', 'Campanian Restaurant', 'Emilia Restaurant', 'Friuli Restaurant', 'Ligu
rian Restaurant', 'Lombard Restaurant', 'Malga', 'Marche Restaurant', 'Piadineria', 'Piedmontese Restaurant',
'Puglia Restaurant', 'Romagna Restaurant', 'Roman Restaurant', 'Sardinian Restaurant', 'Sicilian Restaurant',
'South Tyrolean Restaurant', 'Trattoria/Osteria', 'Trentino Restaurant', 'Tuscan Restaurant', 'Umbrian Restau
rant', 'Veneto Restaurant']
    is_restaurant = True
    for excl in excluded_categories :
        if excl in rest_cat_name :
            is_restaurant = False
            break
    is_italian = rest_cat_name in italian_list

    return is_restaurant, is_italian


def get_neighborhood_restaurants(lat, lng, radius) :
#def get_neighborhood_restaurants(venue) :
    URL_4SQU_EXPLORE = 'https://api.foursquare.com/v2/venues/explore'

    explore_params = REQUEST_DEFAULT_PARAMS.copy()

    explore_params.update({
        "ll" : '{}, {}'.format(lat, lng),
        "radius" : radius,
        "section" : 'food',
        #"query" : 'restaurant',
        "sortByPopularity" : 1
```

```
        })
    restaurants_list = requests.get(url=URL_4SQU_EXPLORE, params=explore_params).json()['response']['groups']
[0]['items']
    return restaurants_list

#london_df[['Neighbourhood' == 'Barnet']]
restaurants_df = pd.DataFrame(columns=['Neighbourhood', 'Restaurant_ID', 'Restaurant_Name', 'Category_ID', 'C
ategory_Name', 'Restaurant_Latitude', 'Restaurant_Longitude', 'Is_Italian', 'Rating'])
for lat, lng, radius, neighbourhood in zip(london_df.Latitude, london_df.Longitude, london_df['Area-Radius']
, london_df.Neighbourhood) :
    rest_lst = get_neighborhood_restaurants(lat, lng, radius)
    for rest_info in rest_lst :
        restaurants_id = rest_info['venue']['id']
        restaurants_name = rest_info['venue']['name']
        rest_cat_id = rest_info['venue']['categories'][0]['id']
        rest_cat_name = rest_info['venue']['categories'][0]['name']
        rest_lat = rest_info['venue']['location']['lat']
        rest_lng = rest_info['venue']['location']['lng']
        is_restaurant, is_italian = validate_rest_category(rest_cat_name)
        rating = 0.0
        if is_restaurant :
            if is_italian :
                rating = get_Restaurant_Rating(restaurants_id)
            restaurants_df = restaurants_df.append({'Neighbourhood' : neighbourhood, 'Restaurant_ID' : restau
rants_id, 'Restaurant_Name' : restaurants_name, 'Category_ID' : rest_cat_id, 'Category_Name' : rest_cat_name,
'Restaurant_Latitude' : rest_lat, 'Restaurant_Longitude' : rest_lng, 'Is_Italian' : is_italian, 'Rating' : ra
ting}, ignore_index=True)


restaurants_df.head()
```

Out[7]:

	Neighbourhood	Restaurant_ID	Restaurant_Name	Category_ID	Category_Name	Restaurant_Latitude	Resta
0	Barking and Dagenham	4bb9e6183db7b713faa0229a	Moby Dick	4bf58dd8d48988d1c4941735	Restaurant	51.580585	
1	Barking and Dagenham	4eb064fb775bbddde7b4f546	Cosmo	4bf58dd8d48988d1c4941735	Restaurant	51.575445	
2	Barking and Dagenham	50baf653183fee49a8b0b4cc	Lara Grill	4f04af1f2fb6e1c99f3db0bb	Turkish Restaurant	51.562445	
3	Barking and Dagenham	4e7a1f5414954a343fb58258	The Pipe Major	4bf58dd8d48988d1c4941735	Restaurant	51.545795	
4	Barking and Dagenham	4c3218fa3896e21e07f2e790	The Compasses (Harvester)	52e81612bcbc57f1066b7a05	English Restaurant	51.557806	

In [8]:
```
london_df = london_df.merge(restaurants_df, on='Neighbourhood', how='inner')

london_df.head()
```

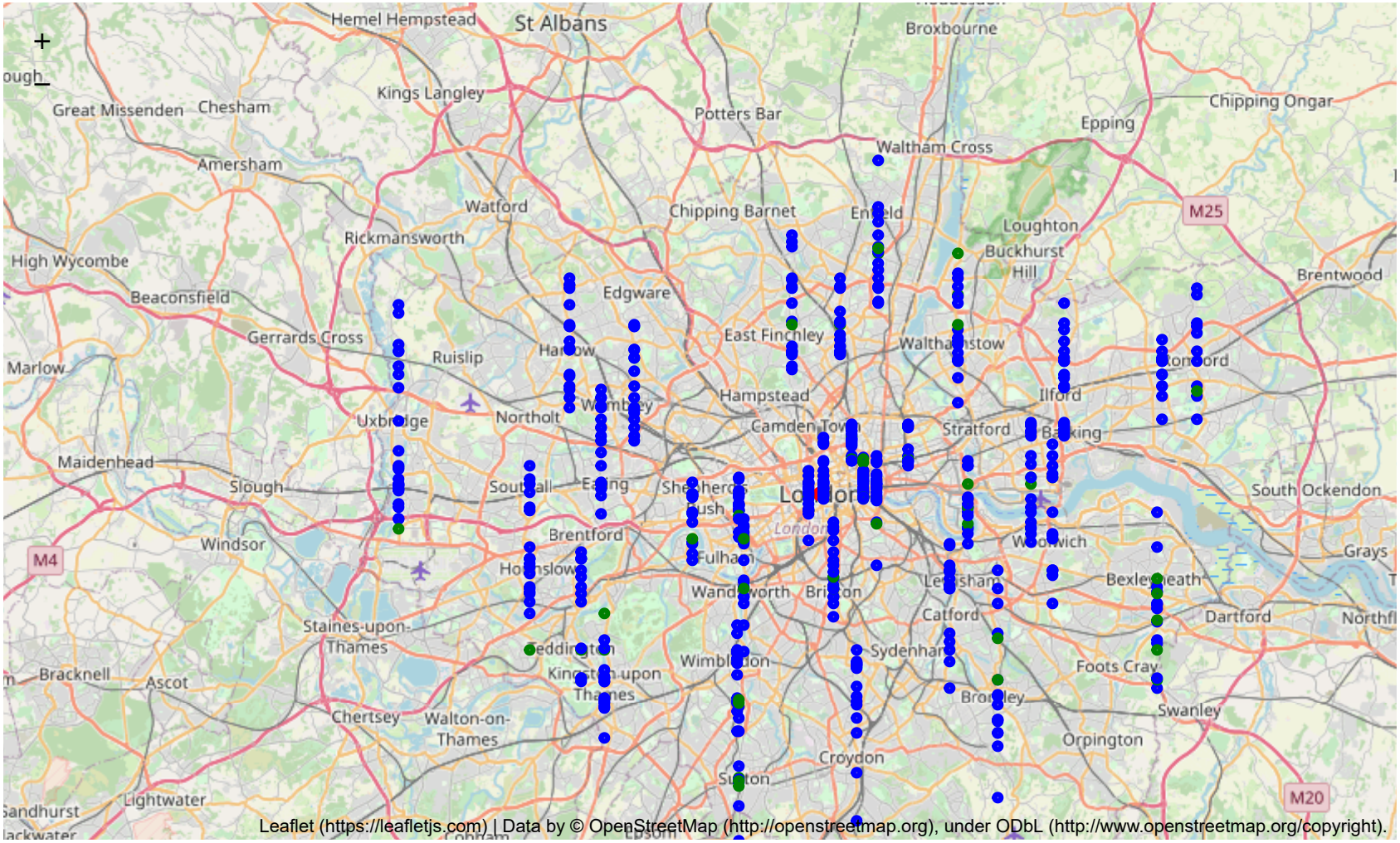
Out[8]:

	Neighbourhood	Latitude	Longitude	Area	Area-Radius	Restaurant_ID	Restaurant_Name	Category_
0	Barking and Dagenham	51.5607	0.1557	36078700.0	4405.486312	4bb9e6183db7b713faa0229a	Moby Dick	4bf58dd8d48988d1c49417
1	Barking and Dagenham	51.5607	0.1557	36078700.0	4405.486312	4eb064fb775bbddde7b4f546	Cosmo	4bf58dd8d48988d1c49417
2	Barking and Dagenham	51.5607	0.1557	36078700.0	4405.486312	50baf653183fee49a8b0b4cc	Lara Grill	4f04af1f2fb6e1c99f3db0l
3	Barking and Dagenham	51.5607	0.1557	36078700.0	4405.486312	4e7a1f5414954a343fb58258	The Pipe Major	4bf58dd8d48988d1c49417
4	Barking and Dagenham	51.5607	0.1557	36078700.0	4405.486312	4c3218fa3896e21e07f2e790	The Compasses (Harvester)	52e81612bcbc57f1066b7a(

Now let's visualize the results on the map of London.

In [9]:
```
map_london = folium.Map(location=london_location, zoom_start=10)

folium.CircleMarker(
    london_location,
    radius=3,
    color='red',
    popup=address,
    fill=True,
    fill_color='red',
    fill_opacity=0.6
).add_to(map_london)

for lat, lng, radius, label, special in zip(london_df.Restaurant_Latitude, london_df.Longitude, london_df.Res
taurant_Longitude , london_df.Restaurant_Name, london_df.Is_Italian) :
    color = 'blue'
    if special :
        color = 'green'

    folium.CircleMarker(
        [lat, lng],
        radius=2,
        color=color,
        popup=label,
        fill = True,
        fill_color= color,
        fill_opacity=0.6
    ).add_to(map_london)

map_london
```

Out[9]:



This conclude our Data Section.

Now we have the list of restaurants and their locations. Also, we know which restaurant is Italian or not.

Methodology

In this project we will try to detect neighborhoods in London with low number of restaurants specially with low number of Italian restaurants. Also, since London contains many restaurants, we will consider the rating of the Italian restaurants so we pick neighborhoods with low average rating of Italian restaurants.

In the first secion of the report, we gathered the data that will be used in our report:

- List of neighborhoods of London.
- The restaurants in each neighborhood.
- The Italian restaurants in each neighborhood.
- The rating of each Italian restaurant.

In the second section of the report, we will perform some analysis on the data we gathered so we could have better understanding of the distribution of restaurants in London and -if needed- we will enrich the data with any extra information to provide better decision.

In the third section, we will use the ***K-Means*** clustering to split London neighborhoods into similar clusters based on the criteria defined above and accordingly, we can decied which neighborhood(s) best to openning a new Italian restaurant.

Analysis

Let's perform some basic analysis on the data.

In [10]:
```
london_boroughs_url = 'london_boroughs.json'
with open(london_boroughs_url) as fp :
    london_boroughs_json = json.load(fp)

def boroughs_style(feature):
    return { 'color': 'blue', 'fill': False }
```

All Restaurants Count

Let's check the total count of restaurants per neighborhood.

In [11]:
```
london_df['count'] = 1

london_df[['Neighbourhood', 'count']].groupby(['Neighbourhood']).sum().reset_index().sort_values(by=['count'
], ascending=False)
```

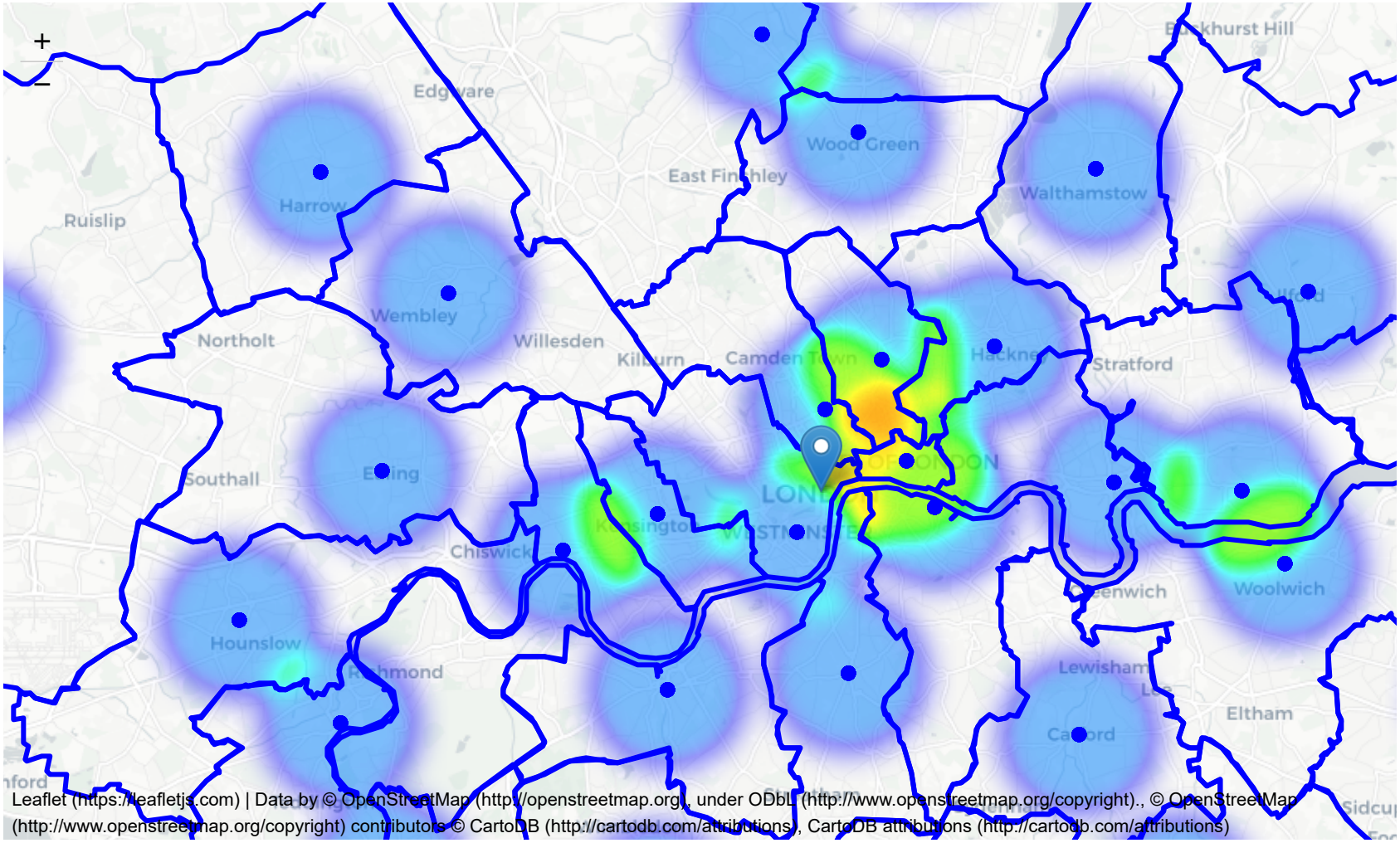
Out[11]:

	Neighbourhood	count
6	City of London	27
16	Hillingdon	26
5	Camden	24
27	Southwark	23
19	Kensington and Chelsea	23
17	Hounslow	21
32	Westminster	21
29	Tower Hamlets	20
14	Harrow	20
21	Lambeth	20
24	Newham	19
31	Wandsworth	19
2	Bexley	19
18	Islington	19
8	Ealing	19
30	Waltham Forest	18
25	Redbridge	18
7	Croydon	17
9	Enfield	17
12	Hammersmith and Fulham	17
1	Barnet	17
4	Bromley	17
3	Brent	17
20	Kingston upon Thames	16
23	Merton	16
13	Haringey	16

	Neighbourhood	count
22	Lewisham	15
10	Greenwich	15
26	Richmond upon Thames	14
28	Sutton	14
0	Barking and Dagenham	12
15	Havering	11
11	Hackney	11

In [12]:
```
all_heat_count = london_df[['Latitude', 'Longitude', 'count']].groupby(['Latitude', 'Longitude']).sum().reset
_index().values.tolist()

map_london = folium.Map(location=london_location, zoom_start=11)
folium.TileLayer('cartodbpositron').add_to(map_london)
folium.Marker(london_location).add_to(map_london)
HeatMap(data=all_heat_count, radius=50).add_to(map_london)
folium.GeoJson(london_boroughs_json, style_function=boroughs_style, name='geojson').add_to(map_london)

for lat, lng, label in zip(london_df.Latitude, london_df.Longitude, london_df.Neighbourhood) :
    folium.CircleMarker(
        [lat, lng],
        radius=3,
        color='blue',
        popup=label,
        fill = True,
        fill_color='blue',
        fill_opacity=0.6
    ).add_to(map_london)

map_london
```

Out[12]:



Italian Restaurants Count

Let's check the Italian Restaurants in each neighborhood.

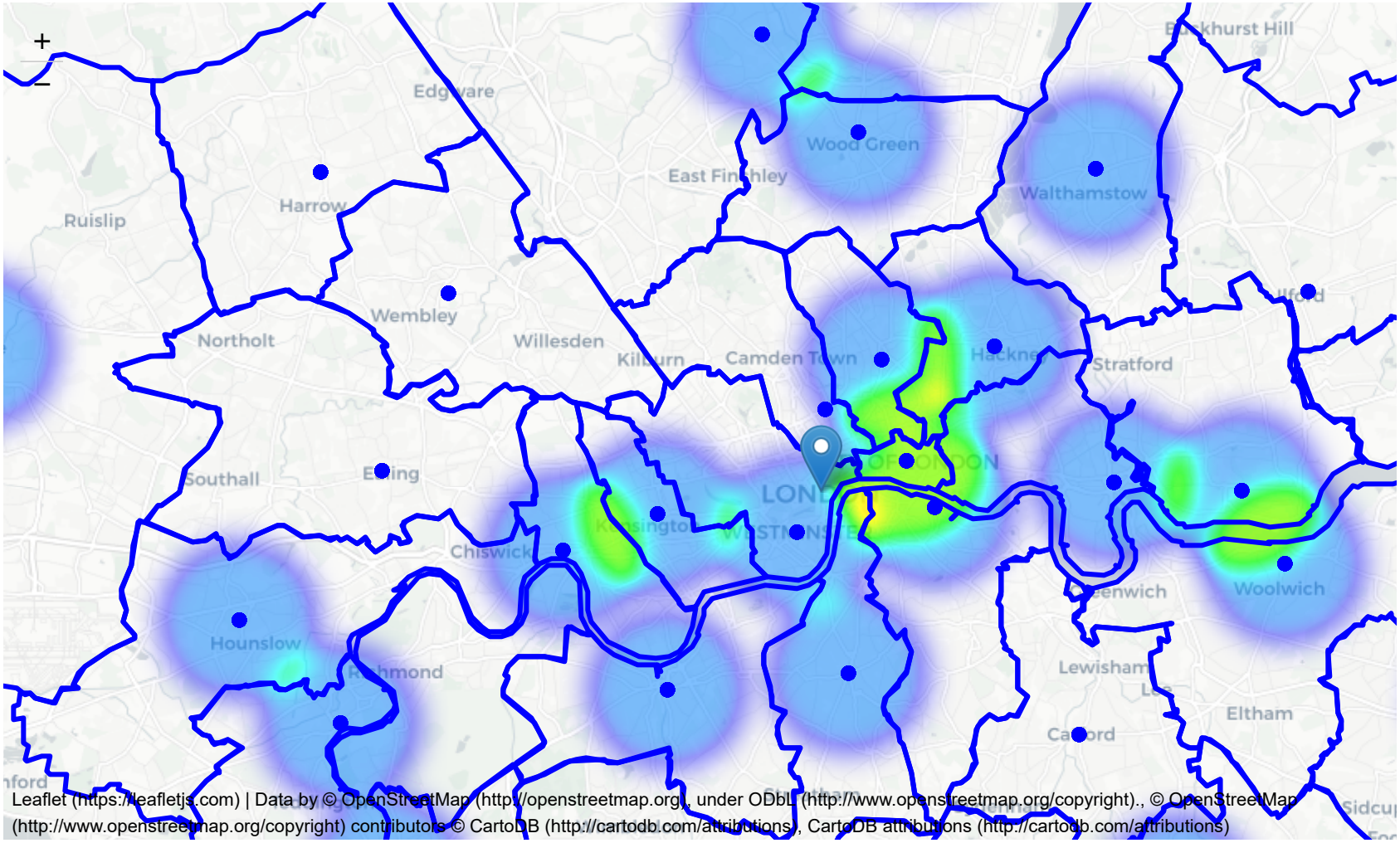
In [13]:
```
italian_df = london_df[london_df.Is_Italian].copy()

italian_df[['Neighbourhood', 'count']].groupby(['Neighbourhood']).sum().reset_index().sort_values(by=['count'], ascending=False)
```

Out[13]:

	Neighbourhood	count
19	Sutton	6
1	Bexley	5
3	City of London	4
20	Tower Hamlets	4
21	Waltham Forest	3
12	Islington	2
22	Wandsworth	2
18	Southwark	2
16	Newham	2
15	Lambeth	2
14	Kingston upon Thames	2
13	Kensington and Chelsea	2
23	Westminster	2
4	Enfield	2
2	Bromley	2
11	Hounslow	1
10	Hillingdon	1
9	Havering	1
8	Haringey	1
17	Richmond upon Thames	1
7	Hammersmith and Fulham	1
6	Hackney	1
5	Greenwich	1
0	Barnet	1

In [14]:
```
ita_heat_count = italian_df[['Latitude', 'Longitude', 'count']].groupby(['Latitude', 'Longitude']).sum().rese
t_index().values.tolist()

map_london = folium.Map(location=london_location, zoom_start=11)
folium.TileLayer('cartodbpositron').add_to(map_london)
folium.Marker(london_location).add_to(map_london)
HeatMap(data=ita_heat_count, radius=50).add_to(map_london)
folium.GeoJson(london_boroughs_json, style_function=boroughs_style, name='geojson').add_to(map_london)

for lat, lng, label in zip(london_df.Latitude, london_df.Longitude, london_df.Neighbourhood) :
    folium.CircleMarker(
        [lat, lng],
        radius=3,
        color='blue',
        popup=label,
        fill = True,
        fill_color='blue',
        fill_opacity=0.6
    ).add_to(map_london)

map_london
```

Out[14]:



Italian Restaurants Rating

Let's check the average rating of the Italian restaurants in each neighborhood.

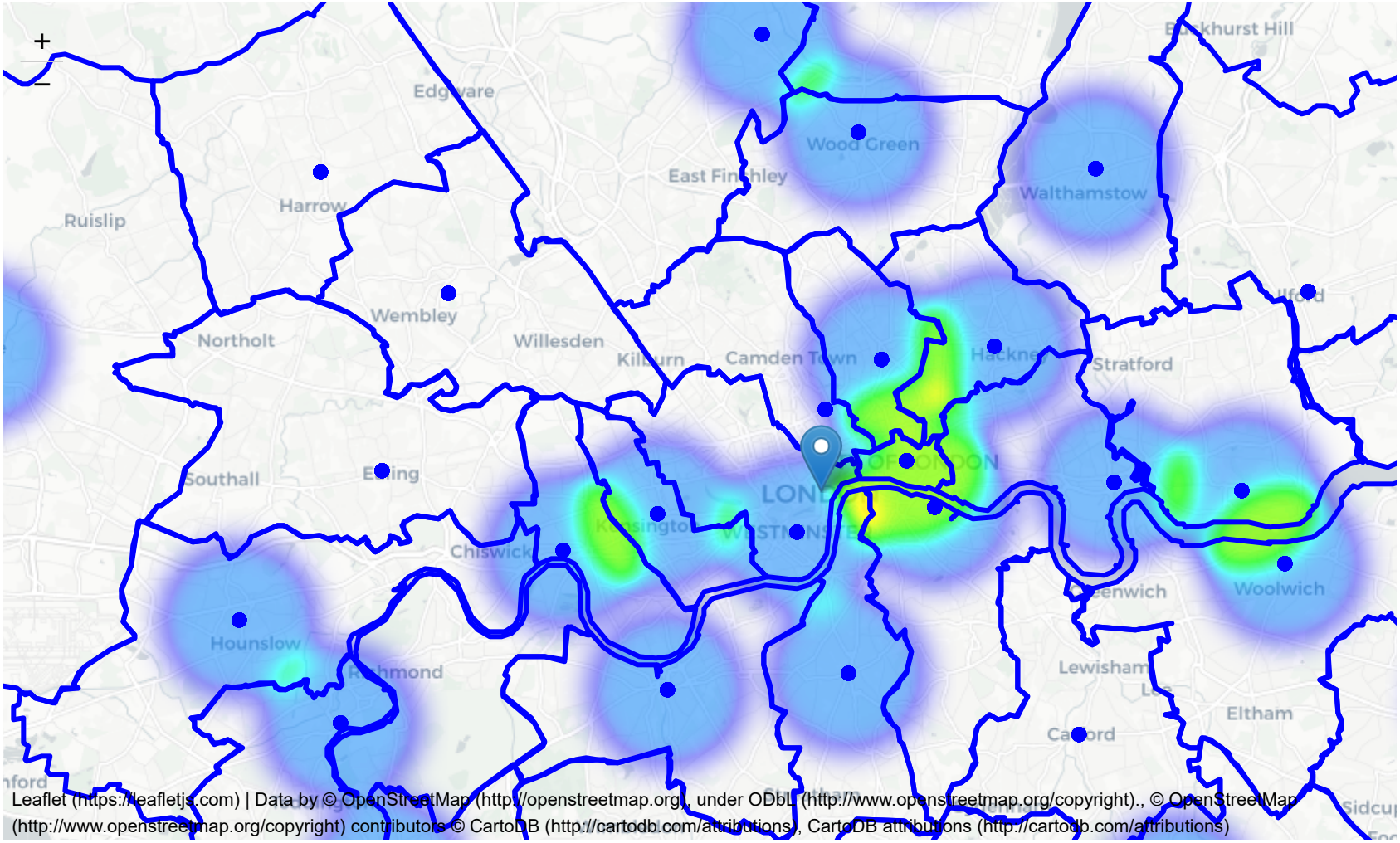
In [15]:
```
italian_df[['Neighbourhood', 'Rating']].groupby(['Neighbourhood']).mean().reset_index().sort_values(by=['Rating'], ascending=False)
```

Out[15]:

	Neighbourhood	Rating
18	Southwark	8.900000
23	Westminster	8.850000
13	Kensington and Chelsea	8.850000
3	City of London	8.500000
7	Hammersmith and Fulham	8.400000
8	Haringey	8.400000
0	Barnet	8.400000
22	Wandsworth	8.200000
6	Hackney	8.100000
12	Islington	8.100000
21	Waltham Forest	7.800000
15	Lambeth	7.650000
20	Tower Hamlets	7.425000
11	Hounslow	7.300000
4	Enfield	7.300000
17	Richmond upon Thames	7.300000
5	Greenwich	6.700000
9	Havering	6.700000
16	Newham	6.550000
19	Sutton	5.766667
1	Bexley	5.500000
14	Kingston upon Thames	3.650000
10	Hillingdon	0.000000
2	Bromley	0.000000

In [16]:
```
ita_heat_rating = italian_df[['Latitude', 'Longitude', 'Rating']].groupby(['Latitude', 'Longitude']).mean().r
eset_index().values.tolist()

map_london = folium.Map(location=london_location, zoom_start=11)
folium.TileLayer('cartodbpositron').add_to(map_london)
folium.Marker(london_location).add_to(map_london)
HeatMap(data=ita_heat_rating, radius=50).add_to(map_london)
folium.GeoJson(london_boroughs_json, style_function=boroughs_style, name='geojson').add_to(map_london)

for lat, lng, label in zip(london_df.Latitude, london_df.Longitude, london_df.Neighbourhood) :
    folium.CircleMarker(
        [lat, lng],
        radius=3,
        color='blue',
        popup=label,
        fill = True,
        fill_color='blue',
        fill_opacity=0.6
    ).add_to(map_london)

map_london
```

Out[16]:



As we can see from the analysis and visualization, we can see that **City of London** is a hot place for restaurants. This is a typical for most of the cities.

Based on that, we can include calculate the distance to the **City of London** as an attribute to each neighborhood.

In [17]:
```
def calculate_distance(row) :
    loc = (row['Latitude'], row['Longitude'])
    london_loc = (london_df[london_df.Neighbourhood == 'City of London']['Latitude'].iloc[0], london_df[londo
n_df.Neighbourhood == 'City of London']['Longitude'].iloc[0])
    return geopy.distance.distance(loc, london_loc).meters


london_df["Distance_To_Center"] = london_df.apply(lambda row : calculate_distance(row), axis=1)
```

In [18]:
```
london_df.head()
```

Out[18]:

	Neighbourhood	Latitude	Longitude	Area	Area-Radius	Restaurant_ID	Restaurant_Name	Category_
0	Barking and Dagenham	51.5607	0.1557	36078700.0	4405.486312	4bb9e6183db7b713faa0229a	Moby Dick	4bf58dd8d48988d1c49417
1	Barking and Dagenham	51.5607	0.1557	36078700.0	4405.486312	4eb064fb775bbddde7b4f546	Cosmo	4bf58dd8d48988d1c49417
2	Barking and Dagenham	51.5607	0.1557	36078700.0	4405.486312	50baf653183fee49a8b0b4cc	Lara Grill	4f04af1f2fb6e1c99f3db0
3	Barking and Dagenham	51.5607	0.1557	36078700.0	4405.486312	4e7a1f5414954a343fb58258	The Pipe Major	4bf58dd8d48988d1c49417
4	Barking and Dagenham	51.5607	0.1557	36078700.0	4405.486312	4c3218fa3896e21e07f2e790	The Compasses (Harvester)	52e81612bcbc57f1066b7a

Now we will prepare the dataset that will be used for our modeling.
For each neighborhood, we will aggregate:

1. Count of ALL restaurants.
2. Count of Italian restaurants.
3. Average rating of Italian restaurants.
4. Also we will include the distance to the City Center.

In [23]:
```
rest_count = london_df[['Neighbourhood', 'count']].groupby(['Neighbourhood']).sum().reset_index()
rest_count.rename(columns={'count' : 'Restaurant_Count'}, inplace = True)

italian_rate = italian_df[['Neighbourhood', 'Rating']].groupby(['Neighbourhood']).mean().reset_index()
italian_rate.rename(columns={'Rating' : 'Average_Ratings'}, inplace= True)

italian_count = italian_df[['Neighbourhood', 'count']].groupby(['Neighbourhood']).sum().reset_index()
italian_count.rename(columns={'count' : 'Italian_Count'}, inplace= True)

london_distance = london_df[['Neighbourhood', 'Distance_To_Center']].groupby(['Neighbourhood', 'Distance_To_C
enter']).size().reset_index(name='freq').drop('freq', axis=1)


dataset_df = pd.merge(rest_count, london_distance, how='left', on='Neighbourhood')
dataset_df = pd.merge(dataset_df, italian_count, how='left', on='Neighbourhood')
dataset_df = pd.merge(dataset_df, italian_rate, how='left', on='Neighbourhood')
dataset_df = dataset_df.fillna(0)
```

Now we have the final dataset for our K-Means clustering model. Let's normalize our dataset.

In [24]:
```
X = dataset_df.values[:,1:]
X = np.nan_to_num(X)
cluster_dataset = StandardScaler().fit_transform(X)
#cluster_dataset
```

Modeling

In [25]:
```
num_clusters = 5

k_means = KMeans(init="k-means++", n_clusters=num_clusters, n_init=12)
k_means.fit(cluster_dataset)
labels = k_means.labels_

print(labels)
```

```
[0 1 3 0 0 0 2 0 0 1 1 1 1 1 0 1 4 1 2 2 1 2 0 0 1 0 1 2 3 2 1 1 2]
```

Note that each row in our dataset represents a neighborhood, and therefore, each row is assigned a label.

In [26]:
```
dataset_df['Labels'] = labels
dataset_df.sort_values(by='Labels', ascending=True)
```

Out[26]:

	Neighbourhood	Restaurant_Count	Distance_To_Center	Italian_Count	Average_Ratings	Labels
0	Barking and Dagenham	12	17920.030817	0.0	0.000000	0
22	Lewisham	15	9257.906442	0.0	0.000000	0
25	Redbridge	18	12512.433585	0.0	0.000000	0
3	Brent	17	14003.045824	0.0	0.000000	0
4	Bromley	17	14654.616692	2.0	0.000000	0
5	Camden	24	2756.412095	0.0	0.000000	0
7	Croydon	17	16036.658074	0.0	0.000000	0
8	Ealing	19	15045.632474	0.0	0.000000	0
23	Merton	16	14594.407020	0.0	0.000000	0
14	Harrow	20	18735.296447	0.0	0.000000	0
20	Kingston upon Thames	16	19061.098358	2.0	3.650000	1
26	Richmond upon Thames	14	17898.540544	1.0	7.300000	1
17	Hounslow	21	19686.870384	1.0	7.300000	1
31	Wandsworth	19	9481.306493	2.0	8.200000	1
15	Havering	11	20486.772699	1.0	6.700000	1
12	Hammersmith and Fulham	17	10160.583093	1.0	8.400000	1
11	Hackney	11	4162.847895	1.0	8.100000	1
10	Greenwich	15	11287.461679	1.0	6.700000	1
9	Enfield	17	15410.781065	2.0	7.300000	1
30	Waltham Forest	18	10002.976870	3.0	7.800000	1
1	Barnet	17	12883.465693	1.0	8.400000	1
13	Haringey	16	9500.135755	1.0	8.400000	1
24	Newham	19	9695.637179	2.0	6.550000	1
27	Southwark	23	1566.402203	2.0	8.900000	2
29	Tower Hamlets	20	6023.360469	4.0	7.425000	2
32	Westminster	21	3723.168447	2.0	8.850000	2

	Neighbourhood	Restaurant_Count	Distance_To_Center	Italian_Count	Average_Ratings	Labels
19	Kensington and Chelsea	23	7273.068499	2.0	8.850000	2
18	Islington	19	2985.616195	2.0	8.100000	2
6	City of London	27	0.000000	4.0	8.500000	2
21	Lambeth	20	6322.539825	2.0	7.650000	2
28	Sutton	14	18520.635266	6.0	5.766667	3
2	Bexley	19	18156.865207	5.0	5.500000	3
16	Hillingdon	26	26823.258913	1.0	0.000000	4

Results and Discussion

As we can see from the results of our Analysis, London neighborhoods of **Labels 0** are the most interesting neighborhoods to open new Italian restaurant because they have the least Italian restaurants with low rating.

Let's check the neighborhoods in **cluster 4** and order the result based on the **Distance to the City Center**.

In [27]:
```
dataset_df[dataset_df['Labels'] == 0].sort_values(by='Distance_To_Center')
```

Out[27]:

	Neighbourhood	Restaurant_Count	Distance_To_Center	Italian_Count	Average_Ratings	Labels
5	Camden	24	2756.412095	0.0	0.0	0
22	Lewisham	15	9257.906442	0.0	0.0	0
25	Redbridge	18	12512.433585	0.0	0.0	0
3	Brent	17	14003.045824	0.0	0.0	0
23	Merton	16	14594.407020	0.0	0.0	0
4	Bromley	17	14654.616692	2.0	0.0	0
8	Ealing	19	15045.632474	0.0	0.0	0
7	Croydon	17	16036.658074	0.0	0.0	0
0	Barking and Dagenham	12	17920.030817	0.0	0.0	0
14	Harrow	20	18735.296447	0.0	0.0	0

As we can see, neighborhood ***Camden*** is the nearest neighborhood to the City Center. However, it has many restaurants which might be a downside because of competition for other restaurants serving other cuisines.

Second in the list is neighborhood ***Lewisham*** which might be better option as it has less number of restaurants despite being little far from the City Center compared to **Camden**

Conclusion

The objective of this report is to identify the neighborhoods in London with least number of restaurants specially Italian restaurants and close to the city center. This should help the report's stackholders to narrow down the neighborhoods to search for openning a new Italian restaurant.

In order to achieve this objective, we gathered geo-data of London neighborhoods and used Foursquare endpoints to gather information about the restaurants in each neighborhood. Also, we used maps visualizaiton to help the user with the data exploration stage.

Based on the neighborhoods clustering we did, we provided 2 recomendations to the end-user to choose from.

For sure, there are many factors that might influence the end-user decision (i.e. other business in the area, easy access and transportation, etc) which is out of the scope of this report.