IBM DataScience Professional Certificate

Applied DataScience Capstone - The Battle of Neighborhoods

By Kevin Gilson.

November 2020

Table of contents

Introduction - A Business Problem

Young entrepreneurs are always in demand of good advices, and always looking towards the keys of success. DataScience can help them analyzing the market, and get information such as:

- Which places of a city are the most prolific?
- Which types of business are present in which areas?
- Which types of business are lacking in which areas?
- ...

This kind of information are particularly relevant for people looking to open food related businesses.

With almost 9 millions people in 2020, London is a big city. Not only is it the capital of the United Kingdom, it is also one of the top financial places in Europe, and a hugely touristic city. With that in mind, what could possibly leverage a young entrepreneur looking to open a restaurant in London, if he wanted to be successful? Where are located the **hot** zone of the city? Which place would be best to maximize his profit?

Thanks to Data Science, we have a way to analyze raw data and come up with suggestions to help this young man, grow as a successful entrepreneur.

In this projet, we will first help this entrepreneur decide on whether it is best to open an **Italian** restaurant, or a **French** one. After that, we will find the best place around the Center of London for him to start his business.

Data

London is split into Borough, holding various Neighborhoods.

In order to leverage further geographical data, we will first scrape Wikipedia to get the complete list of each Borough and Neighborhoods. We will also retrieve the list of borough and their characteristics, such as whether they are located in **Inner** or **Greater** London.

Then, we will use Python libraries to link each Neighborhoods to its relative geographical coordinates, and see whether these information are enough for us to continue. If not, we will proceed mathematically in order define our own grid of research around the center of the city.

After that, we will leverage the Foursquare location data to retrieve the restaurants near by each zone, and come up with suggestions on the best places to open a new business.

Our primary data, scraped from Wikipedia, for the **Neighborhoods**, adopt the form of:

Location	London Borough	Post Town	Postcode District	Dial code	OS Grid Ref
Abbey Wood	Bexley, Greenwich [7]	LONDON	SE2	020	TQ465785
Acton	Ealing, Hammersmith and Fulham[8]	LONDON	W3, W4	020	TQ205805
Addington	Croydon[8]	CROYDON	CR0	020	TQ375645
Addiscombe	Croydon[8]	CROYDON	CR0	020	TQ345665
Albany Park	Bexley	BEXLEY, SIDCUP	DA5, DA14	020	TQ478728

For the **Borough**, the data follow the same pattern as the table below:

Borough	Inner	Status	Local authority	Political control	Headquarters	Area (sqmi)	Population (2013 est)	Coordinates	Nr. in map
Barking and Dagenham [note 1]			Barking and Dagenham London Borough Council	Labour	Town Hall, 1 Town Square	13.93	194,352	51.5607°N 0.1557°E	25
Barnet			Barnet London Borough Council	Conservative	Barnet House, 2 Bristol Avenue, Colindale	33.49	369,088	51.6252°N 0.1517°W	31
Bexley			Bexley London Borough Council	Conservative	Civic Offices, 2 Watling Street	23.38	236,687	51.4549°N 0.1505°E	23
Brent			Brent London Borough Council	Labour	Brent Civic Centre, Engineers Way	16.70	317,264	51.5588°N 0.2817°W	12
Camden	V		Camden London Borough Council	Labour	Camden Town Hall, Judd Street	8.40	229,719	51.5290°N 0.1255°W	11

As we can see, the data will need to be cleansed, but it is also important to note that OS Grid References can quite easily be converted into coordinates.

On top of that, we might also retrieve GeoJSON coordinates for the bouroughs and other boundaries.

Scraping the WikiPedia page

As the first step, we will need to scrape the WikiPedia page in order to retrieve its table, and therefore datas about the London Districts and Areas. In order to do so, we will:

1. Scrape the page using BeautifulSoup;
2. Convert the OS Grid References found to coordinates (latitude, longitude) using the OSGRridConverter library; missing OS Grid References will results in NaN values for their coordinates;
3. Check the shape of the recovered dataframe

Let's first do it for our **Neighborhood** (*Area*) data:

```
The resulting dataframe has a shape of: (532, 8)
```

	Location	London Borough	Post Town	Post District	Dial Code	OS Grid Ref	Latitude	Longitude
0	Abbey Wood	Bexley, Greenwich [7]	LONDON	SE2	020	TQ465785	51.486484	0.109318
1	Acton	Ealing, Hammersmith and Fulham[8]	LONDON	W3, W4	020	TQ205805	51.510591	-0.264585
2	Addington	Croydon[8]	CROYDON	CR0	020	TQ375645	51.362934	-0.025780
3	Addiscombe	Croydon[8]	CROYDON	CR0	020	TQ345665	51.381625	-0.068126
4	Albany Park	Bexley	BEXLEY, SIDCUP	DA5, DA14	020	TQ478728	51.434929	0.125663

Now let's scrape the **Borough** data and have a look at them:

```
The resulting dataframe has a shape of: (33, 10)
```

	Borough	Inner	Status	Local authority	Political control	Headquarters	Area (sqmi)	Population (2013 est)	Coordinates	Nr. in map
0	Barking and Dagenham [note 1]			Barking and Dagenham London Borough Council	Labour	Town Hall, 1 Town Square	13.93	194,352	51°33′39″N 0°09′21″E / 51.5607°N 0.1557°E /...	25
1	Barnet			Barnet London Borough Council	Conservative	Barnet House, 2 Bristol Avenue, Colindale	33.49	369,088	51°37′31″N 0°09′06″W / 51.6252°N 0.1517°W /...	31
2	Bexley			Bexley London Borough Council	Conservative	Civic Offices, 2 Watling Street	23.38	236,687	51°27′18″N 0°09′02″E / 51.4549°N 0.1505°E /...	23
3	Brent			Brent London Borough Council	Labour	Brent Civic Centre, Engineers Way	16.70	317,264	51°33′32″N 0°16′54″W / 51.5588°N 0.2817°W /...	12
4	Bromley			Bromley London Borough Council	Conservative	Civic Centre, Stockwell Close	57.97	317,899	51°24′14″N 0°01′11″E / 51.4039°N 0.0198°E /...	20

Columns cleansing

We can see that the resulting dataframes aren't looking their best; let's upgrade it a bit by:

1. First we will drop columns that won't be used later;
2. Then we will rename a few of the remaining ones into clearer names;
3. Finally we will reorder the columns.

Let's have a look at the resulting **Neighborhood** data:

	Borough	Neighborhood	Town	Latitude	Longitude
0	Bexley, Greenwich [7]	Abbey Wood	LONDON	51.486484	0.109318
1	Ealing, Hammersmith and Fulham[8]	Acton	LONDON	51.510591	-0.264585
2	Croydon[8]	Addington	CROYDON	51.362934	-0.025780
3	Croydon[8]	Addiscombe	CROYDON	51.381625	-0.068126
4	Bexley	Albany Park	BEXLEY, SIDCUP	51.434929	0.125663

Let's do the same for the **Borough**

	Borough	Inner	Area	Population	Coordinates
0	Barking and Dagenham [note 1]		13.93	194,352	51°33′39″N 0°09′21″E / 51.5607°N 0.1557°E /...
1	Barnet		33.49	369,088	51°37′31″N 0°09′06″W / 51.6252°N 0.1517°W /...
2	Bexley		23.38	236,687	51°27′18″N 0°09′02″E / 51.4549°N 0.1505°E /...
3	Brent		16.70	317,264	51°33′32″N 0°16′54″W / 51.5588°N 0.2817°W /...
4	Bromley		57.97	317,899	51°24′14″N 0°01′11″E / 51.4039°N 0.0198°E /...

Data cleansing

We can still see a few problems with our data, such as:

- Some values contains WikiPedia links references (i.e. "[1]");
- Some Neighborhoods are affected to two or more Towns;
- Some values are followed by explainations in between paratheses;
- The case of the data are not normalized, mixing upper and lower cases.

Let's use regular expression to clean the data of brackets, parentheses, and others. And then, let's capitalize each value (i.e. "VALUES" and "values" will become "Values").

On our **Neighborhood** data, we obtain the following result:

```
(532, 5)
```

	Borough	Neighborhood	Town	Latitude	Longitude
0	Bexley	Abbey Wood	London	51.486484	0.109318
1	Ealing	Acton	London	51.510591	-0.264585
2	Croydon	Addington	Croydon	51.362934	-0.025780
3	Croydon	Addiscombe	Croydon	51.381625	-0.068126
4	Bexley	Albany Park	Bexley	51.434929	0.125663

Now that our columns data are appropriate, let's check their data types:

```
Borough          object
Neighborhood     object
Town             object
Latitude        float64
Longitude       float64
dtype: object
```

For the **Neighborhood** data, the types are correct.

Let's apply the same process to our **Borough** data:

```
(33, 5)
```

	Borough	Inner	Area	Population	Coordinates
0	Barking and Dagenham		13.93	194,352	51°33′39″N 0°09′21″E / 51.5607°N 0.1557°E /...
1	Barnet		33.49	369,088	51°37′31″N 0°09′06″W / 51.6252°N 0.1517°W /...
2	Bexley		23.38	236,687	51°27′18″N 0°09′02″E / 51.4549°N 0.1505°E /...
3	Brent		16.70	317,264	51°33′32″N 0°16′54″W / 51.5588°N 0.2817°W /...
4	Bromley		57.97	317,899	51°24′14″N 0°01′11″E / 51.4039°N 0.0198°E /...

We can see that the coordinates aren't appropriate. Let's have a complete look at how they look like:

```
'51°33’39”N 0°09’21”E\ufeff / \ufeff51.5607°N 0.1557°E\ufeff / 51.5607; 0.1557\ufeff'
```

First, we need to remove the unicode characters *ufeff*

```
'51°33’39”N 0°09’21”E / 51.5607°N 0.1557°E / 51.5607; 0.1557'
```

Now we can split it based on the slash character, and only keep the last column (i.e. coordinates under the decimal form):

	Borough	Inner	Area	Population	Coordinates
0	Barking and Dagenham		13.93	194,352	51.5607; 0.1557
1	Barnet		33.49	369,088	51.6252; -0.1517
2	Bexley		23.38	236,687	51.4549; 0.1505
3	Brent		16.70	317,264	51.5588; -0.2817
4	Bromley		57.97	317,899	51.4039; 0.0198

Perfect. Now let's create two columns, Latitude and Longitude, by splitting the coordinates on the semi-colon:

	Borough	Inner	Area	Population	Coordinates	Latitude	Longitude
0	Barking and Dagenham		13.93	194,352	51.5607; 0.1557	51.5607	0.1557
1	Barnet		33.49	369,088	51.6252; -0.1517	51.6252	-0.1517
2	Bexley		23.38	236,687	51.4549; 0.1505	51.4549	0.1505
3	Brent		16.70	317,264	51.5588; -0.2817	51.5588	-0.2817
4	Bromley		57.97	317,899	51.4039; 0.0198	51.4039	0.0198

Finally we will drop the Coordinates column, as it is now useless:

	Borough	Inner	Area	Population	Latitude	Longitude
0	Barking and Dagenham		13.93	194,352	51.5607	0.1557
1	Barnet		33.49	369,088	51.6252	-0.1517
2	Bexley		23.38	236,687	51.4549	0.1505
3	Brent		16.70	317,264	51.5588	-0.2817
4	Bromley		57.97	317,899	51.4039	0.0198

The **Inner** column is a boolean, with values being either *Y*, *(Y)*, or emptyness. Let's replace them by 0 and 1:

	Borough	Inner	Area	Population	Latitude	Longitude
0	Barking and Dagenham	0	13.93	194,352	51.5607	0.1557
1	Barnet	0	33.49	369,088	51.6252	-0.1517
2	Bexley	0	23.38	236,687	51.4549	0.1505
3	Brent	0	16.70	317,264	51.5588	-0.2817
4	Bromley	0	57.97	317,899	51.4039	0.0198

Let's now have a look at the data types:

```
Borough       object
Inner          int64
Area          object
Population    object
Latitude      object
Longitude     object
dtype: object
```

We can see that 4 columns, which clearly contains numbers, are of the object datatype:

	Area	Population	Latitude	Longitude
0	13.93	194,352	51.5607	0.1557
1	33.49	369,088	51.6252	-0.1517
2	23.38	236,687	51.4549	0.1505
3	16.70	317,264	51.5588	-0.2817
4	57.97	317,899	51.4039	0.0198

We need to change that and check that it is now all good:

```
Borough        object
Inner           int64
Area          float64
Population    float64
Latitude      float64
Longitude     float64
dtype: object
```

Now let's convert the area to the metric system, and therefore square meters:

	Borough	Inner	Area	Population	Latitude	Longitude
0	Barking and Dagenham	0	3.607853e+07	194352.0	51.5607	0.1557
1	Barnet	0	8.673870e+07	369088.0	51.6252	-0.1517
2	Bexley	0	6.055392e+07	236687.0	51.4549	0.1505
3	Brent	0	4.325280e+07	317264.0	51.5588	-0.2817
4	Bromley	0	1.501416e+08	317899.0	51.4039	0.0198

Let's also compute the radius, in meters:

	Borough	Inner	Area	Population	Latitude	Longitude	Radius
0	Barking and Dagenham	0	3.607853e+07	194352.0	51.5607	0.1557	3388.827846
1	Barnet	0	8.673870e+07	369088.0	51.6252	-0.1517	5254.501527
2	Bexley	0	6.055392e+07	236687.0	51.4549	0.1505	4390.320264
3	Brent	0	4.325280e+07	317264.0	51.5588	-0.2817	3710.497851
4	Bromley	0	1.501416e+08	317899.0	51.4039	0.0198	6913.143932

Finally, we can convert the area to squared kilometers for reading purposes:

	Borough	Inner	Area	Population	Latitude	Longitude	Radius
0	Barking and Dagenham	0	36.078534	194352.0	51.5607	0.1557	3388.827846
1	Barnet	0	86.738702	369088.0	51.6252	-0.1517	5254.501527
2	Bexley	0	60.553922	236687.0	51.4549	0.1505	4390.320264
3	Brent	0	43.252801	317264.0	51.5588	-0.2817	3710.497851
4	Bromley	0	150.141611	317899.0	51.4039	0.0198	6913.143932

Missing values

Before going further, we want to make sure that we have the coordinates of each neighborhoods. Let's check the Latitudes and Longitudes on our **Neighborhood** data:

```
Missing data in column Latitude, at index:
Int64Index([53, 232], dtype='int64')
Missing data in column Longitude, at index:
Int64Index([53, 232], dtype='int64')
```

We can see that two neighborhoods are missing their coordinates.

Using the GeoCoder library, let's retrieve their latitudes and longitudes, then check if there is any remaining missing values:

```
Remaining missing values in Neighborhood data: False
```

Let's check if there is any missing values in **Borough** data

```
No missing data found.
```

Perfect, let's have a quick look at our two dataframes

```
Neighborhood data shape: (532, 5)
```

	Borough	Neighborhood	Town	Latitude	Longitude
0	Bexley	Abbey Wood	London	51.486484	0.109318
1	Ealing	Acton	London	51.510591	-0.264585
2	Croydon	Addington	Croydon	51.362934	-0.025780
3	Croydon	Addiscombe	Croydon	51.381625	-0.068126
4	Bexley	Albany Park	Bexley	51.434929	0.125663
5	Redbridge	Aldborough Hatch	Ilford	51.585581	0.099459
6	City of London	Aldgate	London	51.514885	-0.078356
7	Westminster	Aldwych	London	51.512819	-0.117388
8	Brent	Alperton	Wembley	51.537976	-0.292401
9	Bromley	Anerley	London	51.408585	-0.066989

```
Borough data shape: (33, 7)
```

	Borough	Inner	Area	Population	Latitude	Longitude	Radius
0	Barking and Dagenham	0	36.078534	194352.0	51.5607	0.1557	3388.827846
1	Barnet	0	86.738702	369088.0	51.6252	-0.1517	5254.501527
2	Bexley	0	60.553922	236687.0	51.4549	0.1505	4390.320264
3	Brent	0	43.252801	317264.0	51.5588	-0.2817	3710.497851
4	Bromley	0	150.141611	317899.0	51.4039	0.0198	6913.143932
5	Camden	1	21.755900	229719.0	51.5290	-0.1255	2631.561911
6	Croydon	0	86.531503	372752.0	51.3714	-0.0977	5248.221870
7	Ealing	0	55.529345	342494.0	51.5130	-0.3089	4204.228765
8	Enfield	0	82.206223	320524.0	51.6538	-0.0799	5115.374215
9	Greenwich	1	47.344983	264008.0	51.4892	0.0648	3882.058222

Exploring the dataset

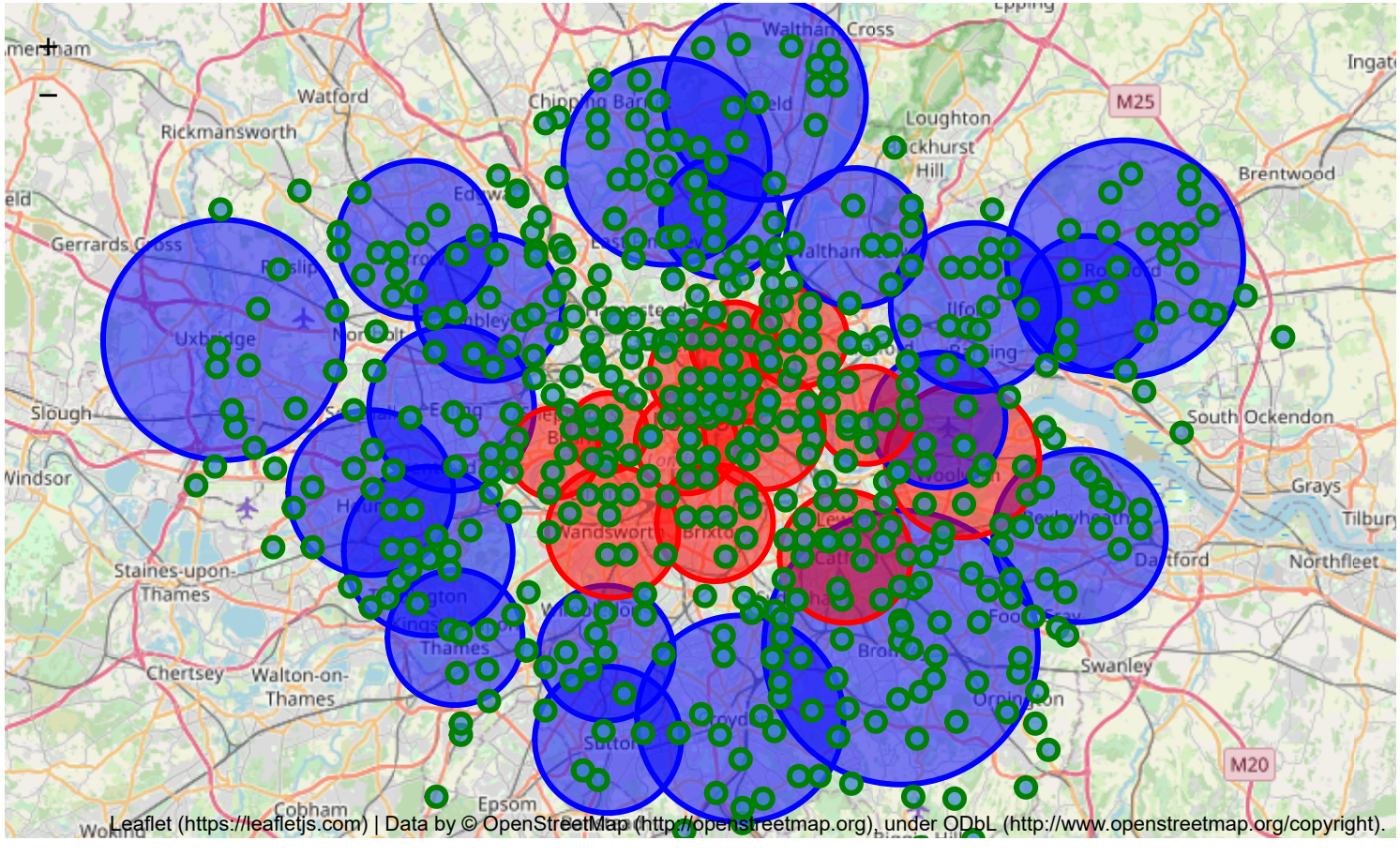
Let's have a first look at how our dataset look, by looking at its shape, and list its Neighborhoods and Towns:

```
The dataframe has 37 boroughs, 64 neighborhoods, and 532 towns.

The boroughs are the following:
['Bexley' 'Ealing' 'Croydon' 'Redbridge' 'City of London' 'Westminster'
 'Brent' 'Bromley' 'Islington' 'Havering' 'Barnet' 'Enfield' 'Wandsworth'
 'Southwark' 'Barking and Dagenham' 'Richmond upon Thames' 'Newham'
 'Sutton' 'Lewisham' 'Harrow' 'Camden' 'Kingston upon Thames'
 'Tower Hamlets' 'Greenwich' 'Haringey' 'Hounslow' 'Lambeth'
 'Kensington and Chelseahammersmith and Fulham' 'Waltham Forest'
 'Kensington and Chelsea' 'Merton' 'Hillingdon' 'Hackney'
 'Islington & City' 'Hammersmith and Fulham' 'Camden and Islington'
 'Haringey and Barnet']

The towns are the following:
['London' 'Croydon' 'Bexley' 'Ilford' 'Wembley' 'Westerham' 'Hornchurch'
 'Barnet' 'Barking' 'Bexleyheath' 'Dartford' 'Beckenham' 'Dagenham'
 'Wallington' 'Harrow' 'Sutton' 'Belvedere' 'Surbiton' 'Bromley' 'Sidcup'
 'Enfield' 'Brentford' 'Edgware' 'Carshalton' 'Romford' 'Sutton/Merton'
 'Orpington' 'Chessington' 'Chislehurst' 'Erith' 'West Wickham'
 'Kingston Upon Thames' 'Coulsdon' 'Uxbridge' 'Hounslow' 'Upminster'
 'Sevenoaks' 'Feltham' 'Welling' 'Pinner' 'Twickenham' 'Teddington'
 'Greenford' 'Chigwell' 'Richmond' 'Hampton' 'Hayes' 'West Drayton'
 'Isleworth' 'Kenley' 'Keston' 'Morden' 'Mitcham' 'New Malden' 'Northolt'
 'Northwood' 'Southall' 'Worcester Park' 'Purley' 'Rainham' 'Ruislip'
 'South Croydon' 'Stanmore' 'Thornton Heath']
```

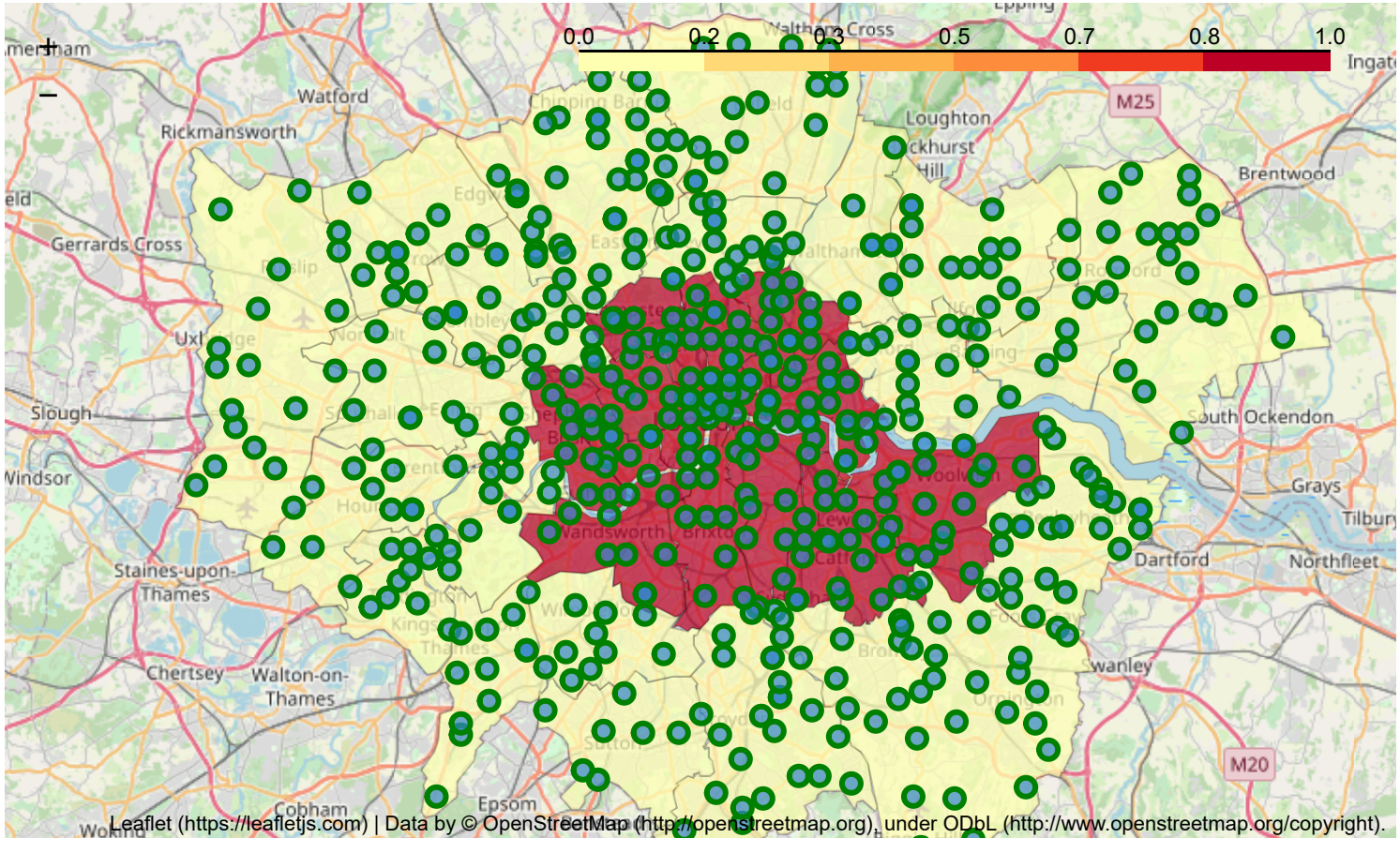
Considering that we are analyzing London, it might be appropriate to retrieve its coordinates:

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

Now let's plot the neighborhoods on a map of London, in order to get a better view of the situation; we will leverage the coordinates that we just found to center the map. We will also use the radius we calculated to have a approximate view of the boroughs area:



As we can see, circling the borough isn't the most appropriate way to visualize neighborhoods. Let's try with a GeoJSON of the borough limits:



It is already much clearer. And as we can see, the neighborhoods are pretty spread appart.

London is a big city, and even the inner part of it is huge. Therefore, it migth be best to create a sub-set of the data focusing on the City of London borough. Let's first create a new dataset with a dummy column indicating whether a neighborhood is part of the City of London or not:

```
Dataframe shape: (180, 6)
```

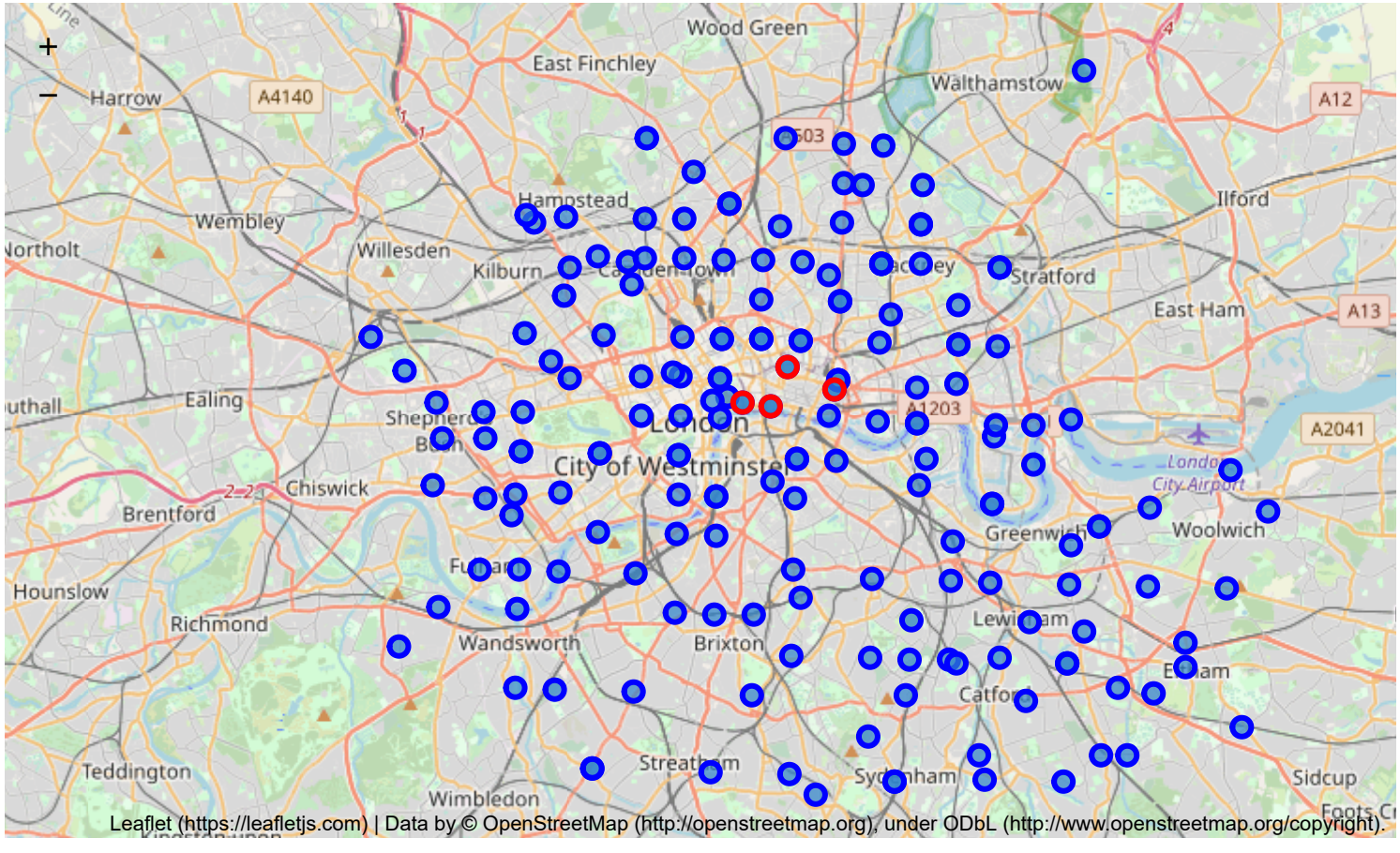
	Borough	Neighborhood	Town	Latitude	Longitude	City
0	City of London	Aldgate	London	51.514885	-0.078356	1
1	City of London	Barbican	London	51.519660	-0.095466	1
2	City of London	Blackfriars	London	51.510767	-0.101607	1
3	City of London	Temple	London	51.511828	-0.111659	1
4	Westminster	Aldwych	London	51.512819	-0.117388	0

Let's also create another one containing only the neighborhoods for the City of London:

```
Dataframe shape: (4, 5)
```

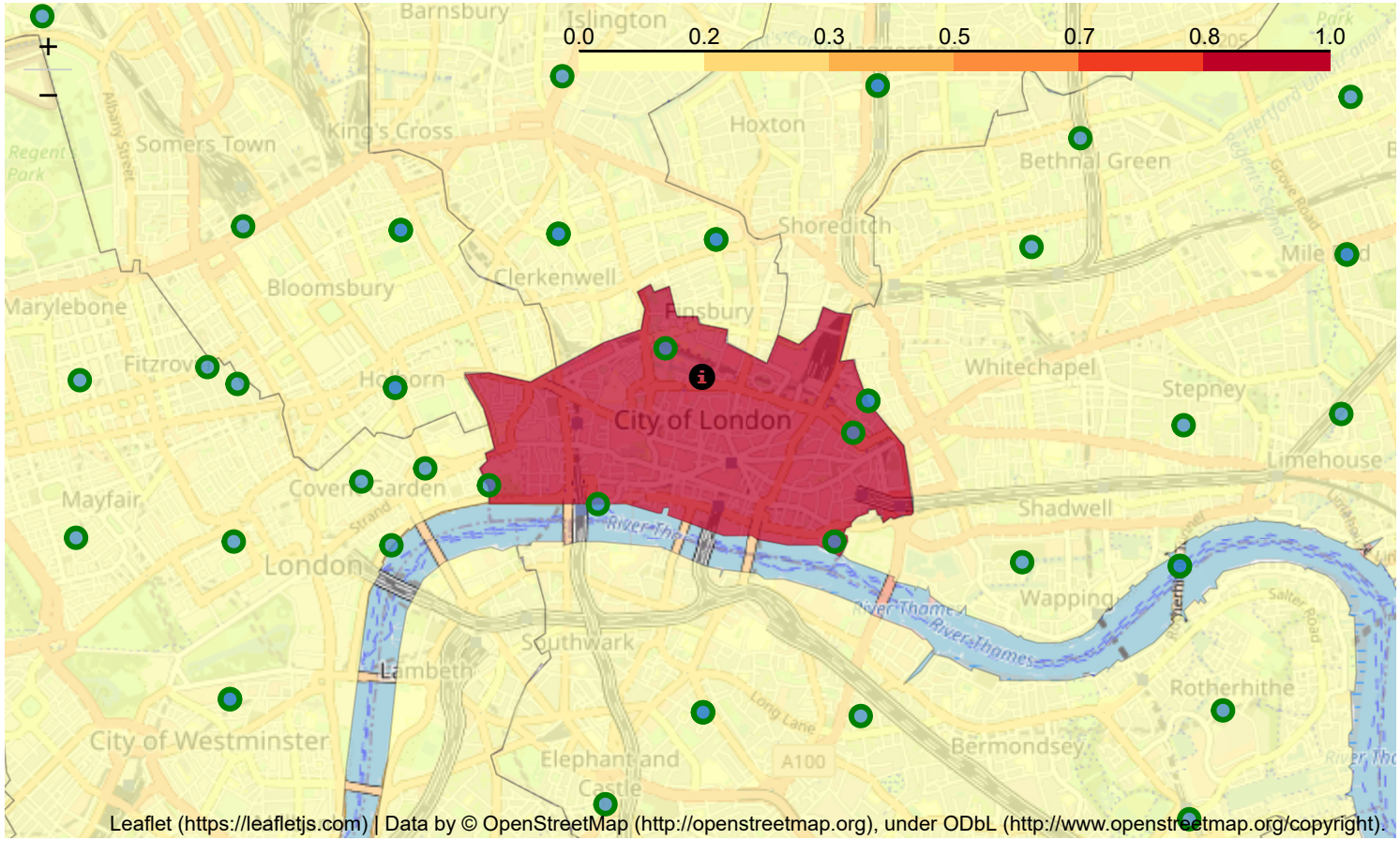
	Borough	Neighborhood	Town	Latitude	Longitude
0	City of London	Aldgate	London	51.514885	-0.078356
1	City of London	Barbican	London	51.519660	-0.095466
2	City of London	Blackfriars	London	51.510767	-0.101607
3	City of London	Temple	London	51.511828	-0.111659

Let's now compare the two dataset on a map, by plotting first every neighborhoods in blue, then turning red the ones in London city:



It is now clear that we need to focus more on areas around the City of London. If we don't, our results will be too spread appart and won't be relevant.

Let's have a quick look at how neighborhoods are spread around the City of London, hereby in red:



We can see that the City of London has few neighborhoods, but the other boroughs contains some that are pretty close. Let's try another approach and see whether it gives us better results at splitting the area.

We will create a grid of cells covering the City of London, with a radius of 5km, therefore covering an approximate area of 10x10 kilometers around the City of London. For each grid, we will determine the coordinates of its centroids, under the form of Latitude/Longitude, but also X/Y carthesian coordinates. Those will be used to calculate distances. Our neighborhoods will be defined as circular areas with a radius of 250 meters, centers being 500m appart.

Now let's generate a grid of areas around the center of the City of London

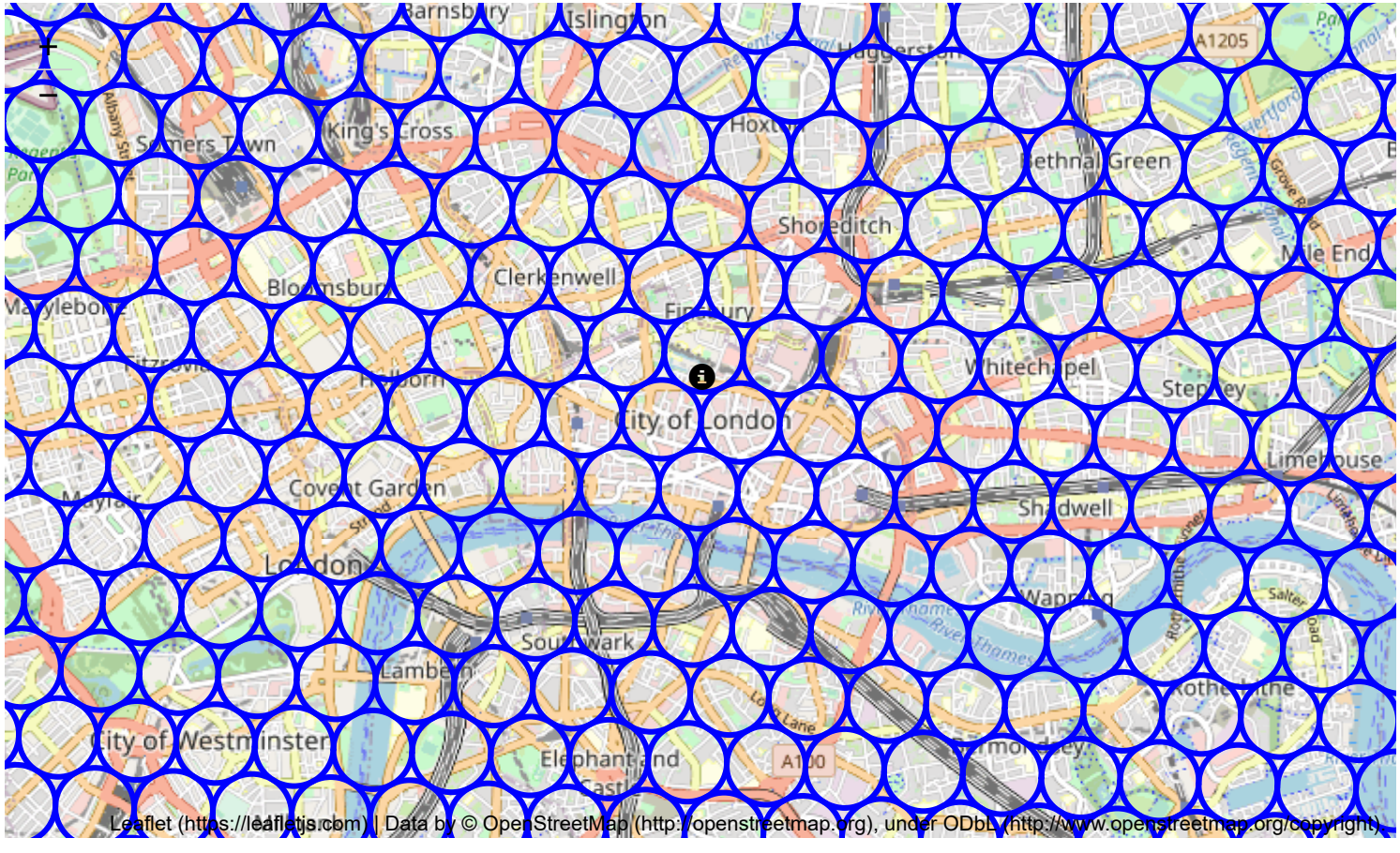
Let's create a **hexagonal grid of cells**: we offset every other row, and adjust vertical row spacing so that **every cell center is equally distant from all it's neighbors**.

After that, we will store the results into a dataframe:

```
364 candidate neighborhood centers generated.
```

	Latitude	Longitude	Distance from Center	X	Y
0	51.473259	-0.116492	4993.746089	700250.415693	5.706399e+06
1	51.473082	-0.109302	4866.980583	700750.415693	5.706399e+06
2	51.472904	-0.102112	4789.311015	701250.415693	5.706399e+06
3	51.472726	-0.094922	4763.139721	701750.415693	5.706399e+06
4	51.472548	-0.087733	4789.311015	702250.415693	5.706399e+06

Let's visualize the data we have so far, by plotting our areas around the center of the city:



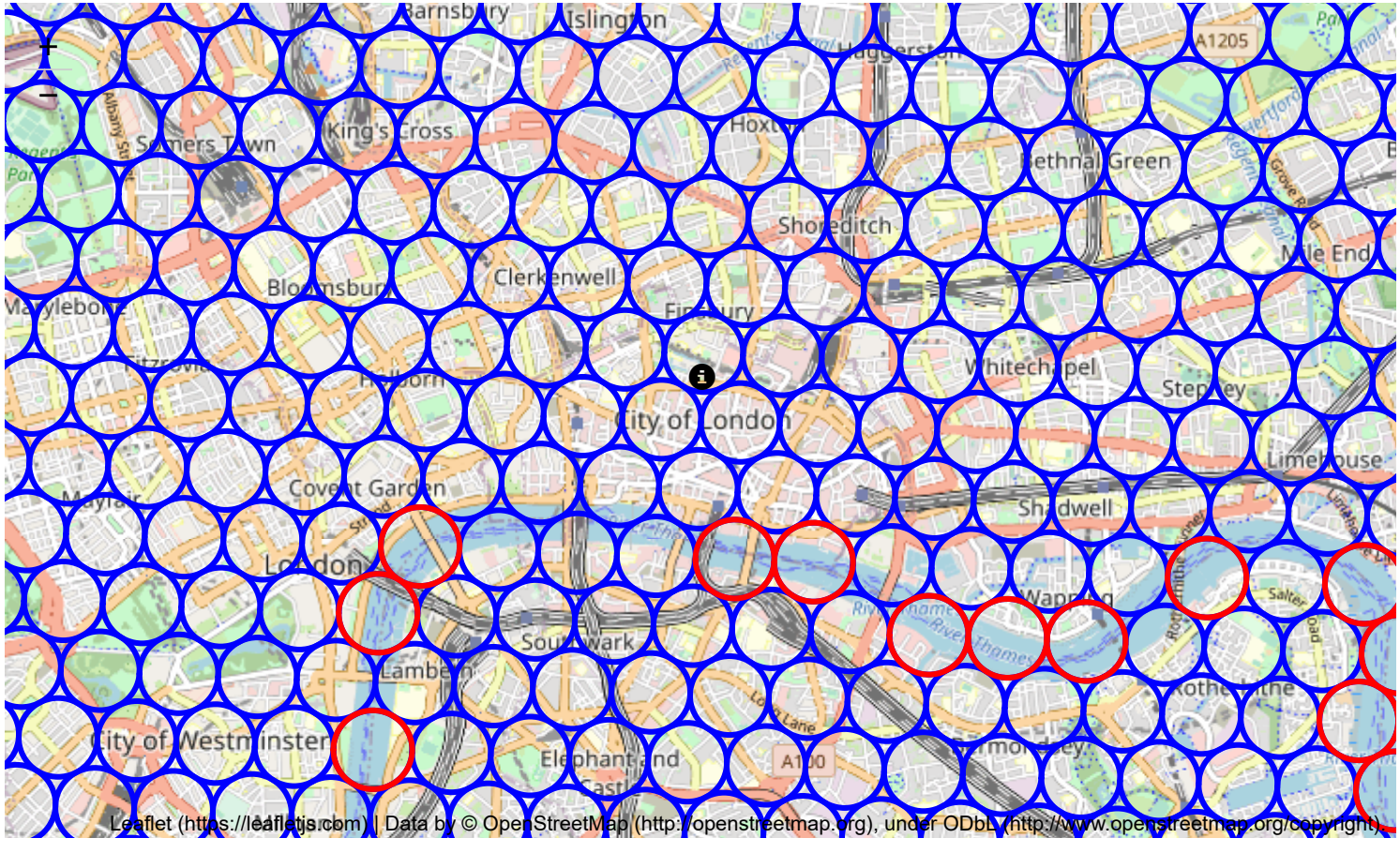
Now we can already see one problem with our areas: the River Thames. Indeed, restaurants in the middle of the river would'nt be practical.

Let's retrieve the River Thames coordinates using a GeoJSON, then list all areas within it, and storing them into a dataframe:

```
14 coordinates to be removed out of 364
```

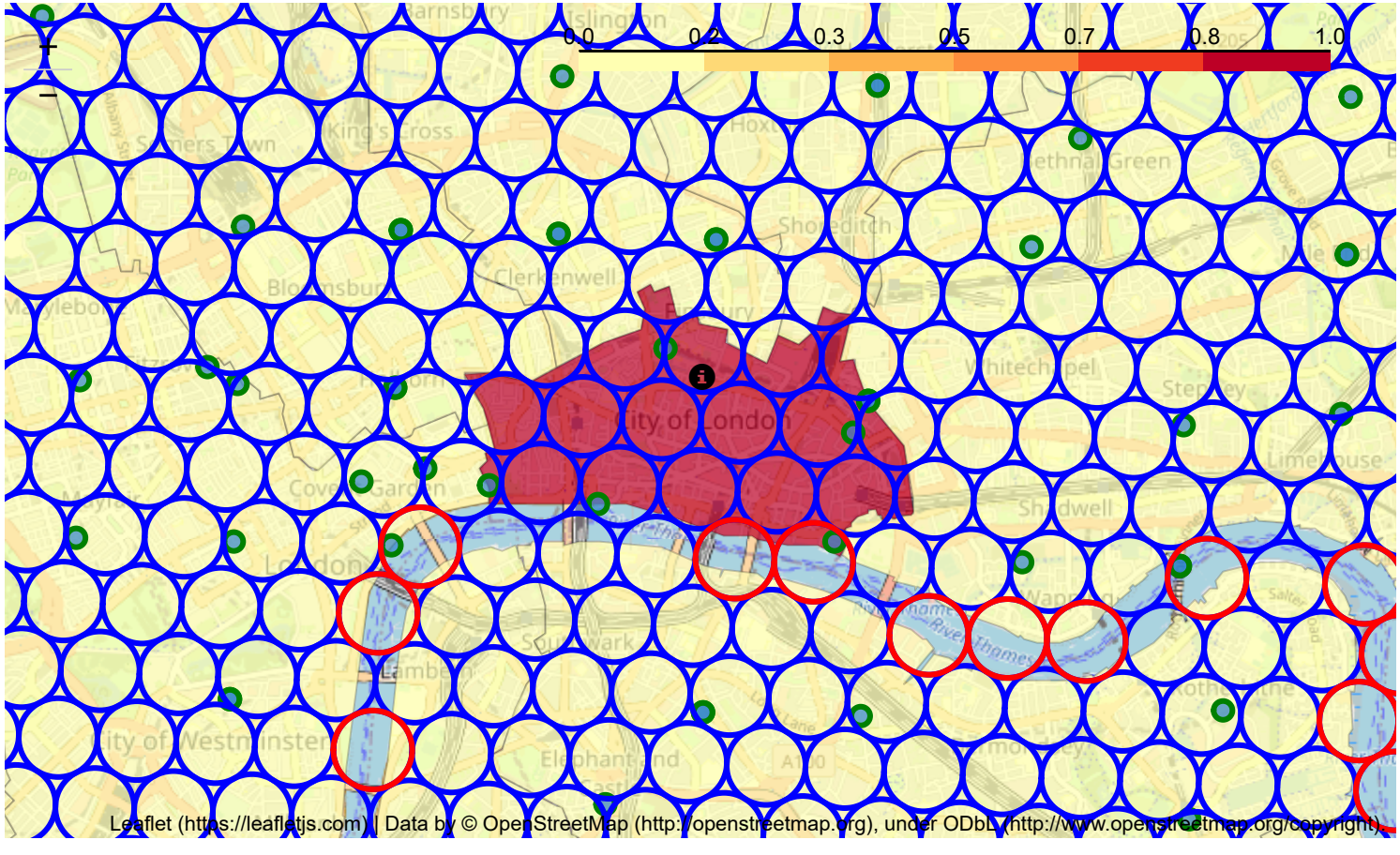
	Latitude	Longitude	Distance from Center	X	Y
0	51.485366	-0.133735	4422.951503	699000.415693	5.707698e+06
1	51.496768	-0.122212	2947.456531	699750.415693	5.708997e+06
2	51.494434	-0.028701	4993.746089	706250.415693	5.708997e+06
3	51.498413	-0.032044	4589.389938	706000.415693	5.709430e+06
4	51.504545	-0.121722	2384.848004	699750.415693	5.709863e+06

Let's now visualize these on the previous map, but highlighting the bad areas in red:



Perfect.

Now it could be great to superpose these on our previous maps containing the bourough boundaries, still keeping the City of London in red:



Looking good. We can clearly see that our areas covers the whole City of London, but also spread nicely around it. We will be able to use in order to find the best places for our restaurant.

But first, let's store our areas coordinates into a dataframe:

```
Dataframe shape: (350, 5)
```

	Latitude	Longitude	Distance from Center	X	Y
0	51.473259	-0.116492	4993.746089	700250.415693	5.706399e+06
1	51.473082	-0.109302	4866.980583	700750.415693	5.706399e+06
2	51.472904	-0.102112	4789.311015	701250.415693	5.706399e+06
3	51.472726	-0.094922	4763.139721	701750.415693	5.706399e+06
4	51.472548	-0.087733	4789.311015	702250.415693	5.706399e+06

Now, it would be best if we could add another column to it, with the addresses linked to the given coordinates.

Let's first define a function that will do it, using the Reverse Geocoding function of the Geocoder library. Let's also test it with the coordinates of the city center we calculated prior on:

```
City center coordinates address:

'Roman Amphitheatre Site, Guildhall Yard, Barbican, City of London, Greater London, E
ngland, United Kingdom'
```

Looking good. Let's now apply this function to our dataframe, and retrieve the addresses of each rows:

```
Dataframe shape: (350, 6)
```

	Address	Latitude	Longitude	Distance from Center	X	Y
0	Lambeth day nursery, Stockwell Park Road, Stoc...	51.473259	-0.116492	4993.746089	700250.415693	5.706399e+06
1	Dundas Road, Kennington, London Borough of Lam...	51.473082	-0.109302	4866.980583	700750.415693	5.706399e+06
2	2-4, Welby Street, Paulet Road Estate, Kenning...	51.472904	-0.102112	4789.311015	701250.415693	5.706399e+06
3	40, Valmar Road, Camberwell, London Borough of...	51.472726	-0.094922	4763.139721	701750.415693	5.706399e+06
4	4-8, Ribbon Dance Mews, Camberwell, London Bor...	51.472548	-0.087733	4789.311015	702250.415693	5.706399e+06

The results are satisfying. We can now skip to the next step of our data handling.

Leveraging the Foursquare API

Using the Foursquare API, we will retrieve more informations regarding our neighborhoods.

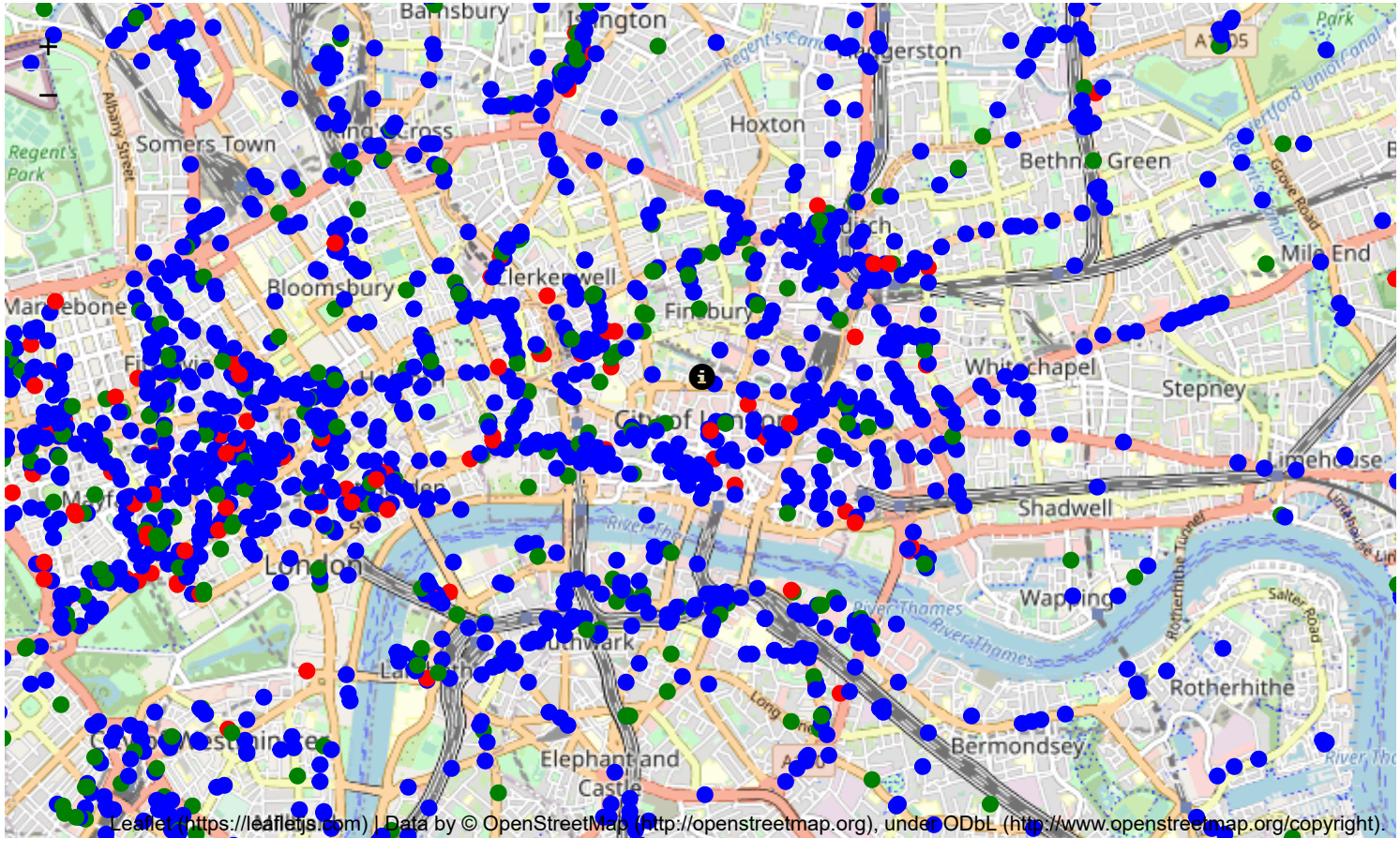
We're interested in venues in 'food' category, but only those that are proper restaurants. Coffe shops, bakeries, fast-foods etc. are not direct competitors and shouldn't be taken into account.

We will specifically look for restaurants, and moreover make a distinction between **Italian**, **French**, and others.

Let's obtain the venues for our areas, using either the Foursquare API, or, if we had already done so and persisted the results in a Pickle file, by loading it:

```
Restaurant data loaded.
```

Let's now see all the collected restaurants in our area of interest on map. We will highligh **Italian** restaurants in green, and **French** ones in red:



Methodology

In this project, we will first determine which type of restaurants are the most prevalent: **Italian**, or **French**.

Then, for the less prevalent of the two, we will find areas around London City that have a low restaurant density, and with particularly low number of this type of restaurant. We will limit our analysis to a radius of 5km.

The first step was to collect and process our data, which we did. We now have usable areas, alongside their restaurants, and restaurants types.

The second step will consist in finding the density. We will use **heatmaps** to locate the best areas, and focus our attention to these.

In the last step, we will focus on the best areas, and create **clusters** that meet the requierements. We will take locations with **no more than two restaurants in a radius of 250 meters**, without **Italian (*french*)** restaurants in a radius of 400m, and no **French (*italian*)** restaurants in a radius of 250m.

We will present those areas on maps, and create clusters using **k-means clustering** to identify the global zone that should be considered.

Analysis

Let's perform some basic explanatory data analysis and derive some additional info from our raw data. First let's count the **number of restaurants in every area candidate**:

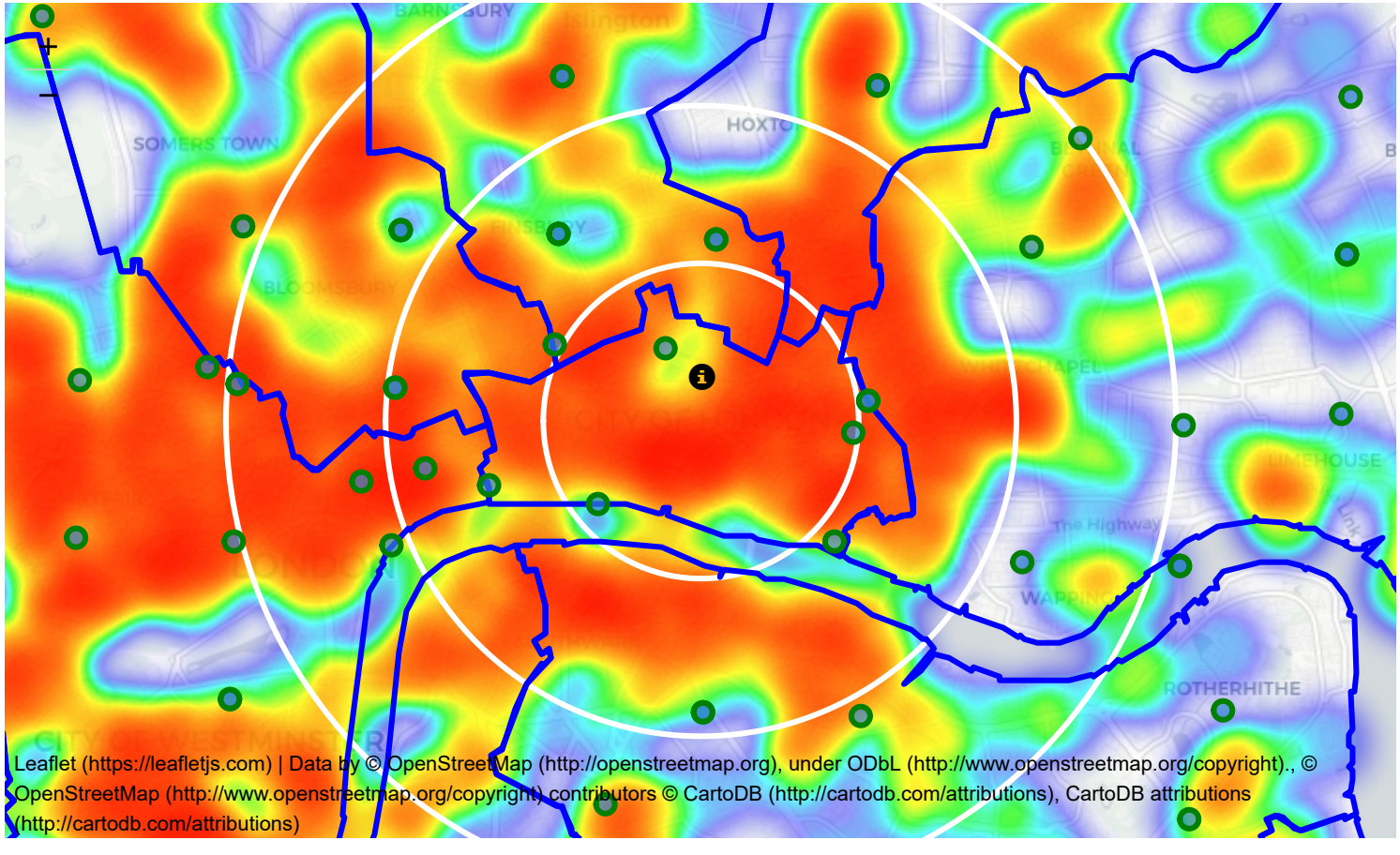
OK, now let's calculate the **distance to nearest Italian and French restaurant from every area candidate center** (not only those within 250m - we want distance to closest one, regardless of how distant it is):

	Address	Latitude	Longitude	Distance from Center	X	Y	Restaurants in area	Distance to Italian restaurant	Distance to French restaurant
0	Lambeth day nursery, Stockwell Park Road, Stoc...	51.473259	-0.116492	4993.746089	700250.415693	5.706399e+06	0	1156.951781	1463.490762
1	Dundas Road, Kennington, London Borough of Lam...	51.473082	-0.109302	4866.980583	700750.415693	5.706399e+06	1	1387.391283	1926.789888
2	2-4, Welby Street, Paulet Road Estate, Kenning...	51.472904	-0.102112	4789.311015	701250.415693	5.706399e+06	1	891.923212	2029.583724
3	40, Valmar Road, Camberwell, London Borough of...	51.472726	-0.094922	4763.139721	701750.415693	5.706399e+06	7	407.675682	2129.980681
4	4-8, Ribbon Dance Mews, Camberwell, London Bor...	51.472548	-0.087733	4789.311015	702250.415693	5.706399e+06	9	192.020588	2335.471111
5	39, Shenley Road, Peckham, London Borough of S...	51.472369	-0.080543	4866.980583	702750.415693	5.706399e+06	4	638.392004	2621.456259
6	Harris Academy at Peckham, 112, Peckham Road, ...	51.472189	-0.073354	4993.746089	703250.415693	5.706399e+06	1	1130.582503	2964.732792
7	Hartington Road, Vauxhall, London Borough of L...	51.477413	-0.127033	4879.805324	699500.415693	5.706832e+06	1	628.490442	603.914257
8	20, Albert Square, Kennington, London Borough ...	51.477236	-0.119842	4670.385423	700000.415693	5.706832e+06	1	657.670108	1088.772929
9	Tesco Brixton Road, Caldwell Street, Kenningto...	51.477059	-0.112651	4506.939094	700500.415693	5.706832e+06	2	977.733564	1583.079452

On average **Italian** restaurant can be found within **~900m**, and a **French** within **~430m** from every area center candidate. Considering on how close it is, we need to focus on our areas more.

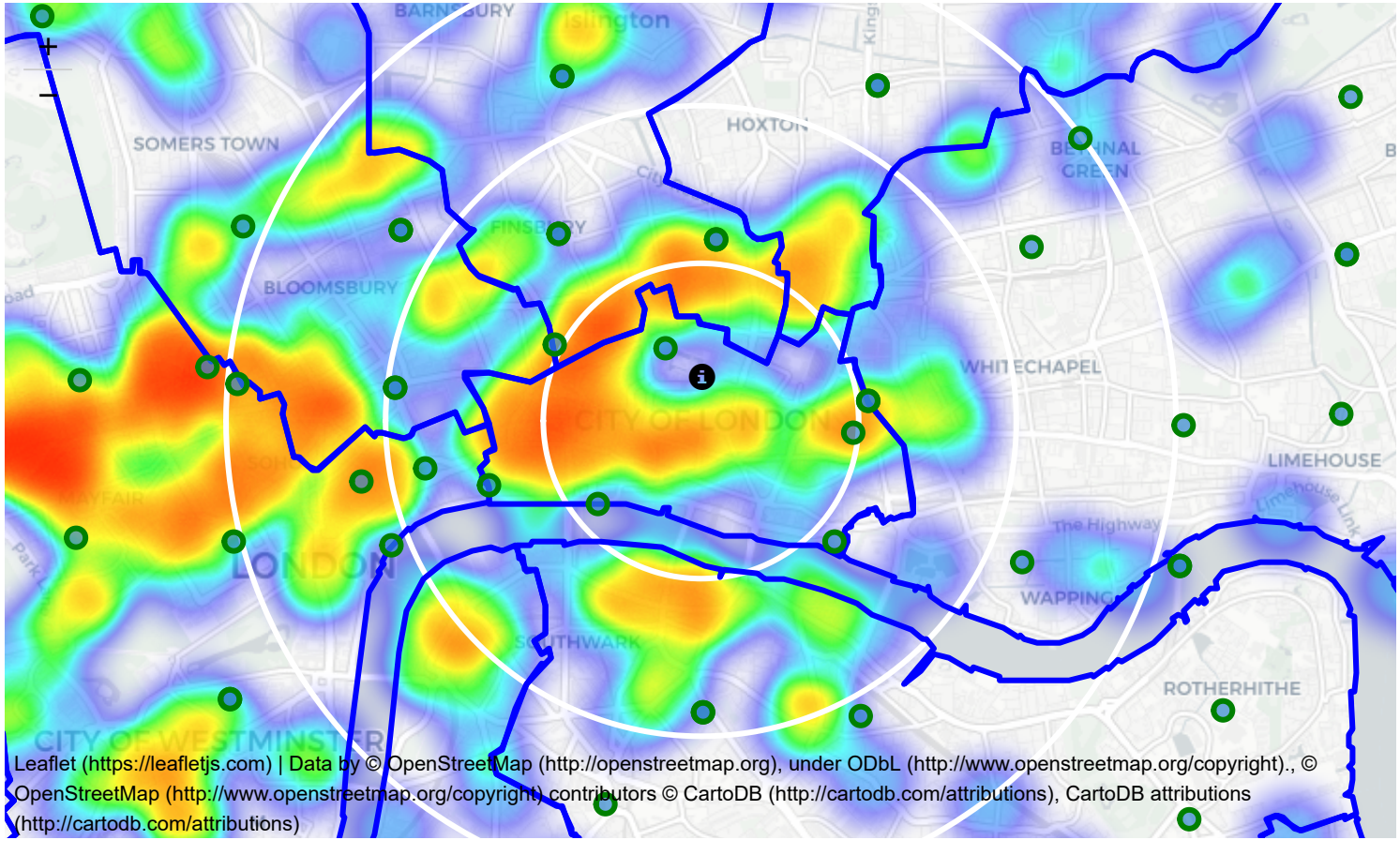
Let us note that it look like **French** restaurants are almost two times less prevalent that **Italian** ones. Let's see if we can confirm that.

Let's create a **heatmap** showing the density of restaurants, and let's use our GeoJSON file to plot the boundaries of each boroughs. We will also add circles around 1km, 2km, and 3km from the center:



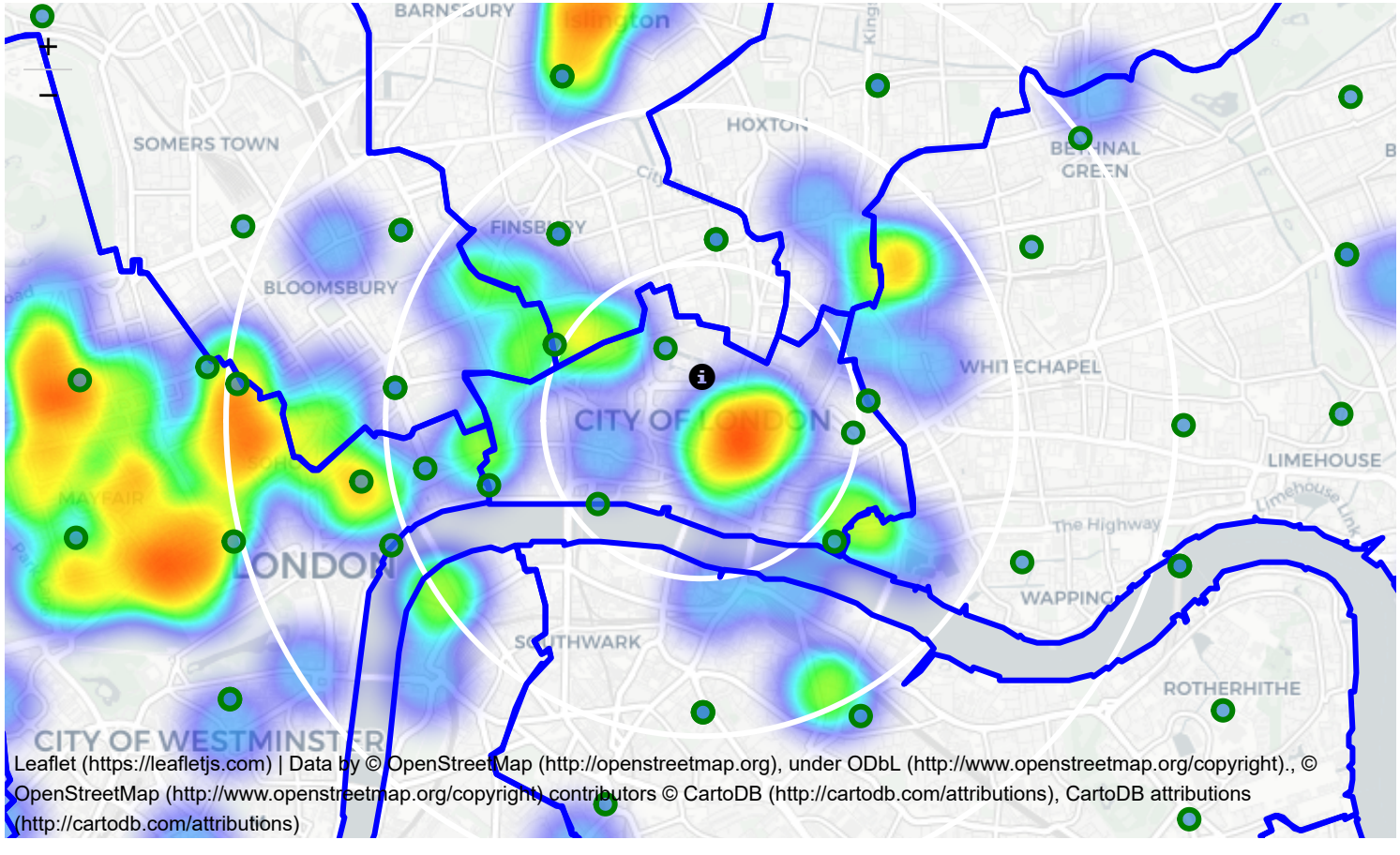
Looks like a most restaurants are located on the western part of the city.

Let's create another heatmap map showing only the **Italian** restaurants:



Italians restaurant are pretty present, and follow the same areas density.

Let's have a look at the **French** ones:

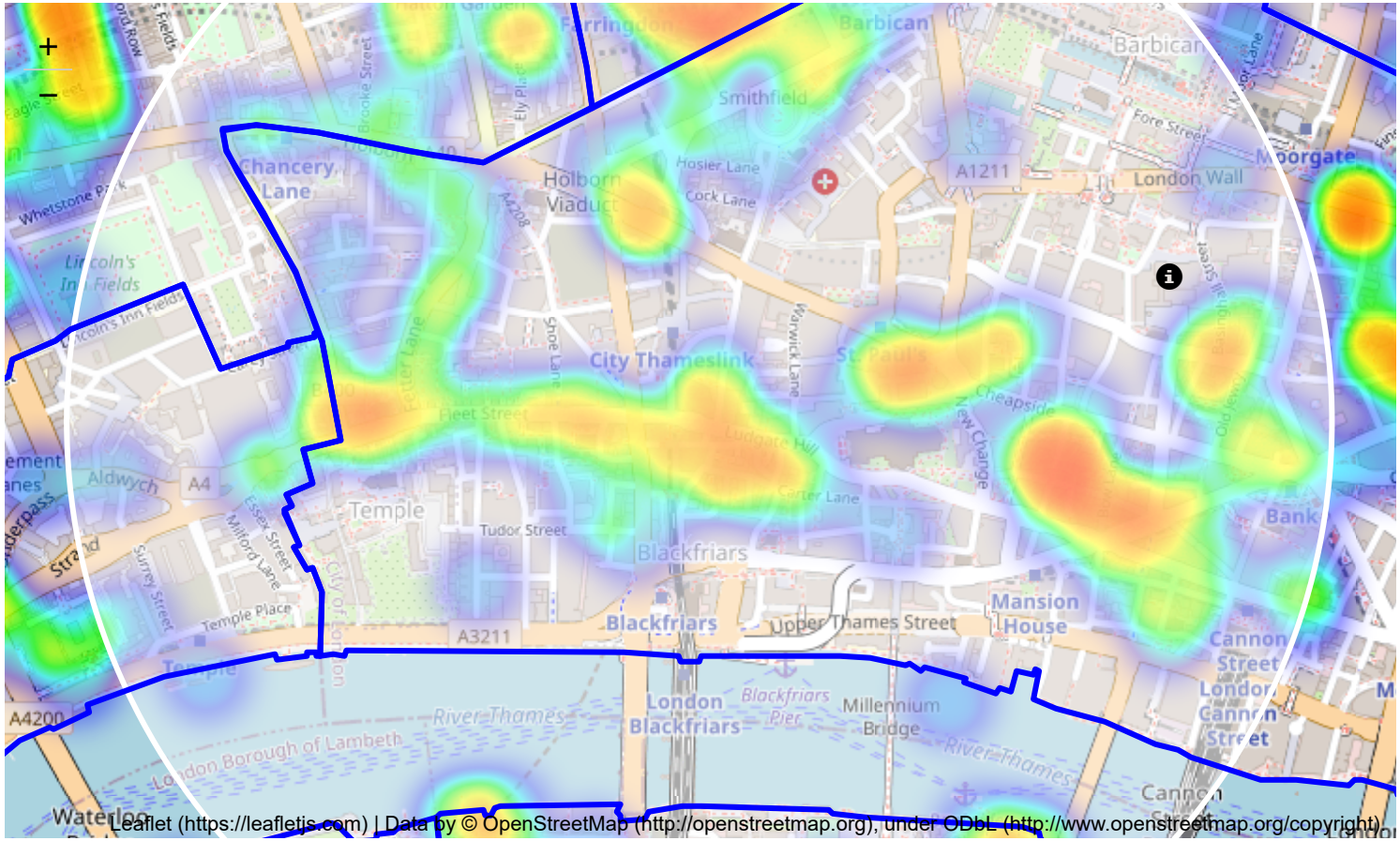


That is already much better. We can clearly see that **French** restaurants are less prevalents than the others, even in high density areas like the western part of the city.

It confirm what we had found before. From now on, we will focus on **French** restaurants.

We can see that most of them are located in Westminster, and that areas around the neighborhoods of Barbican, Blackfriars, and Temple, lack **French** restaurant. They don't lack restaurants in general though.

Let's focus on this area. First, we will define a new, more narrow region at the center of these three neighborhoods. Let's have a look at its restaurant density:



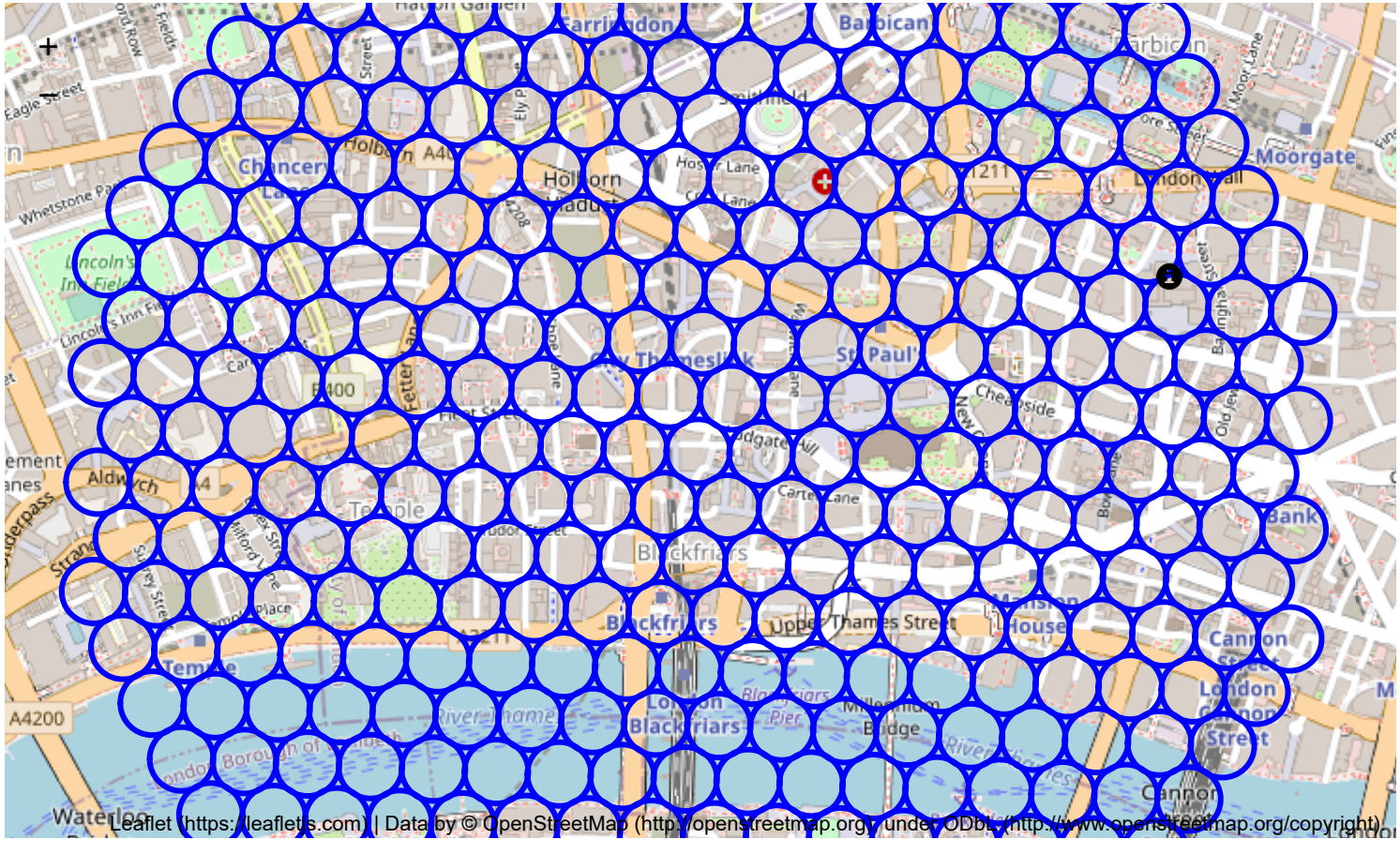
Perfect. It covers nicely our area of interest.

Let's also create new, more dense grid of location candidates. We will make our candidates 100m appart, on a 1x1km hexagone:

```
362 candidate neighborhood centers generated.
```

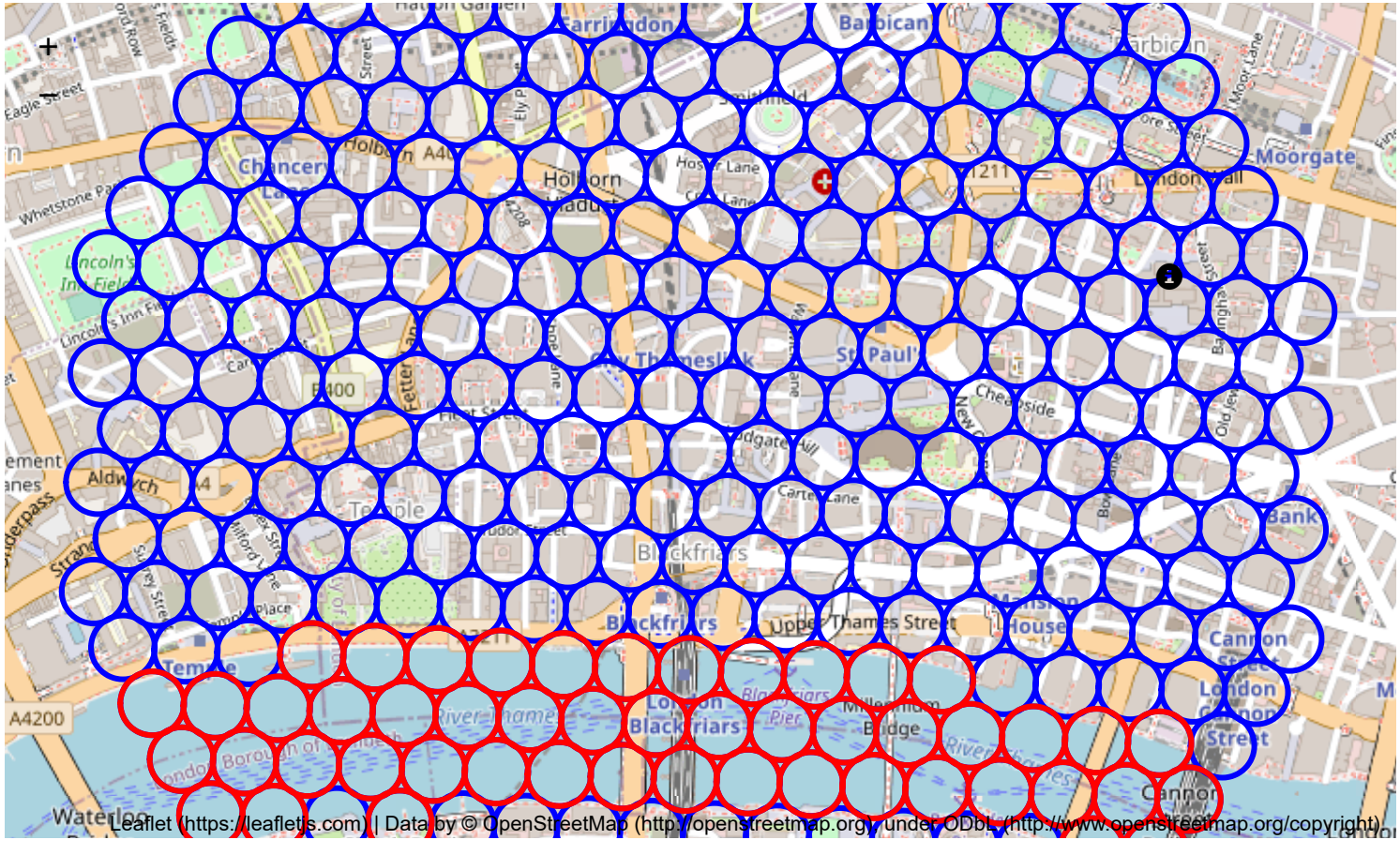
	Latitude	Longitude	X	Y
0	51.506024	-0.109187	700613.6619	5.710062e+06
1	51.505989	-0.107748	700713.6619	5.710062e+06
2	51.505953	-0.106309	700813.6619	5.710062e+06
3	51.505918	-0.104870	700913.6619	5.710062e+06
4	51.505882	-0.103431	701013.6619	5.710062e+06

Let's see that grid on a map:



Once again, the River Thames is a problem.

Let's mimict what we did and remove bad areas:



Perfect, we can now save it all into a dataframe

```
Dataframe shape: (312, 4)
```

	Latitude	Longitude	X	Y
0	51.506024	-0.109187	700613.6619	5.710062e+06
1	51.505989	-0.107748	700713.6619	5.710062e+06
2	51.505953	-0.106309	700813.6619	5.710062e+06
3	51.505918	-0.104870	700913.6619	5.710062e+06
4	51.505882	-0.103431	701013.6619	5.710062e+06

Once again, let's add the corresponding addresses, using the same procedure as before:

```
(312, 5)
```

	Address	Latitude	Longitude	X	Y
0	30, Aquinas Street, South Bank, Lambeth, Londo...	51.506024	-0.109187	700613.6619	5.710062e+06
1	The London Nautical School, Stamford Street, B...	51.505989	-0.107748	700713.6619	5.710062e+06
2	1, Paris Garden, Southwark, London Borough of ...	51.505953	-0.106309	700813.6619	5.710062e+06
3	Colombo Centre, 34-68, Colombo Street, Banksid...	51.505918	-0.104870	700913.6619	5.710062e+06
4	Quadrant House, 15, Burrell Street, Bankside, ...	51.505882	-0.103431	701013.6619	5.710062e+06

Now let's calculate the **number of restaurants in vicinity**, for a radius of 100m and **distance to closest Italian / French restaurant**:

	Address	Latitude	Longitude	X	Y	Restaurants Nearby	Distance to Italian	Distance to French
0	30, Aquinas Street, South Bank, Lambeth, Londo...	51.506024	-0.109187	700613.6619	5.710062e+06	2	243.382059	411.857297
1	The London Nautical School, Stamford Street, B...	51.505989	-0.107748	700713.6619	5.710062e+06	0	206.675344	511.455302
2	1, Paris Garden, Southwark, London Borough of ...	51.505953	-0.106309	700813.6619	5.710062e+06	0	214.929219	611.184604
3	Colombo Centre, 34-68, Colombo Street, Banksid...	51.505918	-0.104870	700913.6619	5.710062e+06	4	263.959165	710.989953
4	Quadrant House, 15, Burrell Street, Bankside, ...	51.505882	-0.103431	701013.6619	5.710062e+06	4	225.864839	810.843269
5	Hilton London Bankside, 2-8, Great Suffolk Str...	51.505847	-0.101992	701113.6619	5.710062e+06	2	135.320996	757.697147
6	93, Southwark Street, Bankside, Southwark, Lon...	51.505811	-0.100553	701213.6619	5.710062e+06	8	74.890712	657.715585
7	Bankside 2, Southwark Street, Bankside, Southw...	51.505775	-0.099114	701313.6619	5.710062e+06	8	21.751122	557.740634
8	Bankside 3, Great Guildford Street, Bankside, ...	51.505740	-0.097675	701413.6619	5.710062e+06	5	15.062640	457.776625
9	95E, Upper Ground, South Bank, Lambeth, London...	51.506855	-0.111297	700463.6619	5.710148e+06	1	306.048930	289.493878

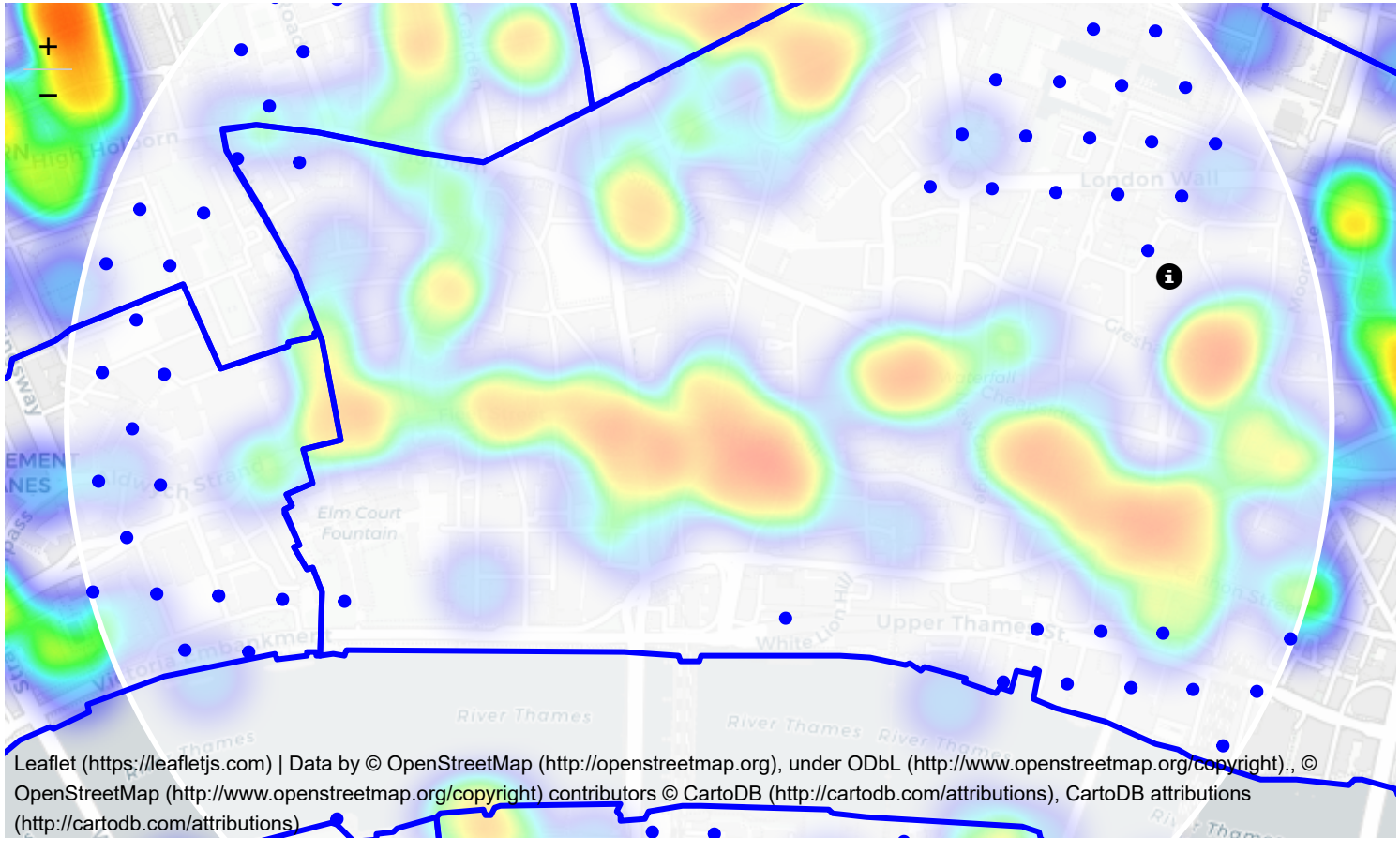
We will now filter on the locations that interest us, meaning:

- No more than two restaurants in a radius of 100m;
- No **Italian** restaurant in a radius of 250m;
- No **French** restaurant in a radius of 400m.

Let's have a look at the resulting places, and plot them on a map:

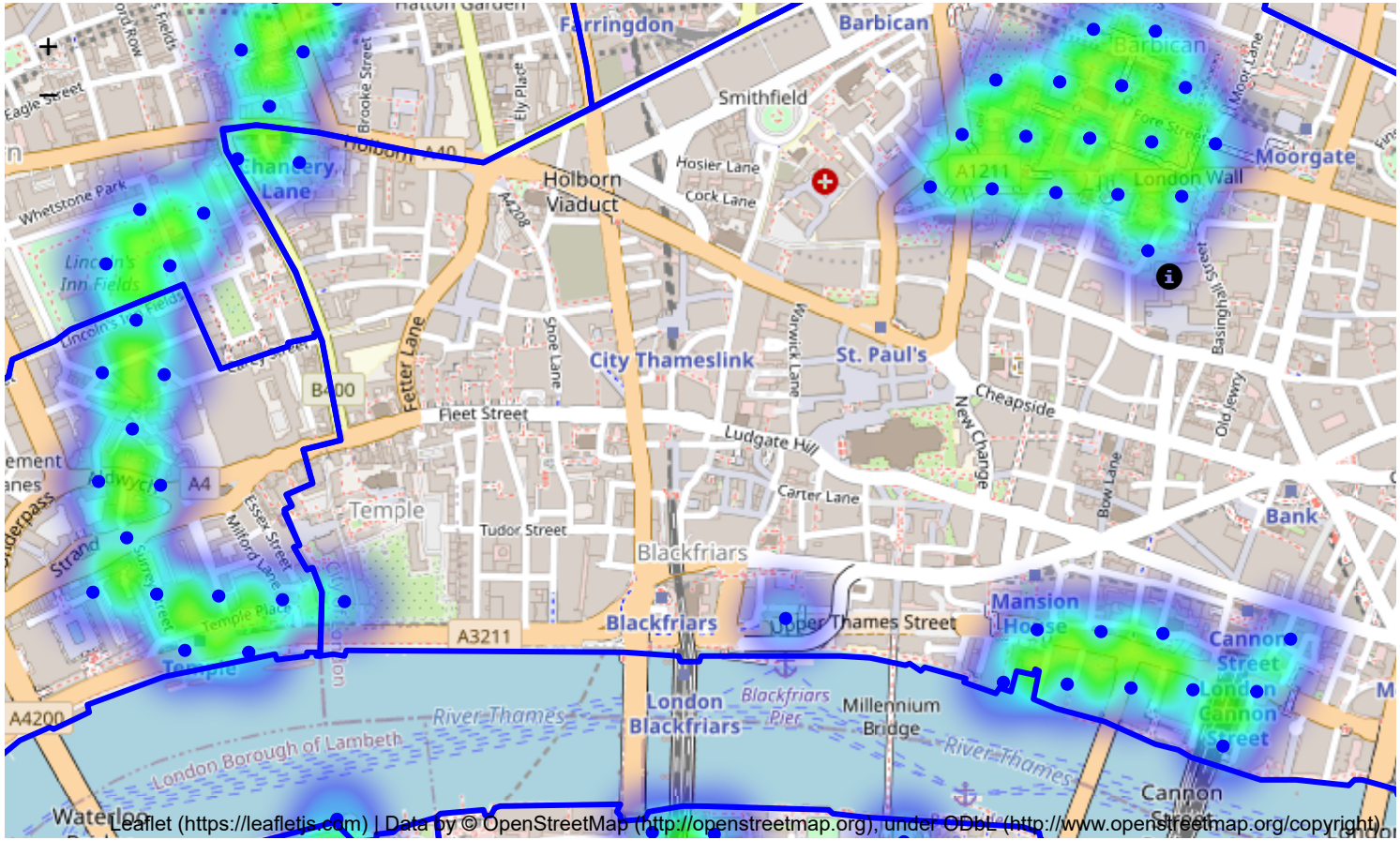
```
Locations with no more than two restaurants nearby: 196
Locations with no Italian restaurants within 200m: 78
Locations with no French restaurants within 400m: 49
Locations with both conditions met: 70
Best places found shape: (70, 8)
```

	Address	Latitude	Longitude	X	Y	Restaurants Nearby	Distance to Italian	Distance to French
0	30, Aquinas Street, South Bank, Lambeth, Londo...	51.506024	-0.109187	700613.6619	5.710062e+06	2	243.382059	411.857297
1	The London Nautical School, Stamford Street, B...	51.505989	-0.107748	700713.6619	5.710062e+06	0	206.675344	511.455302
2	1, Paris Garden, Southwark, London Borough of ...	51.505953	-0.106309	700813.6619	5.710062e+06	0	214.929219	611.184604
9	95E, Upper Ground, South Bank, Lambeth, London...	51.506855	-0.111297	700463.6619	5.710148e+06	1	306.048930	289.493878
10	19, Coin Street, South Bank, Lambeth, London B...	51.506820	-0.109858	700563.6619	5.710148e+06	2	217.130879	381.794184
14	240 Blackfriars Road, 240, Blackfriars Road, B...	51.506678	-0.104101	700963.6619	5.710148e+06	2	247.113951	770.459067
21	The London Studios (ITV), Upper Ground, South ...	51.507651	-0.111967	700413.6619	5.710235e+06	0	334.031345	299.911772
22	Gabriel's Wharf, South Bank, Lambeth, London B...	51.507615	-0.110528	700513.6619	5.710235e+06	0	234.641879	376.705674
27	Pulse - Now Closed, Invicta Plaza, Bankside, S...	51.507438	-0.103333	701013.6619	5.710235e+06	0	235.580149	732.610365
34	South Bank, Lambeth, London Borough of Lambeth...	51.508411	-0.111198	700463.6619	5.710321e+06	0	288.041323	397.551922


+
−
London Wall
Elm Court
Fountain
Upper Thames St
River Thames
River Thames
River Thames
Leaflet (https://leafletjs.com) | Data by © OpenStreetMap (http://openstreetmap.org), under ODbL (http://www.openstreetmap.org/copyright)., © OpenStreetMap (http://www.openstreetmap.org/copyright) contributors © CartoDB (http://cartodb.com/attributions), CartoDB attributions (http://cartodb.com/attributions)

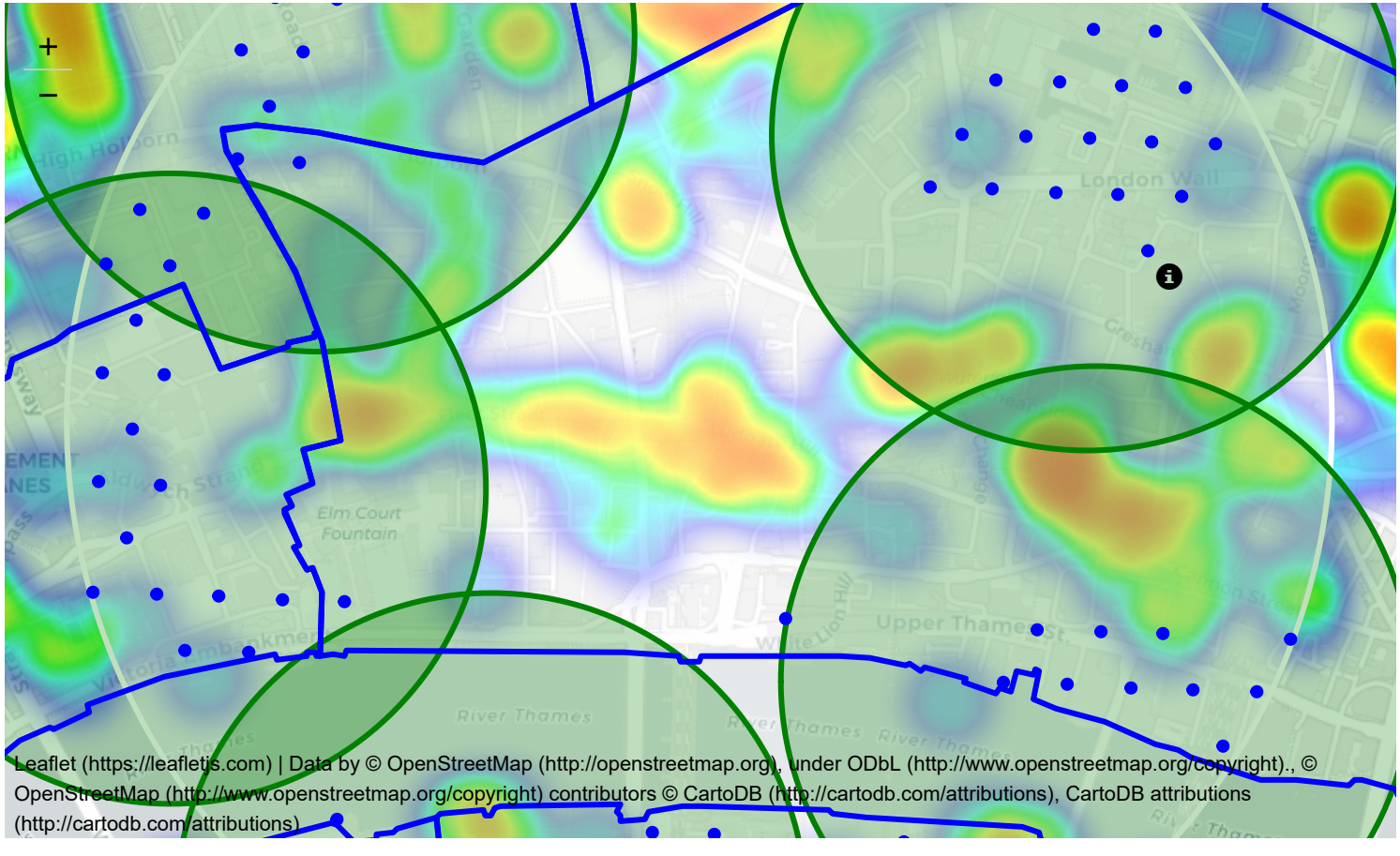
Looking good. We now have a bunch of locations fairly close to the center of the city, and all of them respects our criterias.

Let's see that in the form of a **heatmap**:



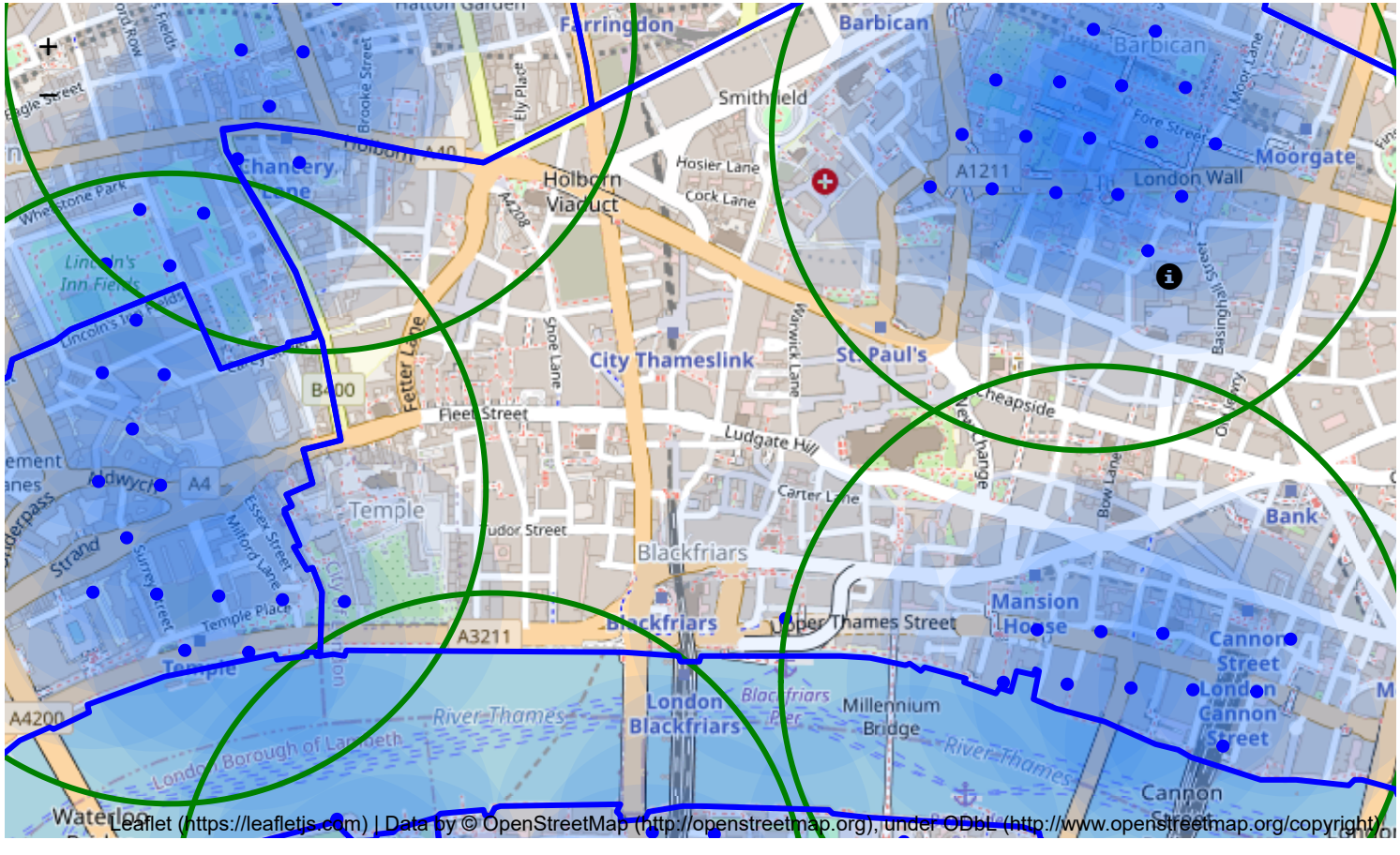
We now have a clear indication of zones with low number of restaurants, low number of **Italiant** restaurant nearby, and no **French** ones close.

Let us now **cluster** those locations to create **centers of zones containing good locations**. Those zones, their centers and addresses will be the final result of our analysis.



Our clusters groups most of the candidates locations, and their centers are placed in the middle of the **hot** zones.

Let's see those zones on a city map without heatmap, using shaded areas to indicate our clusters:



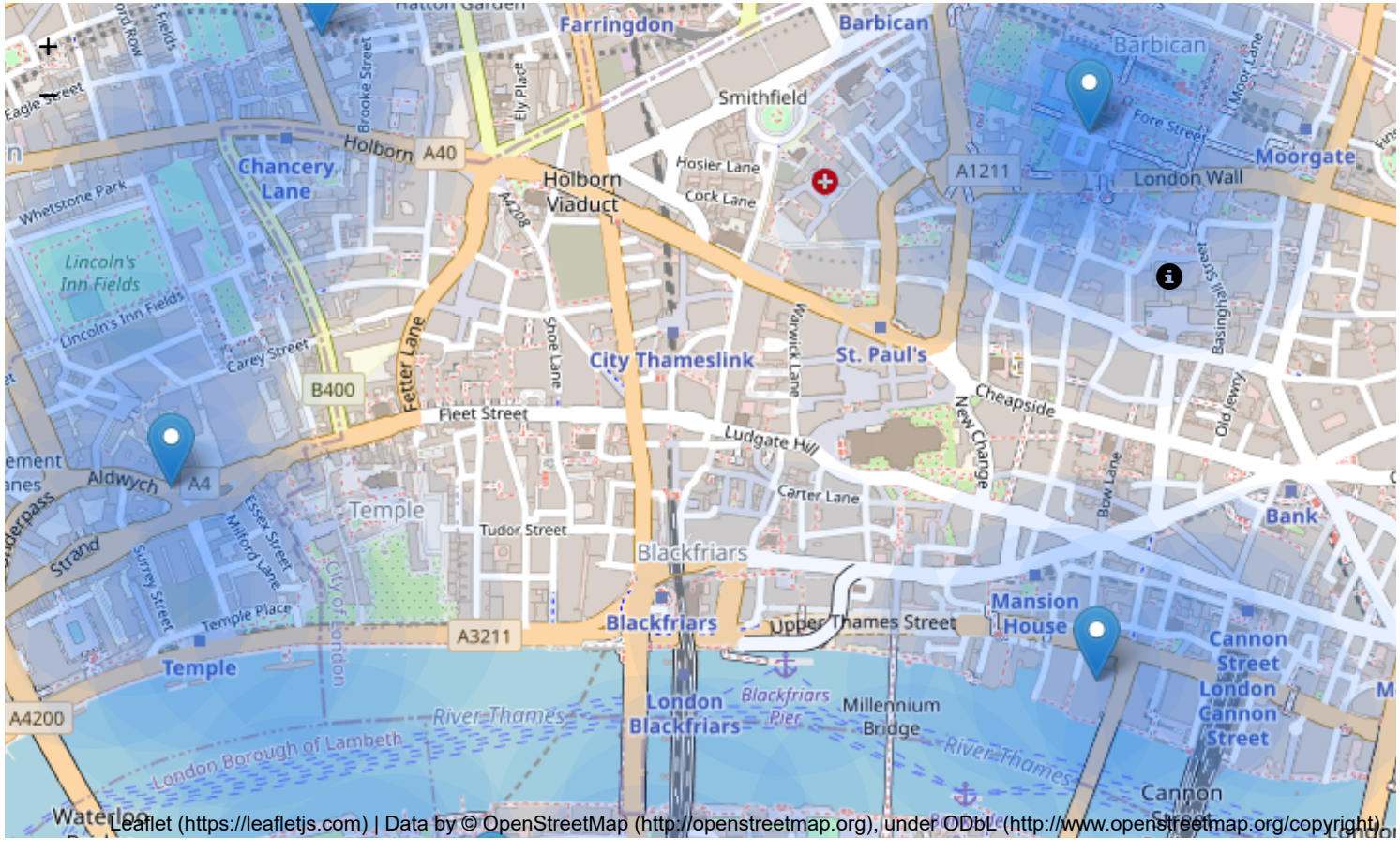
Now it would be best to find the addresses of these clusters. Using the same process as earlier before, let's reverse geocode them:

	Address	Latitude	Longitude	Distance from Center (km)	X	Y
0	3, Brooke's Court, Gray's Inn, Holborn, London...	51.519565	-0.111601	1.420752	700386.738823	5.711561e+06
1	Monkwell Square, Barbican, City of London, Gre...	51.518159	-0.094044	0.322339	701610.720723	5.711452e+06
2	Hatfield House, 52, Stamford Street, Southwark...	51.507155	-0.107674	1.420003	700713.661900	5.710192e+06
3	Vintners Place, Upper Thames Street, Blackfria...	51.510383	-0.093837	0.580570	701659.495233	5.710589e+06
4	Clement House, 97–99, Aldwych, St Clement Dane...	51.513129	-0.114999	1.604640	700179.286900	5.710836e+06

We have created 5 addresses that locate clusters of **hot** zones. Theses zones represent locations with:

- A low number of restaurants;
- Few **Italian** restaurants;
- No **French** restaurants.

They are perfect to establish a new **French** restaurant. Nevertheless, although zones are shown on map with a radius of 250 meters (in green), their shape is quite irregular and their centers/addresses should be considered only as a starting point for exploring area neighborhoods in search for potential restaurant locations.



Results and Discussion

London is a big city. So big that we first had to reduce our zone of research to the City of London borough. And while the zone is already pretty covered in restaurants (more thant 2.000 according to Foursquare, for a 10x10km area), we can still see that the **French** cooking isn't the most represented.

While we can find a high density of **French** restaurants in Westminster, it clearly is not the case around the center. Considering that the center is densely populated by other kitchen types, it is only natural for us to to analysis this area in more details. And that is what we have done.

After directing our attention to this more narrow area of interest, which was 1x1km around Barbican, Blackfriars, and Temple, created a dense grid of location candidates (spaced 100m appart, excluding the River Thames); those locations were then filtered so that those with more than two restaurants in radius of 250m, an **Italian** restaurant close than 250m, or a **French** one close than 400m, were excluded.

We then clustered these candidates into 5 zones of interest, and found their respectives addresses.

It is important to note that this analysis is purely based on numbers. There might be several good reasons for which theses places lacks **French** restaurants, but without analysing the situation *by foot*, we can only suggest that they are, mathematically, the best places to establish a new one.

Conclusions

The goal of this project was to identify the best areas close to the center of London, to establish either an **Italian**, or a **French** restaurant. By calculating the restaurant densities, we were able to advice a **French** one, as they are less prevalent, suggest several locations that were lacking in density.

We used clusters in order to create major zones of interest, as it is more relevant.

Obviously, the final decision will be made by the entrepreneur, and he should also take into account other relevant information that we couldn't analyse, such as:

- Proximity to offices;
- Proximity to parks and / or water;
- Noise levels;
- Availability;
- Rent prices;
- ...

References

- [1] Wikipedia - Areas of London (https://en.wikipedia.org/wiki/List_of_areas_of_London)
- [2] Wikipedia - London Boroughs (https://en.wikipedia.org/wiki/List_of_London_boroughs)
- [3] GeoJSON of London Boroughs (https://skgrange.github.io/www/data/london_boroughs.json)
- [4] GeoJSON of the River Thames (https://jburnford.carto.com/tables/thames/public)
- [5] Foursquare API (https://developer.foursquare.com/)