Capstone Project - The Battle of the Neighborhoods

Guillermo Velasco

Table of contents

Introduction: Business Problem

Madrid is the capital of Spain and has a population of 3.300.000 citizens. With more than 9.400 restaurants, Madrid is considered a great place to enjoy almost any type of cuisine. The negative part of having such a restaurant abundance is that finding the location for a new restaurant is not an easy task.

In this project I will look at the best possible place to open a pizza place in Madrid. The goal would be to find an area without any or few pizza restaurants in the area combined with a high population density.

To obtain this information, I will be using the knowledge acquired during the data science professional certificate course from IBM. The analysis will provide an understanding of the data and insisghts on where would it be better to locate a new pizza place.

Data

To answer the problem stated above, the following information, including the source, needs to be obtained:

- List of neighbourhoods in Madrid including population and area. Source: http://www-2.munimadrid.es/CSE6/control/seleccionDatos?numSerie=14010100012
- Coordinates for each neighbourhood. Source: Geocoder ibrary for Python.
- Pizza restaurants in each neighbourhood. Source: Foursquare API.

Neighborhouds data

The list of neighbourhoods in Madrid is provided by the city of Madrid in an Excel format. The file includes all neighbourhoods sorted by district and the respective area and population. The Excel file is cleaned and imported to Python.

In [1]:
```python
import pandas as pd
df = pd.read_excel('/Users/guillermo/Python/NeighbourhoodsMadrid.xls', sheet_name='Sheet1')
df
```

Out[1]:

	District	Neighbourhood	Area (Ha)	Population
0	Centro	Palacio	146.99	23593
1	Centro	Embajadores	103.37	47048
2	Centro	Cortes	59.19	10771
3	Centro	Justicia	73.94	18021
4	Centro	Universidad	94.80	33418
...	...	...	...	...
126	Barajas	Alameda de Osuna	197.03	19820
127	Barajas	Aeropuerto	2962.61	1900
128	Barajas	Casco Histórico de Barajas	54.94	7683
129	Barajas	Timón	509.45	12853
130	Barajas	Corralejos	468.25	7754

131 rows × 4 columns

A column with the population density is calculated and added to the data set, with density equals to population divided by area.

In [2]:
```python
df["Density"]=df["Population"]/df["Area (Ha)"]
df
```

Out[2]:

	District	Neighbourhood	Area (Ha)	Population	Density
0	Centro	Palacio	146.99	23593	160.507518
1	Centro	Embajadores	103.37	47048	455.141724
2	Centro	Cortes	59.19	10771	181.973306
3	Centro	Justicia	73.94	18021	243.724642
4	Centro	Universidad	94.80	33418	352.510549
...	...	...	...	...	...
126	Barajas	Alameda de Osuna	197.03	19820	100.593818
127	Barajas	Aeropuerto	2962.61	1900	0.641326
128	Barajas	Casco Histórico de Barajas	54.94	7683	139.843466
129	Barajas	Timón	509.45	12853	25.229169
130	Barajas	Corralejos	468.25	7754	16.559530

131 rows × 5 columns

To obtain the latitude and longitude for each neighbourhood the Geocoder librabry is used

In [3]:
```python
import geocoder

latitude=[]
longitude=[]
for code in df['Neighbourhood']:
    g = geocoder.arcgis('{}, Madrid, Madrid'.format(code))
    while (g.latlng is None):
        g = geocoder.arcgis('{}, Madrid, Madrid'.format(code))
    latlng = g.latlng
    latitude.append(latlng[0])
    longitude.append(latlng[1])
```

In [4]:
```python
df["Latitude"]=latitude
df["Longitude"]=longitude
```

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

Out[5]:

	District	Neighbourhood	Area (Ha)	Population	Density	Latitude	Longitude
0	Centro	Palacio	146.99	23593	160.507518	40.41517	-3.71273
1	Centro	Embajadores	103.37	47048	455.141724	40.40803	-3.70067
2	Centro	Cortes	59.19	10771	181.973306	40.41589	-3.69636
3	Centro	Justicia	73.94	18021	243.724642	40.42479	-3.69308
4	Centro	Universidad	94.80	33418	352.510549	40.42565	-3.70726

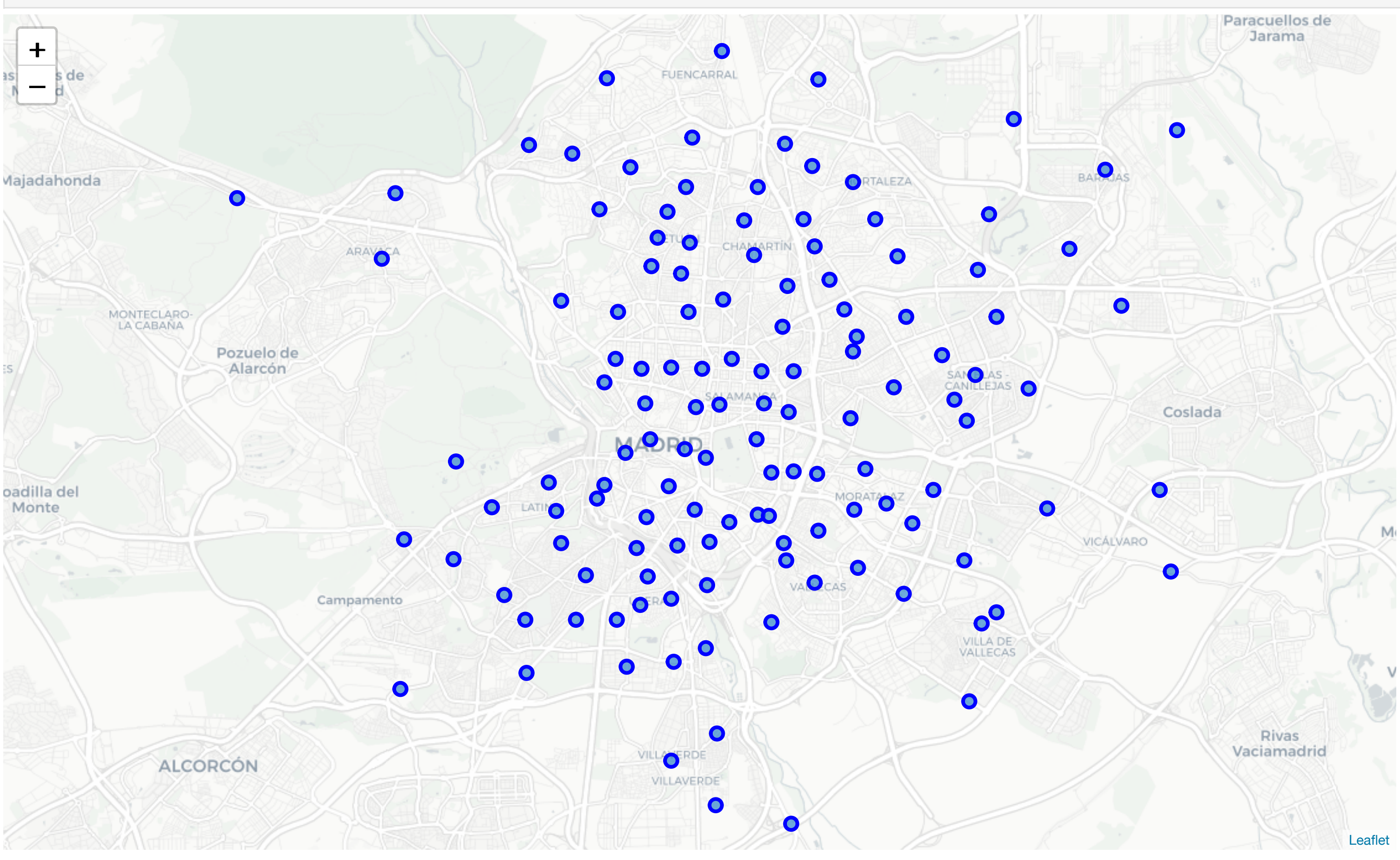
In [6]:
```python
#Using geocoder library to get the latitude and longitude values of Madrid.

g = geocoder.arcgis('Madrid, Madrid')
latlng = g.latlng
latitudeMadrid=latlng[0]
longitudeMadrid=latlng[1]
print('The geograpical coordinates of Madrid are {}, {}.'.format(latitudeMadrid, longitudeMadrid))
```

```
The geograpical coordinates of Madrid are 40.41955000000007, -3.6919599999999377.
```

Now I will plot all neighbourhoods in a map

In [7]:
```python
# create map of Madrid using latitude and longitude values

import folium # map rendering library

map_madrid = folium.Map(location=[latitudeMadrid, longitudeMadrid], zoom_start=12)
folium.TileLayer('cartodbpositron').add_to(map_madrid) #cartodbpositron cartodbdark_matter

# add markers to map
for lat, lng, district, neighbourhood in zip(df['Latitude'], df['Longitude'], df['District'], df['Neighbourhood']):
    label = '{}, {}'.format(neighbourhood, district)
    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,
        parse_html=False).add_to(map_madrid)

map_madrid
```

Out[7]:



Pizza places obtained from Foursquare

By calling the Foursquare API we will obtain all pizza restaurants in the city of Madrid

In [8]:
```python
#Define Foursquare Credentials and Version
CLIENT_ID = 'SE3FSDLHCBUETUXV0P5ANSUJ0HV0NCDCYIEVUJXOY1MVTSVC' # your Foursquare ID
CLIENT_SECRET = 'SGV2YDC3E3G2PN1A1UO32TDRYT5IR2OKKR4OYHGVIJBJUWG2' # your Foursquare Secret
VERSION = '20180605' # Foursquare API version
LIMIT = 100 # A default Foursquare API limit value
```

In [9]:
```python
import json # library to handle JSON files
import requests # library to handle requests
from pandas import json_normalize # tranform JSON file into a pandas dataframe
```

In [10]:
```python
#function to get nearby pizzerias for all the neighborhoods in a radius of 1km
def getNearbyVenues(names, latitudes, longitudes, radius=1000):

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

        # create the API request URL
        url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={},{}&radius={}&limit={}&categoryId={}'.format
            CLIENT_ID,
            CLIENT_SECRET,
            VERSION,
            lat,
            lng,
            radius,
            LIMIT,
            "4bf58dd8d48988d1ca941735") # PIZZA PLACE CATEGORY ID


        # 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)
```

In [11]:
```python
#get nearby pizzeria for all the neighborhoods in Madrid
madrid_pizzerias = getNearbyVenues(names=df['Neighbourhood'],
                                   latitudes=df['Latitude'],
                                   longitudes=df['Longitude']
                                  )
```

In [12]:
```python
madrid_pizzerias
```

Out[12]:

	Neighbourhood	Neighbourhood Latitude	Neighbourhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Palacio	40.41517	-3.71273	Trattoria Malatesta	40.416788	-3.707182	Italian Restaurant
1	Palacio	40.41517	-3.71273	Al Settimo Cielo	40.410509	-3.710321	Pizza Place
2	Palacio	40.41517	-3.71273	Ópera : Pizza	40.417915	-3.708965	Pizza Place
3	Palacio	40.41517	-3.71273	El Horno Azul	40.421598	-3.710031	Pizza Place
4	Palacio	40.41517	-3.71273	López & López	40.409499	-3.704046	Pizza Place
...	...	...	...	...	...	...	...
1449	Alameda de Osuna	40.45818	-3.58953	L'Incontro Trattoria	40.457505	-3.585463	Pizza Place
1450	Alameda de Osuna	40.45818	-3.58953	Pizzamascalzone	40.465177	-3.592820	Pizza Place
1451	Casco Histórico de Barajas	40.47482	-3.57951	Telepizza	40.472902	-3.579102	Pizza Place
1452	Casco Histórico de Barajas	40.47482	-3.57951	Pizzeria La Piazzeta	40.471107	-3.571191	Pizza Place
1453	Corralejos	40.46540	-3.61164	Telepizza	40.469282	-3.616347	Pizza Place

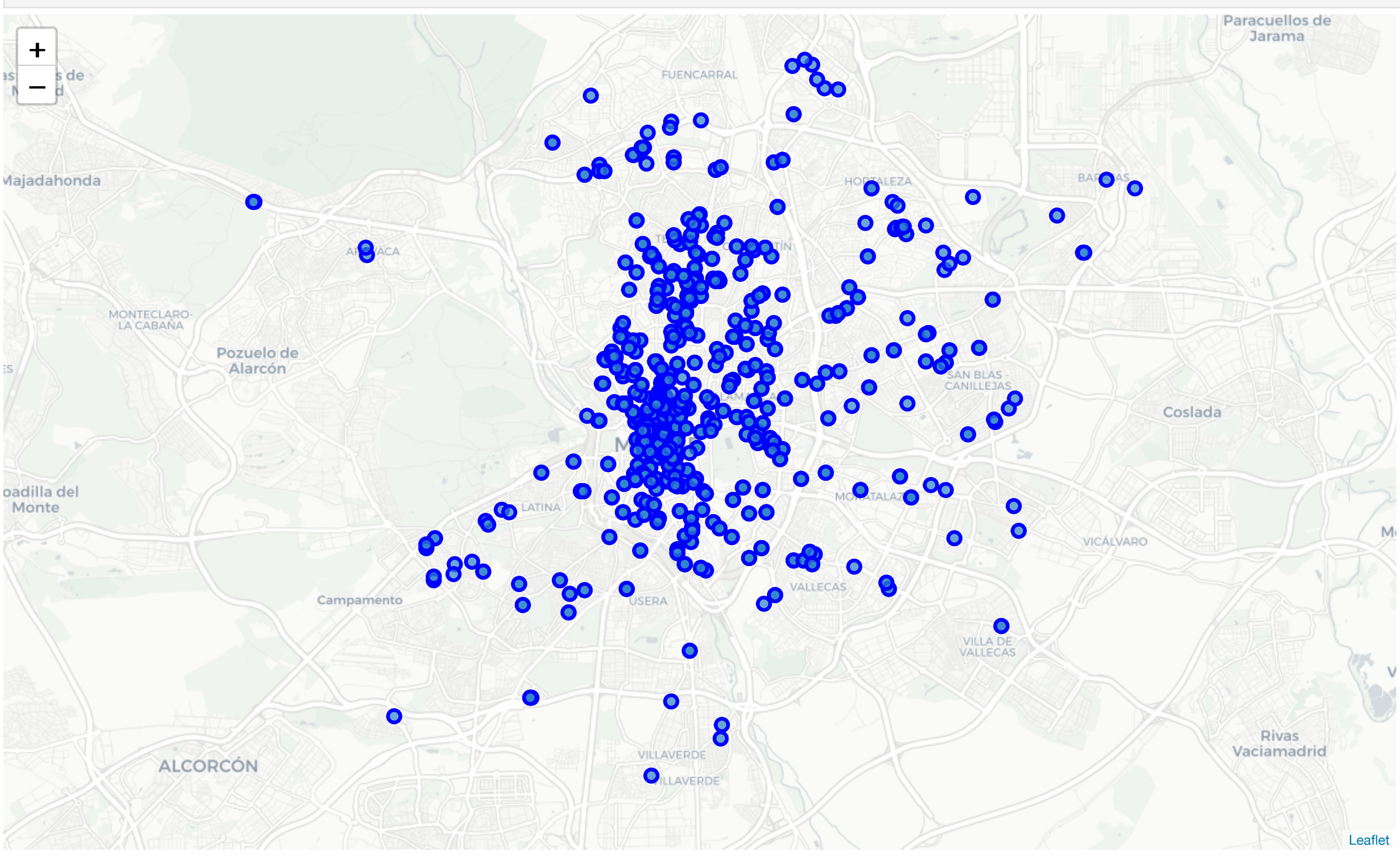
1454 rows × 7 columns

Now that we have all the pizzerias per neighbourhood in a 1km radius lets plot them in a map

In [13]:
```python
from folium import plugins
from folium.plugins import HeatMap
import numpy as np
```

In [15]:
```python
map_madrid = folium.Map(location=[latitudeMadrid, longitudeMadrid], zoom_start=12)
folium.TileLayer('cartodbpositron').add_to(map_madrid) #cartodbpositron cartodbdark_matter
# add markers to map
for lat, lng, venue in zip(madrid_pizzerias['Venue Latitude'], madrid_pizzerias['Venue Longitude'], madrid_pizzerias['Venue']):
    label = '{}'.format(venue)
    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,
        parse_html=False).add_to(map_madrid)

map_madrid
```

Out[15]:



Now we have all the needed data for our analysis.