Housing based on venue preferences and cost of land

Introduction

Buying a house is one of the big financial decision one person does in his life, in this decision is very difficult to be impartial due to the emotional implications persons have at the moment to choose a neighborhood and a house/apartment. To have a more logistic approximation to the decision of where to buy a house, in this study I want to propose an initial approximation that takes into account the venue's preferences of a person, and the average land prices of each neighborhood in a city.

To choose a house the most important characteristics persons consider based on the study of HouseRepay
(https://www.fastrepayhomeloan.com.au/7-factors-to-consider-when-buying-a-house/) are:

- Neighborhood
- Schools and Colleges
- Infrastructure (transportation, connectivity with other neighborhoods)
- Crime (crime index)
- House inspection
- Green open space

In this evaluation, the objective will be to produce personalized results based on preferences of a given person, this means the previously described factors will be grouped to simplify the process of determining the best locations that accomplish the preferences of a person, and the price per square foot on each neighborhood will be included. Additionally, since the information will be at the neighborhood level, specific data like location and house inspection not will be included. The information about venues will be based on personal preferences this means schools/colleges and green spaces will be important only in those cases in which the persons consider this as an important preference. Finally, crime data and infrastructure data not will be included since they are considered out of the scope of this project, but future versions of the model can be included to refine the result of the model.

Data Description

In this case, de case of study to create the model will be the information about the city of Madrid in Spain. To create the model the information will be at the start in five different datasets.

- The first dataset will be the price of the square foot on each neighborhood in Madrid city(https://www.idealista.com/sala-de-prensa/informes-precio-vivienda/venta/madrid-comunidad/madrid-provincia/madrid/).
- The second dataset is the location data of each neighborhood for this geopy was used.
- The third dataset was geographical information to establish boundaries between each neighborhood of Madrid.
- The next dataset is the venue's information of each neighborhood where foursquare API was used.
- Finally, the last dataset will be the user preferences which were used as a recommendation system input to get a recommendation for the given user.

Discussion and Background

found the right places to live are a difficult task and transcendental desition in a person's life, and if at this we sum that cities have become bigger and offer a diversity of activities and places makes this desition even a greater task.

To determine an initial approach to solve the problem I decided to analysis Madrid as a case of study since is a major city, have a multicultural population which causes the city to have a different kind of venues and is possible obtain data of the different neighborhoods of the city.

On each of the following sections will be described the data acquisition and data preparation of the first data sets.

Required Libraries

For this analysis the required libraries are:

geopy - this library will be used to obtain geographical coordinates of each of the neighborhoods.

folium - this is used to draw maps to show the results of each step of the analysis.

geopandas - used to create data frames with polygons and points in order to represent geographical structures into data frames.

```
In [1]: ##!pip install geocoder
        !pip install geopy
        !pip install folium
        !pip install geopandas
```

Methodology

Data Acquisition

In this section is describe how each one of the data sets was obtained. The first data set was the price of the square foot on each neighborhood, for this, I use the data of iedalista.com which is the most popular site to rent and buy properties in Spain. To get this information I scrape the data from:

- (https://www.idealista.com/sala-de-prensa/informes-precio-vivienda/venta/madrid-comunidad/madrid-provincia/madrid/).

But since Idealista has some protection in the form of captcha to avoid his data can be obtained by automated software a few tricks are needed to get the data. Fist once you try to scrape the data using a request, it is possible that the webpage gives you a response indicating that in order to consult the data fulfills a captcha is needed.

```
In [4]: url ='https://www.idealista.com/sala-de-prensa/informes-precio-vivienda/venta/madrid-comunidad/madrid-provincia/madrid/'
        headers = {
            "accept": "text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9",
            "accept-language": "es,es-CO;q=0.9,en;q=0.8,en-US;q=0.7",
            "cache-control": "max-age=0",
            "sec-fetch-dest": "document",
            "sec-fetch-mode": "navigate",
            "sec-fetch-site": "none",
            "sec-fetch-user": "?1",
            "upgrade-insecure-requests": "1",
            "cookie": "_pxhd=99a120b3d70ad9c8e49eab82dacac59a109da6afadb4f4540aaff5eccbc74086:bda91760-b002-11ea-be18-9b17275421ed; cookieDirectiveClosed=true; _pxvid=bda91760-b002-11ea-be18-9b17275421ed;
        }
        response = requests.get(url,headers=headers )
        response.content
```


```
b'<!DOCTYPE html><html lang="es"><head> <meta charset="utf-8"> <meta name="viewport" content="width=device-width, initial-scale=1"> <title>idealista.com : pisos madrid, pisos barcelona, pisos
valencia. inmobiliarias venta, alquiler y obra nueva</title>  <link href="https://fonts.googleapis.com/css?family=Open+Sans:300" rel="stylesheet"> <style>html, body{margin: 0; padding: 0; font
-family: \'Open Sans\', sans-serif; color: #000;}a{color: #c5c5c5; text-decoration: none;}.container{align-items: center; display: flex; flex: 1; justify-content: space-between; flex-directio
n: column; height: 100%;}.container>div{width: 100%; display: flex; justify-content: center;}.container>div>div{display: flex; width: 80%;}.customer-logo-wrapper{padding-top: 2rem; flex-grow:
0; background-color: #fff; visibility: visible;}.customer-logo{border-bottom: 1px solid #000;}.customer-logo>img{padding-bottom: 1rem; max-height: 50px; max-width: auto;}.page-title-wrapper{fl
ex-grow: 2;}.page-title{flex-direction: column-reverse;}.content-wrapper{flex-grow: 5;}.content{flex-direction: column;}.page-footer-wrapper{align-items: center; flex-grow: 0.2; background-col
or: #000; color: #c5c5c5; font-size: 70%;}@media (min-width:768px){html, body{height: 100%;}}</style></head><body> <section class="container"> <div class="customer-logo-wrapper"> <div class="c
ustomer-logo"> <img src=\'https://st1.idealista.com/static/common/img/icons/logo4.png\' alt="Logo"/> </div></div><div class="page-title-wrapper"> <div class="page-title"> <h1></h1> </div></div
><div class="content-wrapper"> <div class="content"> <p>Vaya! parece que estamos recibiendo muchas peticiones tuyas en poco tiempo <br/>Para ver que realmente eres t&uacute; y no un maligno ro
bot, te pedimos que hagas la siguiente validaci&oacute;n:</p><div id="px-captcha"> </div><p> Un saludo,<br/>El equipo de idealista</p><p> Reference ID: #fea92180-b00a-11ea-a687-c31f0b79c8d3 </
p><p>&iquest;No consigues pasar de aqu&iacute;?<br/>Atenci&oacute;n personalizada <strong>902 55 98 58</strong><br/>Coste de una llamada provincial<br/>Lunes a Viernes de 9:00 a 21:00</p><p id
="copyright">Copyright &copy; 2000-<span id="year"></span> idealista</p></div></div><div class="page-footer-wrapper"> <div class="page-footer"> <p> Powered by <a href="https://www.perimeterx.c
om">PerimeterX</a> , Inc. </p></div></div></section> <script>window._pxAppId=\'PXm4C135aU\'; window._pxJsClientSrc=\'/m4C135aU/init.js\'; window._pxFirstPartyEnabled= false ; window._pxVid=
\'\'; window._pxUuid=\'fea92180-b00a-11ea-a687-c31f0b79c8d3\'; window._pxHostUrl=\'/m4C135aU/xhr\'; window._pxreCaptchaTheme=\'dark\'; </script> <script>var s=document.createElement(\'script
\'); s.src=\'/m4C135aU/captcha/captcha.js?a=c&u=fea92180-b00a-11ea-a687-c31f0b79c8d3&v=&m=0\'; var p=document.getElementsByTagName(\'head\')[0]; p.insertBefore(s, null); if ( false ){s.onerror
=function(){s=document.createElement(\'script\'); var suffixIndex=\'/m4C135aU/captcha/captcha.js?a=c&u=fea92180-b00a-11ea-a687-c31f0b79c8d3&v=&m=0\'.indexOf(\'/captcha.js\'); var temperedBlock
Script=\'/m4C135aU/captcha/captcha.js?a=c&u=fea92180-b00a-11ea-a687-c31f0b79c8d3&v=&m=0\'.substring(suffixIndex); s.src=\'//captcha.px-cdn.net/PXm4C135aU\' + temperedBlockScript; p.parentNode.
insertBefore(s, p);};}</script> <script>document.getElementById("year").innerHTML=new Date().getFullYear(); </script>  <script type="text/javascript">var utag_data={"response":{"isError":"1",
"statusCode":"403", "loadBalancer":"captcha", "errorTemplate":"blockingPage" }}</script> <script type="text/javascript">(function(a,b,c,d){a=\'//tags.tiqcdn.com/utag/idealista/es-portal/prod/u
tag.js\'; b=document;c=\'script\';d=b.createElement(c);d.src=a;d.type=\'text/java\'+c;d.async=true; a=b.getElementsByTagName(c)[0];a.parentNode.insertBefore(d,a);})(); </script></body></html>'
```

To solve this simply open the web page in a browser and solve the captcha before sending the request again. If you cannot see the captcha, use CTRL + F5 to force your browser to delete cache and this will cause that the captcha loads correctly.

Once the request returns the correct information I use BeautifulSoup to scrape the data from the result of the request and store it into a data frame.

```
In [6]: soup = BeautifulSoup(response.text, 'html.parser')
        table_data = soup.find_all('table')
        # table_data
        df = pd.read_html(table_data[0].prettify(), flavor='bs4')[0]
        df.head()
```

Out[6]:

	Localización	Precio m2 mayo 2020	Variación mensual	Variación trimestral	Variación anual	Máximo histórico	Variación máximo
0	Madrid	3.782 €/m2	+0,5 %	+1,5 %	-1,0 %	3.822 €/m2 jul 2019	-1,1 %
1	Arganzuela	3.962 €/m2	+1,5 %	+1,8 %	-2,9 %	4.096 €/m2 jul 2019	-3,3 %
2	Barajas	3.144 €/m2	-2,5 %	-2,1 %	-0,8 %	3.663 €/m2 mar 2009	-14,2 %
3	Carabanchel	2.146 €/m2	-0,6 %	-2,8 %	-2,1 %	3.173 €/m2 jun 2007	-32,4 %
4	Centro	5.075 €/m2	+0,3 %	-0,2 %	+1,7 %	5.096 €/m2 ene 2020	-0,4 %

The second data set that was needed, is the geographical location of each neighborhood in Madrid, to get this data I use the names of the neighborhoods in the first data set and with the geopy library, I obtain the geographical coordinates of each one of the neighbors.

In [7]:

```
location_list=[]
for neighborhood in df["Localización"]:
    geolocator = Nominatim(user_agent="madrid_explorer")
    location = geolocator.geocode('Madrid, '+neighborhood)
    latitude = location.latitude
    longitude = location.longitude
    location_list.append([neighborhood, latitude, longitude])
    print('The geograpical coordinate of '+neighborhood+' are {}, {}.'.format(latitude, longitude))

location_list
```

```
The geograpical coordinate of Madrid are 40.4167047, -3.7035825.
The geograpical coordinate of Arganzuela are 40.39806845, -3.6937339526567428.
The geograpical coordinate of Barajas are 40.4733176, -3.5798446.
The geograpical coordinate of Carabanchel are 40.3742112, -3.744676.
The geograpical coordinate of Centro are 40.417652700000005, -3.7079137662915533.
The geograpical coordinate of Chamartín are 40.4589872, -3.6761288.
The geograpical coordinate of Chamberí are 40.43624735, -3.7038303534513837.
The geograpical coordinate of Ciudad Lineal are 40.4484305, -3.650495.
The geograpical coordinate of Fuencarral are 40.4262741, -3.7009067.
The geograpical coordinate of Hortaleza are 40.4725491, -3.6425515.
The geograpical coordinate of Latina are 40.4035317, -3.736152.
The geograpical coordinate of Moncloa are 40.4350196, -3.719236.
The geograpical coordinate of Moratalaz are 40.4059332, -3.6448737.
The geograpical coordinate of Puente de Vallecas are 40.3835532, -3.65453548036571.
The geograpical coordinate of Retiro are 40.4111495, -3.6760566.
The geograpical coordinate of Salamanca are 40.4270451, -3.6806024.
The geograpical coordinate of San Blas are 40.4275001, -3.615954.
The geograpical coordinate of Tetuán are 40.4605781, -3.6982806.
The geograpical coordinate of Usera are 40.383894, -3.7064459.
The geograpical coordinate of Vicálvaro are 40.3965841, -3.5766216.
The geograpical coordinate of Villa de Vallecas are 40.3739576, -3.6121632.
The geograpical coordinate of Villaverde are 40.3456104, -3.6959556.
```

Once we have both data sets I do a process of cleaning and normalize the data by renaming columns, remove no required data, and finally by merging both data sets into a single data frame.

Out[12]:

	neighborhood	price_m2	monthly_variation	quarterly_variation	anual_variation	historical_max	max_variation	latitude	longitude
1	Arganzuela	3.962 €/m2	+1,5 %	+1,8 %	-2,9 %	4.096 €/m2 jul 2019	-3,3 %	40.398068	-3.693734
2	Barajas	3.144 €/m2	-2,5 %	-2,1 %	-0,8 %	3.663 €/m2 mar 2009	-14,2 %	40.473318	-3.579845
3	Carabanchel	2.146 €/m2	-0,6 %	-2,8 %	-2,1 %	3.173 €/m2 jun 2007	-32,4 %	40.374211	-3.744676
4	Centro	5.075 €/m2	+0,3 %	-0,2 %	+1,7 %	5.096 €/m2 ene 2020	-0,4 %	40.417653	-3.707914
5	Chamartin	5.179 €/m2	+0,7 %	+1,4 %	+2,5 %	5.216 €/m2 nov 2018	-0,7 %	40.458987	-3.676129
6	Chamberi	5.432 €/m2	+0,1 %	+1,3 %	+2,8 %	5.432 €/m2 mayo 2020	0,0 %	40.436247	-3.703830
7	Ciudad Lineal	3.021 €/m2	+0,6 %	-0,5 %	-1,7 %	3.578 €/m2 oct 2007	-15,6 %	40.448431	-3.650495
8	Fuencarral	3.573 €/m2	+0,1 %	+1,6 %	+5,0 %	3.726 €/m2 mayo 2008	-4,1 %	40.426274	-3.700907
9	Hortaleza	3.714 €/m2	-1,2 %	-0,6 %	+0,5 %	3.806 €/m2 dic 2007	-2,4 %	40.472549	-3.642552
10	Latina	2.330 €/m2	-1,5 %	+0,8 %	+1,8 %	3.019 €/m2 nov 2007	-22,8 %	40.403532	-3.736152
11	Moncloa	3.898 €/m2	-0,9 %	-1,2 %	-1,3 %	4.012 €/m2 dic 2008	-2,9 %	40.435020	-3.719236
12	Moratalaz	2.499 €/m2	+1,0 %	0,0 %	+0,2 %	2.718 €/m2 sep 2009	-8,1 %	40.405933	-3.644874
13	Puente de Vallecas	1.952 €/m2	0,0 %	+0,3 %	+2,0 %	2.942 €/m2 abr 2008	-33,7 %	40.383553	-3.654535
14	Retiro	4.586 €/m2	+0,6 %	+0,8 %	-1,8 %	4.669 €/m2 mayo 2019	-1,8 %	40.411150	-3.676057
15	Salamanca	5.985 €/m2	+1,6 %	+3,4 %	+2,3 %	5.985 €/m2 mayo 2020	0,0 %	40.427045	-3.680602
16	San Blas	2.497 €/m2	-0,8 %	+0,3 %	-1,8 %	3.603 €/m2 nov 2007	-30,7 %	40.427500	-3.615954
17	Tetuan	3.679 €/m2	-0,9 %	-0,3 %	-1,2 %	3.857 €/m2 dic 2007	-4,6 %	40.460578	-3.698281
18	Usera	2.024 €/m2	-1,7 %	-2,3 %	-0,7 %	3.110 €/m2 nov 2007	-34,9 %	40.383894	-3.706446
19	Vicalvaro	2.350 €/m2	+0,7 %	+0,2 %	+3,7 %	2.656 €/m2 nov 2010	-11,5 %	40.396584	-3.576622
20	Villa de Vallecas	2.437 €/m2	+3,0 %	+3,1 %	+0,7 %	2.955 €/m2 mayo 2008	-17,5 %	40.373958	-3.612163
21	Villaverde	1.728 €/m2	+0,1 %	-0,1 %	0,0 %	2.900 €/m2 feb 2008	-40,4 %	40.345610	-3.695956

Finally, in order to verify that all the data is correct, I draw a map using folium and verify that each marker corresponds to each of the neighbors. In this case, one of the markers was offset (Fuencarral) of the neighbor so the coordinates are overrides with the correct coordinates.

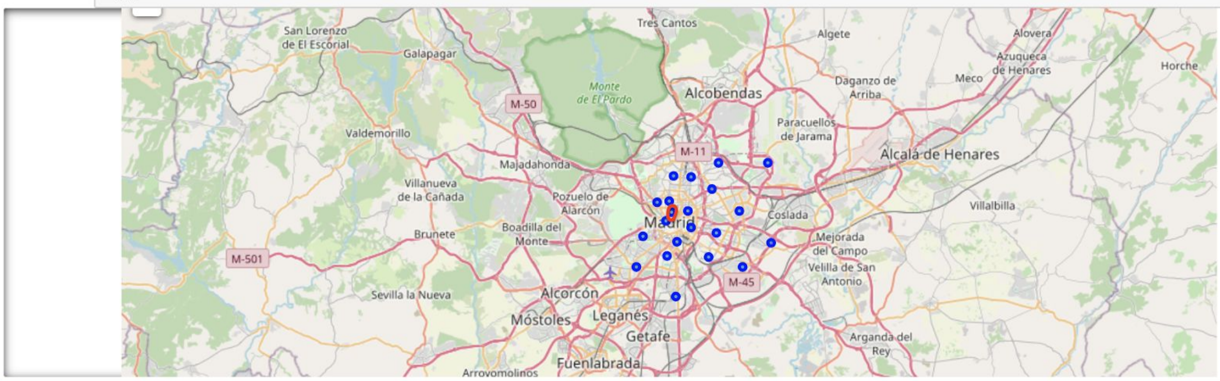
In [15]:
```
#Create map using latitude and longitude values

madrid_data['latitude']
madrid_data['longitude']

map_madrid = folium.Map(width=1000, height=500,location=[madrid_data['latitude'], madrid_data['longitude']], zoom_start=10)

# add markers to map
for lat, lng, neighborhood in zip(df['latitude'], df['longitude'], df['neighborhood']):
    label = neighborhood
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius=3,
        popup=label,
        color='blue',
        fill=True,
        fill_color='#3186cc',
        fill_opacity=0.7,
        parse_html=False).add_to(map_madrid)

map_madrid
```



In [14]:
```
# Fix Fuencarral location
index = int(df[df['neighborhood']=='Fuencarral'].index[0])
df['latitude'][index] = 40.519031
df['longitude'][index] = -3.775905
```

The third data set used was the polygons of the neighborhoods of Madrid, this data was obtained from fantasmagoria.com

- https://fantasmagoria.carto.com/api/v2/sql?filename=distrito_geojson&q=select+*+from+public.distrito_geojson&format=geojson&bounds=&api_key=

The geojson obtained, was loaded as a data frame, drop the unnecessary columns, and merge the data with the main data frame used in the two previous steps.

In [17]:
```
url_geo_madrid='https://fantasmagoria.carto.com/api/v2/sql?filename=distrito_geojson&q=select+*+from+public.distrito_geojson&format=geojson&bounds=&api_key='
response_geo = requests.get(url_geo_madrid)
df_geo = gpd.GeoDataFrame(response_geo.json())
df_geo
```

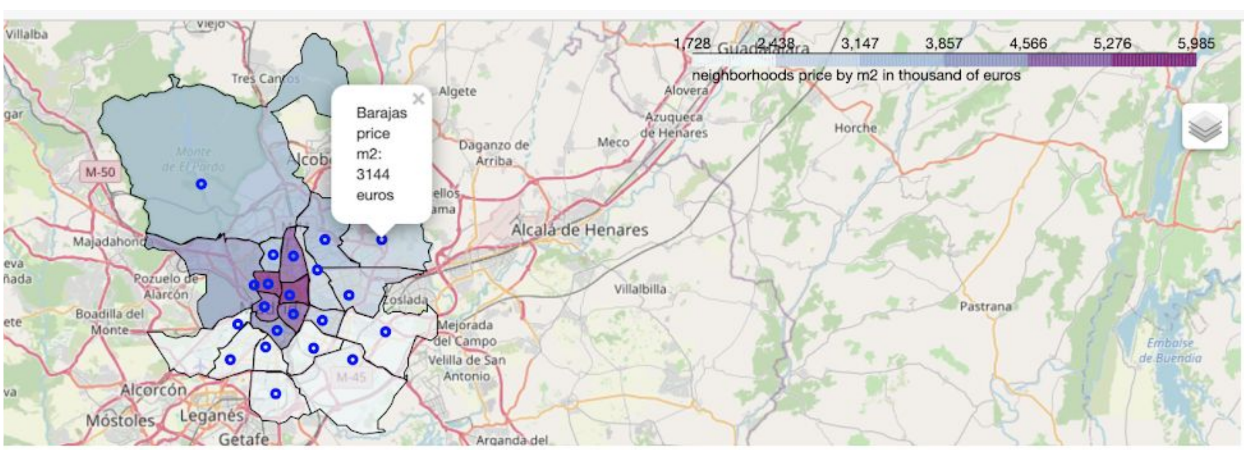
Out[17]:

	type	features
0	FeatureCollection	{'type': 'Feature', 'geometry': {'type': 'Mult...
1	FeatureCollection	{'type': 'Feature', 'geometry': {'type': 'Mult...
2	FeatureCollection	{'type': 'Feature', 'geometry': {'type': 'Mult...
3	FeatureCollection	{'type': 'Feature', 'geometry': {'type': 'Mult...
4	FeatureCollection	{'type': 'Feature', 'geometry': {'type': 'Mult...
5	FeatureCollection	{'type': 'Feature', 'geometry': {'type': 'Mult...

Out[25]:

	neighborhood	price_m2	monthly_variation	quarterly_variation	anual_variation	historical_max	max_variation	latitude	longitude	geometry.coordinates	properties.codigo
0	Arganzuela	3962	+1,5 %	+1,8 %	-2,9 %	4.096 €/m2 jul 2019	-3,3 %	40.398068	-3.693734	[[[[-3.703413, 40.405096], [-3.703165, 40.4050...	28079602
1	Barajas	3144	-2,5 %	-2,1 %	-0,8 %	3.663 €/m2 mar 2009	-14,2 %	40.473318	-3.579845	[[[[-3.561544, 40.510729], [-3.56154, 40.51071...	28079621
2	Carabanchel	2146	-0,6 %	-2,8 %	-2,1 %	3.173 €/m2 jun 2007	-32,4 %	40.374211	-3.744676	[[[[-3.724663, 40.404549], [-3.724586, 40.4045...	28079611
3	Centro	5075	+0,3 %	-0,2 %	+1,7 %	5.096 €/m2 ene 2020	-0,4 %	40.417653	-3.707914	[[[[-3.712148, 40.430235], [-3.71205, 40.43022...	28079601
4	Chamartin	5179	+0,7 %	+1,4 %	+2,5 %	5.216 €/m2 nov 2018	-0,7 %	40.458987	-3.676129	[[[[-3.673517, 40.482855], [-3.673633, 40.4822...	28079605
5	Chamberi	5432	+0,1 %	+1,3 %	+2,8 %	5.432 €/m2 mayo 2020	0,0 %	40.436247	-3.703830	[[[[-3.698789, 40.446603], [-3.698725, 40.4465...	28079607
6	Ciudad Lineal	3021	+0,6 %	-0,5 %	-1,7 %	3.578 €/m2 oct 2007	-15,6 %	40.448431	-3.650495	[[[[-3.668803, 40.484158], [-3.668583, 40.4841...	28079615
7	Fuencarral	3573	+0,1 %	+1,6 %	+5,0 %	3.726 €/m2 mayo 2008	-4,1 %	40.519031	-3.775905	[[[[-3.645824, 40.639182], [-3.64445, 40.63802...	28079608
8	Hortaleza	3714	-1,2 %	-0,6 %	+0,5 %	3.806 €/m2 dic 2007	-2,4 %	40.472549	-3.642552	[[[[-3.644942, 40.507951], [-3.644803, 40.5078...	28079616
9	Latina	2330	-1,5 %	+0,8 %	+1,8 %	3.019 €/m2 nov 2007	-22,8 %	40.403532	-3.736152	[[[[-3.721954, 40.409653], [-3.721702, 40.4084...	28079610
10	Moncloa	3898	-0,9 %	-1,2 %	-1,3 %	4.012 €/m2 dic 2008	-2,9 %	40.435020	-3.719236	[[[[-3.800239, 40.469298], [-3.800069, 40.4691...	28079609
11	Moratalaz	2499	+1,0 %	0,0 %	+0,2 %	2.718 €/m2 sep 2009	-8,1 %	40.405933	-3.644874	[[[[-3.642962, 40.414523], [-3.64, 40.414452],...	28079614
12	Puente de Vallecas	1952	0,0 %	+0,3 %	+2,0 %	2.942 €/m2 abr 2008	-33,7 %	40.383553	-3.654535	[[[[-3.677395, 40.36066], [-3.678431, 40.36063...	28079613
13	Retiro	4586	+0,6 %	+0,8 %	-1,8 %	4.669 €/m2 mayo 2019	-1,8 %	40.411150	-3.676057	[[[[-3.66327, 40.410011], [-3.663304, 40.40993...	28079603
14	Salamanca	5985	+1,6 %	+3,4 %	+2,3 %	5.985 €/m2 mayo 2020	0,0 %	40.427045	-3.680602	[[[[-3.659193, 40.441787], [-3.659127, 40.4414...	28079604
15	San Blas	2497	-0,8 %	+0,3 %	-1,8 %	3.603 €/m2 nov 2007	-30,7 %	40.427500	-3.615954	[[[[-3.585322, 40.449894], [-3.582224, 40.4498...	28079620
16	Tetuan	3679	-0,9 %	-0,3 %	-1,2 %	3.857 €/m2 dic 2007	-4,6 %	40.460578	-3.698281	[[[[-3.698151, 40.474618], [-3.697655, 40.4745...	28079606
17	Usera	2024	-1,7 %	-2,3 %	-0,7 %	3.110 €/m2 nov 2007	-34,9 %	40.383894	-3.706446	[[[[-3.68322, 40.364736], [-3.683158, 40.36421...	28079612
18	Vicalvaro	2350	+0,7 %	+0,2 %	+3,7 %	2.656 €/m2 nov 2010	-11,5 %	40.396584	-3.576622	[[[[-3.57609, 40.413095], [-3.572811, 40.41181...	28079619
19	Villa de Vallecas	2437	+3,0 %	+3,1 %	+0,7 %	2.955 €/m2 mayo 2008	-17,5 %	40.373958	-3.612163	[[[[-3.608442, 40.387471], [-3.608203, 40.3873...	28079618
20	Villaverde	1728	+0,1 %	-0,1 %	0,0 %	2.900 €/m2 feb 2008	-40,4 %	40.345610	-3.695956	[[[[-3.70508, 40.363653], [-3.703341, 40.36354...	28079617

Now that we have clean the information, and merge it to the main data frame we can draw the map again, this time with the neighborhood boundaries and information as popups with the price per square meter on each neighborhood.



The next data set we need is the venue’s information, for this, I use the foursquare API with a restriction of 200 sites per query (1 query per neighborhood) and without radius limit. With this configuration, foursquare should return 200 (this number can be changed if you want to get more venues per call) most relevant venues around a given location. Since the area of each neighborhood can overlap with other neighborhoods we have to check for each point if are inside the boundaries of each polygon we have defined for each neighborhood in the case is not, the point will be drop.

```
In [26]: # Since foursquare returns venius based in a radious we have to filter venues that does not belong to each neighborhood
# the number of venues can change based on the moment of call the Foursquare API
counter = 0
for neighborhood, ven_lon,ven_lat in zip(madrid_venues['Neighborhood'], madrid_venues['Venue Latitude'], madrid_venues['Venue Longitude']):
    poly_index = df[df['neighborhood'] == neighborhood].index[0]
    row = df[df['neighborhood'] == neighborhood]
    polygon_array=row['geometry.coordinates'][poly_index][0][0]
    point = Point(ven_lat,ven_lon)
    #print(point)
    polygon = Polygon(polygon_array)
    #print(polygon)
    if(not polygon.contains(point)):
        # print("Removing venue"+str(point))
        # print("Not inside neighborhood"+str(polygon))
        madrid_venues.drop(counter, inplace=True)
    counter=counter+1
madrid_venues.reset_index()
madrid_venues.shape

Out[26]: (1348, 7)
```

The next step now we have the data of the venues inside each neighborhood is to determine what are the most common venues for which is used the function get_dummies to create a data frame that shows us the distribution of venues per neighborhood.

```
In [28]: # now that we have most relevant venues for each neighborhood we have to determine which are the most common venues.
#For this we transform the information to categorical information
madrid_onehot = pd.get_dummies(madrid_venues[['Venue Category']], prefix="", prefix_sep="")

# get Neighborhood column index
NhIndex = madrid_onehot.columns.get_loc("Neighborhood")

# copy Neighborhood into a temporal variable
tmp_Nh = madrid_onehot['Neighborhood']

# delete Neighborhood column
madrid_onehot.drop(madrid_onehot.columns[NhIndex], axis=1, inplace=True)

# insert the column in the index 0 of the dataframe
madrid_onehot.insert(0, 'Neighborhood', tmp_Nh)

# add neighborhood data back to dataframe
madrid_onehot['Neighborhood'] = madrid_venues['Neighborhood']

madrid_onehot.head()
```

Out[28]:

	Neighborhood	Accessories Store	Airport	Airport Lounge	Airport Service	American Restaurant	Aquarium	Arcade	Arepa Restaurant	Argentinian Restaurant	...	Trail	Train Station
0	Arganzuela	0	0	0	0	0	0	0	0	0	...	0	0
1	Arganzuela	0	0	0	0	0	0	0	0	0	...	0	0
2	Arganzuela	0	0	0	0	0	0	0	0	0	...	0	0
3	Arganzuela	0	0	0	0	0	0	0	0	0	...	0	0
4	Arganzuela	0	0	0	0	0	0	0	0	0	...	0	0

Now that we have the number of venues per neighbor we calculate the mean to obtain ve average of each type of venue per neighborhood.

	Neighborhood	Accessories Store	Airport	Airport Lounge	Airport Service	American Restaurant	Aquarium	Arcade	Arepa Restaurant
0	Arganzuela	0.0000	0.0000	0.0000	0.000	0.000000	0.000000	0.000000	0.01
1	Barajas	0.0125	0.0125	0.0625	0.075	0.000000	0.000000	0.000000	0.00
2	Carabanchel	0.0000	0.0000	0.0000	0.000	0.000000	0.000000	0.000000	0.00
3	Centro	0.0100	0.0000	0.0000	0.000	0.010000	0.000000	0.000000	0.00
4	Chamartin	0.0000	0.0000	0.0000	0.000	0.010309	0.000000	0.010309	0.00
5	Chamberi	0.0000	0.0000	0.0000	0.000	0.010753	0.000000	0.000000	0.00
6	Ciudad Lineal	0.0000	0.0000	0.0000	0.000	0.000000	0.000000	0.000000	0.00

Recommendation System

With the result information of each neighborhood, we can create a recommendation system that based on the preferences of a user can show him which neighborhood has more venues in common of whit the user preferences and which is the cost per square meter of each neighborhood.

In order to create the recommendation system, the first step is to determine with the information we have what type of recommendation system we can use. In this case, the recommendation will be based on the information we have defined for each neighborhood (venues), so this type of recommendation system is a content-based recommendation system. In this type of recommendation first, we define features/preferences for each user. In this case, we gonna create a fake user and assign a set of features as user preferences.

```
user_preferences = ['Bar','Garden','Park','Gym']
user_preferences
```

```
user_profile = pd.DataFrame(features)
user_profile.rename(columns={0:'features'}, inplace=True)
user_profile.head()
```

Now that preferences are defined as a list we have to convert it into numerical values and then multiplied by the matrix of neighborhood information.

At this point, we can assign weights to the user preferences if we want that one preference will be more important than others, for simplicity in this case all the preferences will be valued as 1.

```python
feature_values=[]
for feature in user_profile['features']:
    #print(str(feature)+" vs "+str(user_preferences))
    if(feature in user_preferences):
        feature_values.append(1)
        print(feature)
    else:
        feature_values.append(0)
feature_values
```

As a result of multiply, the user profile with the neighborhood data we will get a matrix that represents if a neighborhood has one or more of the user preferences.

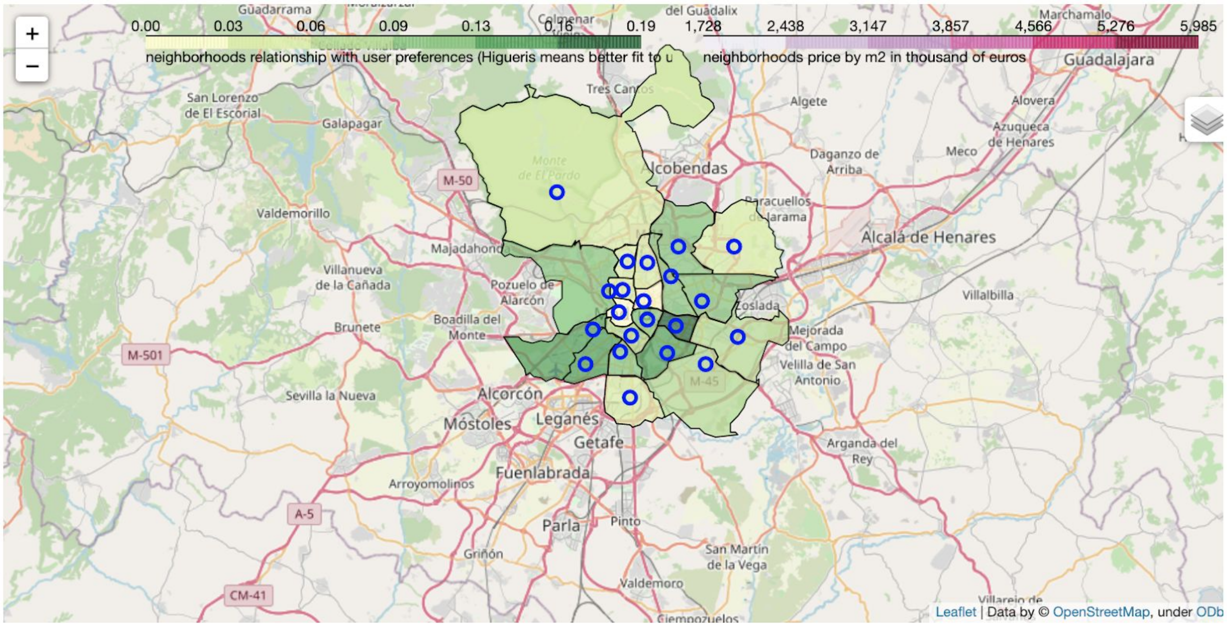
```python
df_features=pd.DataFrame(feature_values, columns={'features_val'})
madrid_recomendation_matrix = features_neighborhood.mul(feature_values, axis=0)
madrid_recomendation_matrix
```

The final result of the recommendation will be calculated as the sum per neighborhood where at a higher number most related is the neighbor with the user preferences

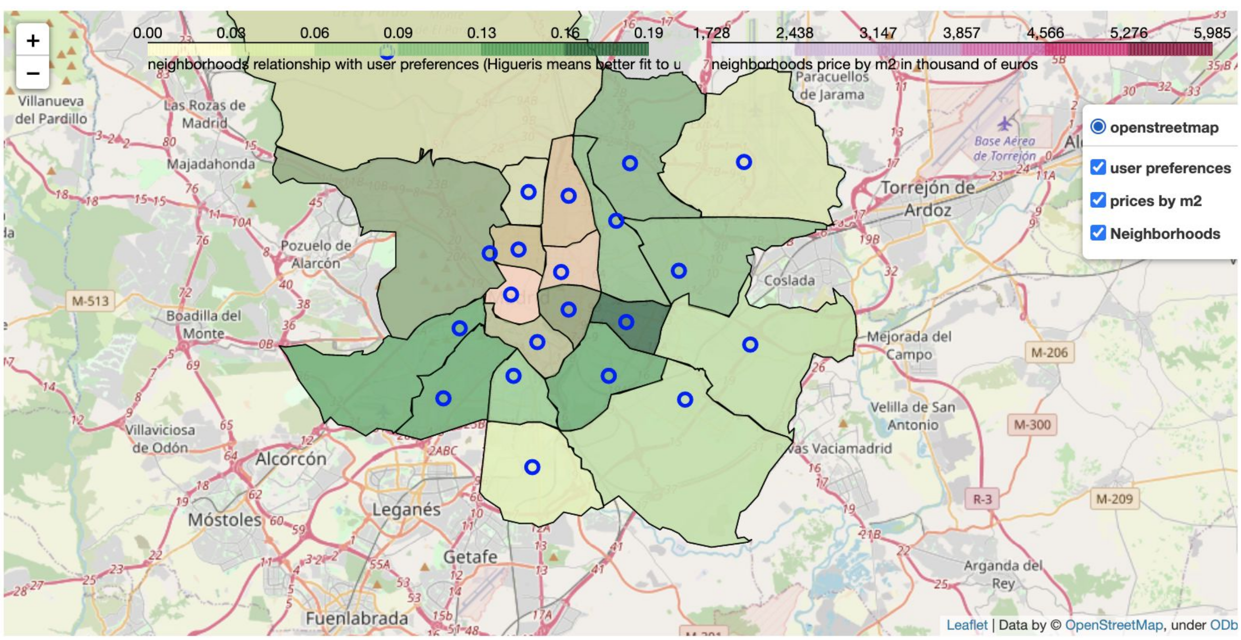
```python
recomentdation_totals = madrid_recomendation_matrix.sum(axis = 0, skipna = True)
recomentdation_totals.head()
```

```
Neighborhood
Arganzuela      0.070000
Barajas         0.037500
Carabanchel     0.157143
Centro          0.030000
Chamartin       0.051546
dtype: float64
```

With this result, we can order the data from higher values to lowe values and determine which neighborhoods are more related to user preferences. This can be better represented by a map where darker green means higher relation with the user preferences.



Finally to relate the price with the recommendation system result is possible to use layer in the choropleth map so we can visualize bot results or filter just by each of them. In this case, the relation is not applied directly since it is unknown how much a user is willing to pay per square meter in a location that covers his preferences, so we left this decision to each user at least in this initial part of the analysis.



Results

As a result, with this type of map, we can overlap bot results prices and recommendations and analyze based on how much the person can pay wich neighborhood is better and fits his preferences. Other possible conclusions are:

- The price of the square meter is higher in the center and north parts of the city.
- This type of recommendation system produces a broad result so it helps the user to make a more rational decision, but there are still factors that can affect the user decisions.
- In this case, some of the neighborhoods that cover most of the user preferences are not the most expensive in the city, so the preferences of the user will affect how much a user should pay for a place to live.
- In this case, Mortaraz, Latina, Carabanchel, punte de Vallecas cover with the most preferences of the user at lower prices by square meter.
- Mortaraz is the neighborhood that covers all the user preferences.

This analysis is an initial approach of how to create a recommendation based on a given user, there are still different aspects that can be improved as weights in the user preferences, the relation between recommendation results with the prices, crime index on each neighborhood and type of buildings per neighborhood (houses, apartments, etc).

Conclusion

During the development of this project is possible to establish how once we acquire data different possibilities of analysis arouses, each one of them with very promising results. Is very important always have in mind what is your goal from the start of the analysis and focus on getting the necessary data to answer the initial question because is very easy to get distracted with the additional result the data can produce and loss the focus on what was really the initial objective. This project gives a structured vision of how a problem can be approached solving step by step how to obtain the necessary data to respond to the proposed question. The next steps for this analysis add new important perspectives at the moment to look for a house and try to obtain additional information on the type of property the user is looking for. I hope this analysis will be interest for other persons and I be open to any commentary.