The Best Place in Madrid

Installing Charging Station of Electrical Vehicle

MADRID

Introduction/Business Problem

Madrid is the capital of Spain and the most populous city in the country that has an expanding metropolitan area that includes more than 6.5 million inhabitants that place it among the top 5 of the European Union.

It is a city with a long history and therefore its morphology has evolved through the ages. It presents a great variety of urban paths. It is divided into districts, which are subdivided into neighborhoods. It has 21 districts and 131 neighborhoods.

The personal transport vehicle has been one of the distinctive elements of urban transport due to the dispersion of the population and the historical difficulties in accessing work and leisure centers by public transport. The city traffic jams are famous.

In recent years, sustainable mobility plans have been enhanced to facilitate the lives of its citizens and help prevent climate change.

One of the key elements in this new mobility model is personal transport using electric vehicles, which enjoy great environmental and fiscal advantages.

The objective of this project is to analyze the charging points of electric vehicles available in the city of Madrid, to help an investor find the best location between the neighborhoods of Madrid to install a new charging point.

Approach

- Collect the Madrid city data from ttps://es.wikipedia.org/wiki/Madrid#Distritos
- Collect the Madrid station charge of electrical vehicles from Google Earth.
- Using FourSquare API we will find all venues for each station charging.
- Using FourSquare API we will find all venues for each neighborhood.
- Cluster the data to find the ideal location of a charging station.

Questions that can be asked using the above mentioned datasets

- What is best location in Madrid for stations charging electrical vehicles?
- Which areas have potential station charging electrical vehicles market?

Part 1 : DATA.

For this project we need the following data:

- Data of the city of Madrid containing the list of Districts and neighborhoods containing their latitude and longitude.
  - Data source: https://es.wikipedia.org/wiki/Madrid#Distritos
  - Description: On this page we find the list of districts and neighborhoods linking each of them a page with their geographical data.
- Electric vehicle charging points in the city of Madrid.

    - Data Source: Gogle Earth Pro. Search “electric vehicle charging station madrid”. Provide the kml file charging stations
    - Description: Using Google Earth we can download a file in kml format with the location of the current charging points. A kml file is an xml file that contains the coordinates of the current points.
- Places near each charging station.
    - Data Source: Foursquare API
    - Description: By using this api we will get all the venues in each neighborhood. We will use these places to look for patterns in current locations and apply those patterns to find new locations.

Get the data

[1]:

```
import requests
import pandas as pd

#uncomment this line on error
!pip install bs4
from bs4 import BeautifulSoup
#uncomment this line on error
!pip install Wikipedia
import wikipedia as wp
print('Libraries imported.')
```

Libraries imported.

[2]:

```
website_url = requests.get('https://es.wikipedia.org/wiki/Madrid#Distritos').text
#website_url
```

Reading the web page.

[3]:

```
#convert the request
soup = BeautifulSoup(website_url, 'html.parser')
#print(soup.prettify())            #Uncomment if you want to see the data
```

The page has a section with an ordered list of districts. Their neighborhoods are listed next to each district.

[4]:

```
listas = soup.find_all('ol')
```

The list of Districts corresponds to the first list that appears on the page.

[5]:

```
listas[0]
```

```
<ol><li><b><a href="/wiki/Centro_(Madrid)" title="Centro (Madrid)">Centro</a>.</b> <a href
="/wiki/Palacio_(Madrid)" title="Palacio (Madrid)">Palacio</a>, <a href="/wiki/Embajadores_(
Madrid)" title="Embajadores (Madrid)">Embajadores</a>, <a href="/wiki/Cortes_(Madrid)" titl
e="Cortes (Madrid)">Cortes</a>, <a href="/wiki/Justicia_(Madrid)" title="Justicia (Madrid)">J
usticia</a>, <a class="mw-redirect" href="/wiki/Malasa%C3%B1a" title="Malasaña">Universid
ad</a> y <a href="/wiki/Sol_(Madrid)" title="Sol (Madrid)">Sol</a>.</li>
```

Arganzuela. <a class="mw-re
direct" href="/wiki/Imperial_(Arganzuela)" title="Imperial (Arganzuela)">Imperial, <a clas
s="mw-redirect" href="/wiki/Las_Acacias" title="Las Acacias">Acacias, <a class="mw-red
irect" href="/wiki/La_Chopera" title="La Chopera">La Chopera, <a href="/wiki/Legazpi_(
Madrid)" title="Legazpi (Madrid)">Legazpi, <a class="mw-redirect" href="/wiki/Las_Delic
ias_(Madrid)" title="Las Delicias (Madrid)">Delicias, <a href="/wiki/Palos_de_Moguer_(
Madrid)" title="Palos de Moguer (Madrid)">Palos de Moguer y <a href="/wiki/Atocha_(Ma
drid)" title="Atocha (Madrid)">Atocha.
Retiro. <a href="/wi
ki/Pac%C3%ADfico_(Madrid)" title="Pacífico (Madrid)">Pacífico, <a href="/wiki/Adelfas
" title="Adelfas">Adelfas, Estr
ella, Ibiza, <a href="/wiki/Jer%
C3%B3nimos_(Madrid)" title="Jerónimos (Madrid)">Jerónimos y <a href="/wiki/Ni%C3%
B1o_Jes%C3%BAs_(Madrid)" title="Niño Jesús (Madrid)">Niño Jesús.
Salamanca:
Recoletos, <a href="/wiki/
Goya_(Madrid)" title="Goya (Madrid)">Goya, <a class="mw-redirect" href="/wiki/Fuente_
del_Berro_(Madrid)" title="Fuente del Berro (Madrid)">Fuente del Berro, <a class="mw-re
direct" href="/wiki/Guindalera" title="Guindalera">Guindalera, <a href="/wiki/Lista_(Madr
id)" title="Lista (Madrid)">Lista y <a href="/wiki/Castellana_(Madrid)" title="Castellana (
Madrid)">Castellana.
Chamartín: <a class="
mw-redirect" href="/wiki/Colonia_de_El_Viso" title="Colonia de El Viso">El Viso, <a href
="/wiki/Prosperidad_(Madrid)" title="Prosperidad (Madrid)">Prosperidad, <a href="/wiki/C
iudad_Jard%C3%ADn_(Madrid)" title="Ciudad Jardín (Madrid)">Ciudad Jardín, <a href="/
wiki/Hispanoam%C3%A9rica_(Madrid)" title="Hispanoamérica (Madrid)">Hispanoamérica, Nueva Esp
aña y Castilla.
Tetuán: <a
href="/wiki/Bellas_Vistas" title="Bellas Vistas">Bellas Vistas, <a href="/wiki/Cuatro_Cam
inos_(Madrid)" title="Cuatro Caminos (Madrid)">Cuatro Caminos, <a href="/wiki/Castillej
os_(Madrid)" title="Castillejos (Madrid)">Castillejos, <a href="/wiki/Almenara_(Madrid)" t
itle="Almenara (Madrid)">Almenara, <a href="/wiki/Valdeacederas" title="Valdeacederas"
>Valdeacederas y Berru
guete.
Chamberí: <a href="/wi
ki/Gaztambide_(Madrid)" title="Gaztambide (Madrid)">Gaztambide, <a href="/wiki/Arapil
es_(Madrid)" title="Arapiles (Madrid)">Arapiles, <a href="/wiki/Trafalgar_(Madrid)" title=
"Trafalgar (Madrid)">Trafalgar, <a href="/wiki/Almagro_(Madrid)" title="Almagro (Madri
d)">Almagro, <a href="/wiki/R%C3%ADos_Rosas_(Madrid)" title="Ríos Rosas (Madrid)"
>Ríos Rosas y Val
lehermoso.
Fuencarral-El Pardo</
a>: El Pardo, <a href="/
wiki/Fuentelarreina" title="Fuentelarreina">Fuentelarreina, <a href="/wiki/Pe%C3%B1agra
nde" title="Peñagrande">Peñagrande, <a class="mw-redirect" href="/wiki/Barrio_del_Pilar
_(Madrid)" title="Barrio del Pilar (Madrid)">Barrio del Pilar, <a href="/wiki/La_Paz_(Madr
id)" title="La Paz (Madrid)">La Paz, <a href="/wiki/Valverde_(Madrid)" title="Valverde (
Madrid)">Valverde, Mirasierra y <a href
="/wiki/El_Goloso" title="El Goloso">El Goloso.
Moncloa-Aravaca: Casa de Campo, <a href="/wik
i/Arg%C3%BCelles_(Madrid)" title="Argüelles (Madrid)">Argüelles, <a href="/wiki/Ciuda
d_Universitaria_de_Madrid" title="Ciudad Universitaria de Madrid">Ciudad Universitaria,
Valdezarza, <a href="/wiki/Valdemar%C3
%ADn" title="Valdemarín">Valdemarín, <a href="/wiki/El_Plant%C3%ADo" title="El Pla
ntío">El Plantío y Aravaca.
Latina: <a href="/wi
ki/Los_C%C3%A1rmenes" title="Los Cármenes">Los Cármenes, <a class="mw-redirect" h
ref="/wiki/Barrio_de_Puerta_del_%C3%81ngel" title="Barrio de Puerta del Ángel">Puerta del

Ángel, Lucer
o, Aluche, <a href="/wiki/Campamento_(Madri
d)" title="Campamento (Madrid)">Campamento, <a href="/wiki/Cuatro_Vientos_(Madrid)"
title="Cuatro Vientos (Madrid)">Cuatro Vientos y <a href="/wiki/Las_%C3%81guilas" title
="Las Águilas">Las Águilas.
Carabanchel: <a class="mw-
redirect" href="/wiki/Barrio_de_Comillas" title="Barrio de Comillas">Comillas, <a class="
mw-redirect" href="/wiki/Barrio_de_Opa%C3%B1el" title="Barrio de Opañel">Opañel, <a
class="mw-redirect" href="/wiki/Barrio_de_San_Isidro_(Madrid)" title="Barrio de San Isidro (
Madrid)">San Isidro, <a class="mw-redirect" href="/wiki/Barrio_de_Vista_Alegre" title="
Barrio de Vista Alegre">Vista Alegre, <a class="mw-redirect" href="/wiki/Barrio_de_Puert
a_Bonita" title="Barrio de Puerta Bonita">Puerta Bonita, <a class="mw-redirect" href="/wi
ki/Barrio_de_Buenavista" title="Barrio de Buenavista">Buenavista y <a class="mw-redirect
" href="/wiki/Barrio_de_Abrantes" title="Barrio de Abrantes">Abrantes.
Usera: <a href="/wiki/Orcasitas" title="O
rcasitas">Orcasitas, Orcasur, <a class="mw-r
edirect" href="/wiki/Barrio_de_San_Ferm%C3%ADn" title="Barrio de San Fermín">San Fermí
n, Almendrales, <a href="/wiki/Mos
card%C3%B3_(Madrid)" title="Moscardó (Madrid)">Moscardó, <a href="/wiki/Zof%C3%
ADo" title="Zofío">El Zofío y <a href="/wiki/Pradolongo_(Madrid)" title="Pradolongo (M
adrid)">Pradolongo.
Puente de Vallecas
: Entrevías, <a href="/wiki/San_
Diego_(Madrid)" title="San Diego (Madrid)">San Diego, <a href="/wiki/Palomeras_Bajas"
title="Palomeras Bajas">Palomeras Bajas, <a href="/wiki/Palomeras_Sureste" title="Palom
eras Sureste">Palomeras Sureste, <a href="/wiki/Portazgo_(Madrid)" title="Portazgo (Madr
id)">Portazgo y Numanci
a.
Moratalaz: <a href="/wiki/Pavon
es" title="Pavones">Pavones,
Horcajo, Marroquina, <a href="/wiki/
Media_Legua_(Madrid)" title="Media Legua (Madrid)">Media Legua, <a href="/wiki/Font
arr%C3%B3n" title="Fontarrón">Fontarrón y
Vinateros.
Ciudad Lineal: <a href=
"/wiki/Ventas_(Madrid)" title="Ventas (Madrid)">Ventas, <a href="/wiki/Pueblo_Nuevo_(
Madrid)" title="Pueblo Nuevo (Madrid)">Pueblo Nuevo, <a href="/wiki/Quintana_(Madrid
)" title="Quintana (Madrid)">Quintana, <a href="/wiki/Concepci%C3%B3n_(Madrid)" title
="Concepción (Madrid)">La Concepción, <a href="/wiki/San_Pascual_(Madrid)" title="Sa
n Pascual (Madrid)">San Pascual, <a href="/wiki/San_Juan_Bautista_(Madrid)" title="San
Juan Bautista (Madrid)">San Juan Bautista, <a href="/wiki/Colina_(Madrid)" title="Colina
(Madrid)">Colina, Atalaya y <a class="mw-redirect" href="/wiki/Pinar_de_Chamart%C3%ADn" title="Pinar de Chamartí
n">Costillares.
Hortaleza: <a href="/wiki/Paloma
s_(Madrid)" title="Palomas (Madrid)">Palomas, Pi
overa, C
anillas, Pinar del
Rey, <a href="/wiki/Ap%C3%B3stol_Santiago_(Madrid)" title="Apóstol Santiago (Madrid)
">Apóstol Santiago y <a href="/wiki/Valdefuentes_(Madrid)" title="Valdefuentes (Madrid)
">Valdefuentes.

Villaverde: <a class="mw-redirect" href="/wiki/Villaverde_Alto_(Madrid)" title="Villa
verde Alto (Madrid)">Villaverde Alto, <a class="mw-redirect" href="/wiki/San_Crist%C3
%B3bal_de_los_%C3%81ngeles" title="San Cristóbal de los Ángeles">San Cristóbal, <a hr
ef="/wiki/Butarque" title="Butarque">Butarque, <a href="/wiki/Los_Rosales_(Madrid)" titl
e="Los Rosales (Madrid)">Los Rosales y <a href="/wiki/Los_%C3%81ngeles_(Madrid)" ti
tle="Los Ángeles (Madrid)">Los Ángeles.
Villa de Vallecas:
<a href="/wiki/Casco_Hist%C3%B3rico_de_Vallecas" title="Casco Histórico de Vallecas">Cas

```
co Histórico de Vallecas</a>, <a href="/wiki/Santa_Eugenia_(Madrid)" title="Santa Eugenia (M
adrid)">Santa Eugenia</a> y <a href="/wiki/Ensanche_de_Vallecas" title="Ensanche de Valleca
s">Ensanche de Vallecas</a>.</li>
<li><b><a href="/wiki/Vic%C3%A1lvaro" title="Vicálvaro">Vicálvaro</a></b>: <a class="m
w-redirect" href="/wiki/Casco_Hist%C3%B3rico_de_Vic%C3%A1lvaro_(barrio)" title="Casco
Histórico de Vicálvaro (barrio)">Casco Histórico de Vicálvaro</a>, <a href="/wiki/Valdebernar
do" title="Valdebernardo">Valdebernardo</a>, <a href="/wiki/Valderrivas" title="Valderrivas"
>Valderrivas</a> y <a href="/wiki/El_Ca%C3%B1averal" title="El Cañaveral">El Cañaveral</
a>.</li>
<li><b><a href="/wiki/San_Blas-Canillejas" title="San Blas-Canillejas">San Blas-Canillejas</a
></b>: <a href="/wiki/Simancas_(Madrid)" title="Simancas (Madrid)">Simancas</a>, <a href=
"/wiki/Hell%C3%ADn_(Madrid)" title="Hellín (Madrid)">Hellín</a>, <a href="/wiki/Amposta
_(Madrid)" title="Amposta (Madrid)">Amposta</a>, <a href="/wiki/Arcos_(Madrid)" title="Ar
cos (Madrid)">Arcos</a>, <a href="/wiki/Rosas_(Madrid)" title="Rosas (Madrid)">Rosas</a>,
<a href="/wiki/Rejas_(Madrid)" title="Rejas (Madrid)">Rejas</a>, <a class="mw-redirect" href
="/wiki/Canillejas_(Madrid)" title="Canillejas (Madrid)">Canillejas</a> y <a href="/wiki/Salva
dor_(Madrid)" title="Salvador (Madrid)">Salvador</a>.</li>
<li><b><a href="/wiki/Barajas" title="Barajas">Barajas</a></b>: <a href="/wiki/Alameda_de_
Osuna" title="Alameda de Osuna">Alameda de Osuna</a>, <a href="/wiki/Aeropuerto_(Madrid
)" title="Aeropuerto (Madrid)">Aeropuerto</a>, <a href="/wiki/Casco_Hist%C3%B3rico_de_B
arajas" title="Casco Histórico de Barajas">Casco Histórico de Barajas</a>, <a href="/wiki/Tim
%C3%B3n_(Madrid)" title="Timón (Madrid)">Timón</a> y <a href="/wiki/Corralejos" title="
Corralejos">Corralejos</a>.</li></ol>
```

We access one of the linked pages and retrieve its content to read the coordinates.

[6]:

```
urlwikipage ='https://es.wikipedia.org' + '/wiki/Arapiles_(Madrid)'
wikipage_url = requests.get(urlwikipage).text
wikipage_url
#convert the request
soupwiki = BeautifulSoup(wikipage_url, 'html.parser')
```

The location, latitude and longitude, is in a link to another page. We locate the links that have the same class.

[7]:

```
celdaCoords=soupwiki.find_all('a',{"class":"external text"})
```

[8]:

```
celdaCoords[1]
```

```
<a class="external text" href="http://tools.wmflabs.org/geohack/geohack.php?language=es&amp
;pagename=Arapiles_(Madrid)&amp;params=40.43416667_N_-3.70777778_E_type:city"><spa
n class="geo-default"><span class="geo-dms" title="Mapas, fotos y otros datos de 40°26′03″N 3
°42′28″O"><span class="latitude">40°26′03″N </span><span class="longitude">3°42′28″O</sp
an></span></span><span class="geo-multi-punct"> / </span><span class="geo-nondefault"><s
pan class="geo-dec" title="Mapas, fotos y otros datos de 40.43416667 -3.70777778"><span clas
s="geo"><span class="latitude">40.43416667, </span><span class="longitude">-3.70777778</s
pan></span></span></span></a>
```

[9]:

```
latitud=float(celdaCoords[1].find_all('span',{"class":"latitude"})[1].text.split(',')[0])
print(latitud)
```

```
40.43416667
```

[10]:

```
longitud=float(celdaCoords[1].find_all('span',{"class":"longitude"})[1].text.split(',')[0])
print(longitud)
```

```
-3.70777778
```

Once we have identified the processes of the data source pages, we will recover the data of each District and its neighborhoods iterating from the initial page to create a dataframe with the coordinates.

[11]:

```
rows=[]
for distritos in listas[0].find_all('li'):
    distri=0
    for distrito in distritos.find_all('a'):
        print(distrito.get('title'))
        urlwikipage ='https://es.wikipedia.org' + distrito.get('href')
        wikipage_url = requests.get(urlwikipage).text
        soupwiki = BeautifulSoup(wikipage_url, 'html.parser')
        celdaCoords=soupwiki.find_all('a',{"class":"external text"})
        try:
            latitud=float(celdaCoords[0].find_all('span',{"class":"latitude"})[1].text.split(',')[0])
            longitud=float(celdaCoords[0].find_all('span',{"class":"longitude"})[1].text.split(',')[0])
        except:
            try:
                latitud=float(celdaCoords[1].find_all('span',{"class":"latitude"})[1].text.split(',')[0])
                longitud=float(celdaCoords[1].find_all('span',{"class":"longitude"})[1].text.split(',')[0])
            except:
                latitud=0
                longitud=0
        if distri==0:
            #print('Distrito:',distrito.get('href'),distrito.get('title'))
            rows.append([distrito.get('title'),distrito.get('href'),'Distrito', latitud, longitud])
        else:
            #print('Barrio:',distrito.get('href'),distrito.get('title'))
            rows.append([distrito.get('title'),distrito.get('href'),'Barrio', latitud, longitud])
        distri +=1
```

```
Centro (Madrid)
Palacio (Madrid)
Embajadores (Madrid)
Cortes (Madrid)
Justicia (Madrid)
Malasaña
Sol (Madrid)
Arganzuela
Imperial (Arganzuela)
Las Acacias
…
…
San Blas-Canillejas
Simancas (Madrid)
Hellín (Madrid)
Amposta (Madrid)
Arcos (Madrid)
Rosas (Madrid)
Rejas (Madrid)
Canillejas (Madrid)
Salvador (Madrid)
Barajas
Alameda de Osuna
Aeropuerto (Madrid)
Casco Histórico de Barajas
Timón (Madrid)
Corralejos
```

[12]:

```
listado=pd.DataFrame(rows)
listado.columns=['Name','link','Type','latitude','longitude']
print(listado.shape)
listado=listado[listado['latitude']!=0]
print(listado.shape)
```

(152, 5)
(151, 5)

We save the dataframe in a csv file for later use.

[13]:

```
listado.to_csv('barrios.csv',index=False)
```

Once downloaded to the csv, the data can be recovered:

[14]:

```
listado = pd.read_csv('barrios.csv')
listado.head()
```

	Name	link	Type	latitude	longitude
0	Centro (Madrid)	/wiki/Centro_(Madrid)	Distrito	40.415347	-3.707371
1	Palacio (Madrid)	/wiki/Palacio_(Madrid)	Barrio	40.415000	-3.713333
2	Embajadores (Madrid)	/wiki/Embajadores_(Madrid)	Barrio	40.408889	-3.699722
3	Cortes (Madrid)	/wiki/Cortes_(Madrid)	Barrio	40.414167	-3.698056
4	Justicia (Madrid)	/wiki/Justicia_(Madrid)	Barrio	40.423889	-3.696389

We set the Madrid city location:

[15]:

```
latitudMadrid=40.418889
longitudMadrid=-3.691944
```

[16]:

```
listadobarrios=listado[listado['Type']=='Barrio'].reset_index()
listadoDistritos=listado[listado['Type']=='Distrito'].reset_index()
```

We prepare to visualize the location of each District, Neighborhood and charging station in Madrid.

[17]:

```
#!conda install -c conda-forge folium=0.5.0 --yes # uncomment this line if you haven't completed the Foursquare API lab
import folium # map rendering library

# create map of Toronto using latitude and longitude values
map_madrid = folium.Map(location=[latitudMadrid, longitudMadrid], zoom_start=11)
```

We load the placeholders of the Districts ...

[18]:

```
# add Distrito markers to map
for lat, lng, label in zip(listadoDistritos['latitude'], listadoDistritos['longitude'],
listadoDistritos['Name']):
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius=5,
        popup=label,
        color='red',
        fill=True,
        fill_color='#FF5733',
        fill_opacity=0.7,
        parse_html=False).add_to(map_madrid)
```

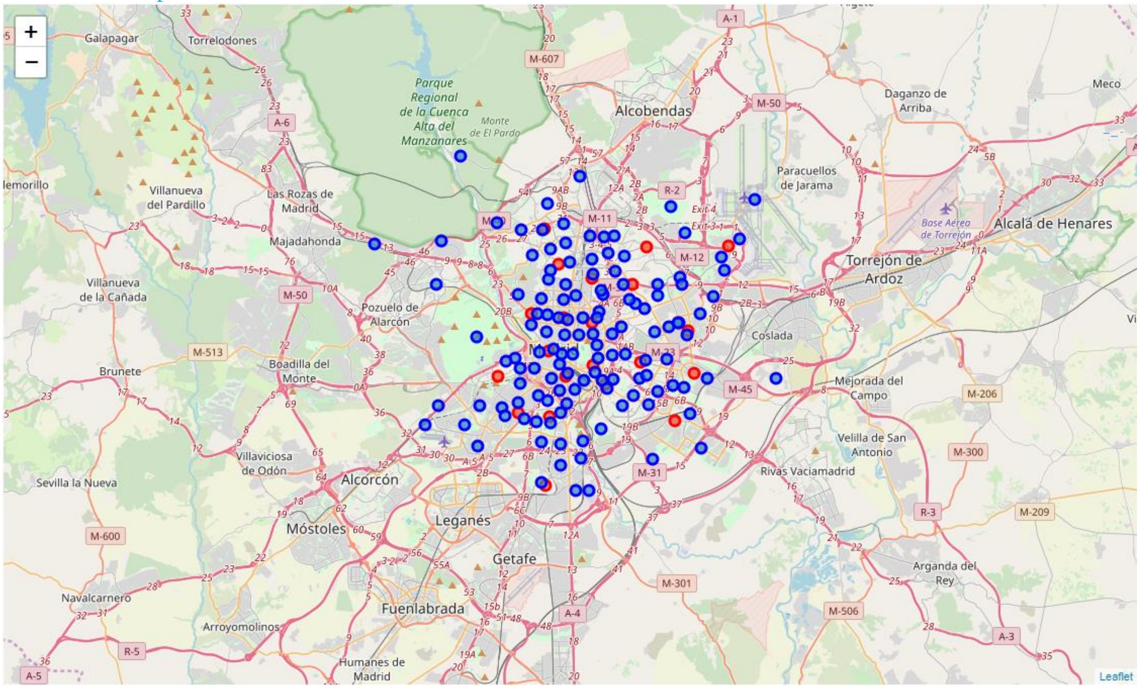
We load the placeholders of each neighborhood ...

[19]:

```
# add markers to map
for lat, lng, label in zip(listadobarrios['latitude'], listadobarrios['longitude'],
listadobarrios['Name']):
    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)
```

[20]:

```
map_madrid
```



We retrieve the location data of the charging stations.

We downloaded the KML file from Google Earth with the export of the search "estación carga vehículo eléctrico madrid".

We read the file from the local route where we have saved it:

[21]:

```
import xml.etree.ElementTree as ET

pathtokml='estacion de carga de vehiculos electrico madrid.kml'
tree=ET.parse(pathtokml)
```

To extract the location data of each charging station we locate the element {http://www.opengis.net/kml/2.2}Placemark that contains the latitude and longitude data.

[22]:

```
places=tree.findall('.//{http://www.opengis.net/kml/2.2}Placemark')
```

We create a dictionary with the Placemark elements and their properties.

[23]:

```
places[0].findall('.//{http://www.opengis.net/kml/2.2}coordinates')[0].text.split(',')[0]
places[0].findall('.//{http://www.opengis.net/kml/2.2}name')[0].text.split(',')[0]
```

'Punto de recarga de Vehículos Eléctricos'

[24]:

```
elementos=[]
for place in places:

latitud=float(place.findall('.//{http://www.opengis.net/kml/2.2}coordinates')[0].text.split(',')[1])

longitud=float(place.findall('.//{http://www.opengis.net/kml/2.2}coordinates')[0].text.split(',')[0])
  nombre=place.findall('.//{http://www.opengis.net/kml/2.2}name')[0].text.split(',')[0]
  direccion=place.findall('.//{http://www.opengis.net/kml/2.2}address')[0].text.split(',')[0]
  elementos.append([nombre,direccion,latitud,longitud])

stations=pd.DataFrame(elementos)
stations.columns=['Name','Address','latitude','longitude']
stations
```

	Name	Address	latitude	longitude
0	Punto de recarga de Vehículos Eléctricos	28046 Madrid	40.431880	-3.689155
1	Enchufauto	Calle de Diego de León	40.434853	-3.678520
2	Recarga eléctrica de coches y motos	Calle de Alfonso XII	40.419270	-3.688927
3	Recarga eléctrica de coches	Paseo de la Castellana	40.458343	-3.689345

	Name	Address	latitude	longitude
4	Recarga eléctrica de coches	Calle de los Chulapos	40.407397	-3.720168
5	Punto de Carga para Vehiculo Eléctrico Sta. En...	Calle de Sta Engracia	40.438250	-3.700276
6	Recarga eléctrica de coches y motos	Calle de Goya	40.424696	-3.671250
7	Punto de Carga Vehículos Eléctricos	Parking ifema	40.464751	-3.618176
8	Punto de recarga Vehículo Eléctrico	Calle de San Bernardo	40.427178	-3.706448

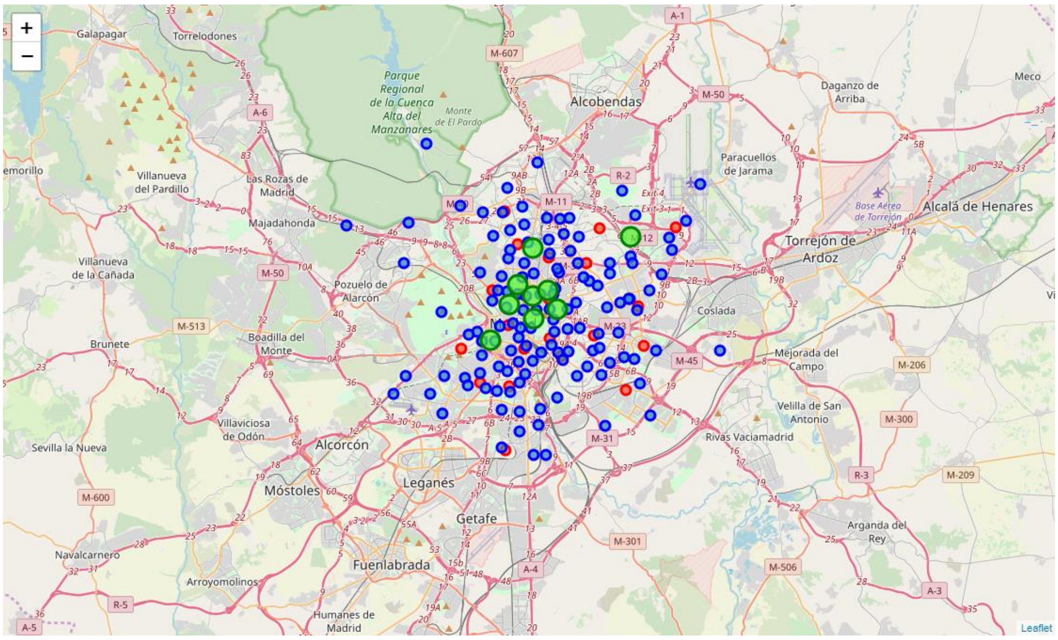
We load the placeholders of each charging station ...

[25]:

```
# add markers to map
for lat, lng, label in zip(stations['latitude'], stations['longitude'], stations['Name']):
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius=10,
        popup=label,
        color='green',
        fill=True,
        fill_color='#4FFF33',
        fill_opacity=0.7,
        parse_html=False).add_to(map_madrid)
```

[26]:

```
map_madrid
```



Explore the neighborhoods in Madrid

Next, we are going to start utilizing the Foursquare API to explore the neighborhoods and segment them.

[27]:

```
VERSION = '20180605' # Foursquare API version

CLIENT_ID = 'myclientid'
CLIENT_SECRET = 'myClientSecret'
```

Let's explore the first neighborhood in our dataframe.

[28]:

```
listadobarrios.loc[0,'Name']
```

[29]:

```
'Palacio (Madrid)'
```

Get the neighborhood's latitude and longitude values.

[30]:

```
neighborhood_latitude = listadobarrios.loc[0, 'latitude'] # neighborhood latitude value
neighborhood_longitude = listadobarrios.loc[0, 'longitude'] # neighborhood longitude value

neighborhood_name = listadobarrios.loc[0, 'Name'] # neighborhood name

print('Latitude and longitude values of {} are {}, {}.'.format(neighborhood_name,
                                                               neighborhood_latitude,
                                                               neighborhood_longitude))
```

```
Latitude and longitude values of Palacio (Madrid) are 40.415, -3.71333333.
```

Now, let's get the top 100 venues that are in Marble Hill within a radius of 500 meters.

First, let's create the GET request URL. Name your URL **url**.

[31]:

```
url =
'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={},{}
&radius={}&limit={}'.format(
    CLIENT_ID,
    CLIENT_SECRET,
    VERSION,
    neighborhood_latitude,
    neighborhood_longitude,
    500,
    100)


url
```

```
'https://api.foursquare.com/v2/venues/explore?&client_id=DPQJAJTCGDD4023TRPCVLR5FL
NABK1LEUIJVA4AQTQEYQ3&client_secret=TFW0DWXXGODRA2JD3OU1NCDMMXA
N1KSKF3R3GAEWI&v=20180605&ll=40.415,-3.71333333&radius=500&limit=100'
```

[32]:

```
results = requests.get(url).json()
```

We know that all the information is in the *items* key. Before we proceed, let's borrow the **get_category_type** function from the Foursquare lab.

[33]:

```
# function that extracts the category of the venue
def get_category_type(row):
    try:
        categories_list = row['categories']
    except:
        categories_list = row['venue.categories']

    if len(categories_list) == 0:
        return None
    else:
        return categories_list[0]['name']
```

Now we are ready to clean the json and structure it into a pandas dataframe.

[34]:

```
results
```


```
{'meta': {'code': 200, 'requestId': '5dc43aace97dfb002c18cea2'},
 'response': {'suggestedFilters': {'header': 'Tap to show:',
   'filters': [{'name': 'Open now', 'key': 'openNow'}]},
  'headerLocation': 'La Latina',
  'headerFullLocation': 'La Latina, Madrid',
  'headerLocationGranularity': 'neighborhood',
  'totalResults': 139,
  'suggestedBounds': {'ne': {'lat': 40.4195000045, 'lng': -3.7074339503134666},
   'sw': {'lat': 40.4104999955, 'lng': -3.719232709686534}},
  'groups': [{'type': 'Recommended Places',
    'name': 'recommended',
    'items': [{'reasons': {'count': 0,
       'items': [{'summary': 'This spot is popular',
         'type': 'general',
         'reasonName': 'globalInteractionReason'}]},
      'venue': {'id': '4adcda38f964a520523c21e3',
       'name': 'Santa Iglesia Catedral de Santa María la Real de la Almudena (Catedral de la Almud
ena)',
       'location': {'address': 'C. Bailén, 8-10',
        'lat': 40.41576693264202,
        'lng': -3.7145161628723145,
        'labeledLatLngs': [{'label': 'display',
          'lat': 40.41576693264202,
          'lng': -3.7145161628723145}],
        'distance': 131,
        'postalCode': '28013',
        'cc': 'ES',
        'city': 'Madrid',
        'state': 'Madrid',
        'country': 'España',
        'formattedAddress': ['C. Bailén, 8-10',
         '28013 Madrid Madrid',
         'España']},
       'categories': [{'id': '4bf58dd8d48988d132941735',
         'name': 'Church',
         'pluralName': 'Churches',
         'shortName': 'Church',
```

'icon': {'prefix': 'https://ss3.4sqi.net/img/categories_v2/building/religious_church_',
'suffix': '.png'},
'primary': True}],
'photos': {'count': 0, 'groups': []}},
'referralId': 'e-0-4adcda38f964a520523c21e3-0'},
{'reasons': {'count': 0,
'items': [{'summary': 'This spot is popular',
'type': 'general',
'reasonName': 'globalInteractionReason'}]},
'venue': {'id': '4c321c9816adc9281693c19c',
'name': 'Cervecería La Mayor',
'location': {'address': 'C. Mayor, 77',
'crossStreet': 'C. Bailén',
'lat': 40.41521786102789,
'lng': -3.7121938520878386,
'labeledLatLngs': [{'label': 'display',
'lat': 40.41521786102789,
'lng': -3.7121938520878386}],
'distance': 99,
…
…
'lat': 40.41376624629832,
'lng': -3.7111119473064993}],
'distance': 233,
'postalCode': '28005',
'cc': 'ES',
'neighborhood': 'La Latina',
'city': 'Madrid',
'state': 'Madrid',
'country': 'España',
'formattedAddress': ['Travesia del conde 4 (calle segovia 10)',
'28005 Madrid Madrid',
'España']},
'categories': [{'id': '4bf58dd8d48988d1c1941735',
'name': 'Mexican Restaurant',
'pluralName': 'Mexican Restaurants',
'shortName': 'Mexican',
'icon': {'prefix': 'https://ss3.4sqi.net/img/categories_v2/food/mexican_',
'suffix': '.png'},
'primary': True}],
'photos': {'count': 0, 'groups': []}},
'referralId': 'e-0-4fdee651bb3dfbb6b77f8fec-98'},
{'reasons': {'count': 0,
'items': [{'summary': 'This spot is popular',
'type': 'general',
'reasonName': 'globalInteractionReason'}]},
'venue': {'id': '4cffc4d3ba1da1cddb118528',
'name': 'Mi Ciudad I',
'location': {'address': 'Calle Fuentes, 11',
'lat': 40.41747812755078,
'lng': -3.708624896659918,
'labeledLatLngs': [{'label': 'display',
'lat': 40.41747812755078,
'lng': -3.708624896659918}],
'distance': 485,
'postalCode': '28013',
'cc': 'ES',
'city': 'Madrid',
'state': 'Madrid',
'country': 'España',

'formattedAddress': ['Calle Fuentes, 11',
 '28013 Madrid Madrid',
 'España']},
 'categories': [{'id': '4bf58dd8d48988d1c1941735',
 'name': 'Mexican Restaurant',
 'pluralName': 'Mexican Restaurants',
 'shortName': 'Mexican',
 'icon': {'prefix': 'https://ss3.4sqi.net/img/categories_v2/food/mexican_',
 'suffix': '.png'},
 'primary': True}],
 'photos': {'count': 0, 'groups': []}},
 'referralId': 'e-0-4cffc4d3ba1da1cddb118528-99'}]}]}}

[35]:

```
import json # library to handle JSON files
from pandas.io.json import json_normalize # tranform JSON file into a pandas dataframe

venues = results['response']['groups'][0]['items']

nearby_venues = json_normalize(venues) # flatten JSON

# filter columns
filtered_columns = ['venue.name', 'venue.categories', 'venue.location.lat', 'venue.location.lng']
nearby_venues =nearby_venues.loc[:, filtered_columns]

# filter the category for each row
nearby_venues['venue.categories'] = nearby_venues.apply(get_category_type, axis=1)

# clean columns
nearby_venues.columns = [col.split(".")[-1] for col in nearby_venues.columns]

nearby_venues.head()
```

	name	categories	lat	lng
0	Santa Iglesia Catedral de Santa María la Real ...	Church	40.415767	-3.714516
1	Cervecería La Mayor	Beer Bar	40.415218	-3.712194
2	Plaza de La Almudena	Plaza	40.416320	-3.713777
3	Mercado Jamón Iberico	Market	40.415309	-3.711633
4	Taberna Rayuela	Tapas Restaurant	40.413179	-3.713496

[36]:

```
nearby_venues
```

	name	categories	lat	lng
0	Santa Iglesia Catedral de Santa María la Real ...	Church	40.415767	-3.714516
1	Cervecería La Mayor	Beer Bar	40.415218	-3.712194

	name	categories	lat	lng
2	Plaza de La Almudena	Plaza	40.416320	-3.713777
3	Mercado Jamón Iberico	Market	40.415309	-3.711633
4	Taberna Rayuela	Tapas Restaurant	40.413179	-3.713496
...	...	...	...	...
95	La Taquería de Birra	Mexican Restaurant	40.411605	-3.713569
96	Taquería Mi Ciudad	Mexican Restaurant	40.416927	-3.708488
97	Taberna del Capitán Alatriste	Spanish Restaurant	40.412623	-3.708385
98	La Mordida De Segovia	Mexican Restaurant	40.413766	-3.711112
99	Mi Ciudad I	Mexican Restaurant	40.417478	-3.708625

100 rows × 4 columns

And how many venues were returned by Foursquare?

[37]:

```
print('{} venues were returned by Foursquare.'.format(nearby_venues.shape[0]))
100 venues were returned by Foursquare.
```

Exploring all Neighborhoods in Madrid

Let's create a function to repeat the same process to all the neighborhoods in Madrid

[38]:

```
def getNearbyVenues(names, latitudes, longitudes, radius=500, limit=100):
    LIMIT=limit
    venues_list=[]
    for name, lat, lng in zip(names, latitudes, longitudes):
        print(name)

        # create the API request URL
        url = 
'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={},{}
&radius={}&limit={}'.format(
            CLIENT_ID, 
            CLIENT_SECRET, 
            VERSION, 
            lat, 
            lng,
```

```
        radius,
        LIMIT)

    # make the GET request
    results = requests.get(url).json()["response"]['groups'][0]['items']

    # return only relevant information for each nearby venue
    venues_list.append([(
        name,
        lat,
        lng,
        v['venue']['name'],
        v['venue']['location']['lat'],
        v['venue']['location']['lng'],
        v['venue']['categories'][0]['name']) for v in results])

nearby_venues = pd.DataFrame([item for venue_list in venues_list for item in venue_list])
nearby_venues.columns = ['Neighbourhood',
              'Neighbourhood Latitude',
              'Neighbourhood Longitude',
              'Venue',
              'Venue Latitude',
              'Venue Longitude',
              'Venue Category']

return(nearby_venues)
```

Now we write the code to run the above function on each neighborhood and create a new dataframe called *madrid_venues*.

[39]:

```
madrid_venues = getNearbyVenues(names=listadobarrios['Name'],
                                 latitudes=listadobarrios['latitude'],
                                 longitudes=listadobarrios['longitude']
                                )
```

Palacio (Madrid)
Embajadores (Madrid)
Cortes (Madrid)
Justicia (Madrid)
Malasaña
Sol (Madrid)
Imperial (Arganzuela)
Las Acacias
…
…
Amposta (Madrid)
Arcos (Madrid)
Rosas (Madrid)
Rejas (Madrid)
Canillejas (Madrid)
Salvador (Madrid)
Alameda de Osuna
Aeropuerto (Madrid)
Casco Histórico de Barajas
Timón (Madrid)
Corralejos

[40]:

```
madrid_venues
```

	Neighbourhood	Neighbourhood Latitude	Neighbourhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Palacio (Madrid)	40.415000	-3.713333	Santa Iglesia Catedral de Santa María la Real ...	40.415767	-3.714516	Church
1	Palacio (Madrid)	40.415000	-3.713333	Cervecería La Mayor	40.415218	-3.712194	Beer Bar
2	Palacio (Madrid)	40.415000	-3.713333	Plaza de La Almudena	40.416320	-3.713777	Plaza
3	Palacio (Madrid)	40.415000	-3.713333	Mercado Jamón Iberico	40.415309	-3.711633	Market
4	Palacio (Madrid)	40.415000	-3.713333	Taberna Rayuela	40.413179	-3.713496	Tapas Restaurant
...	...	...	...	...	...	...	...
3672	Corralejos	40.464444	-3.590000	Restaurante Asiático Hong Yun	40.462819	-3.591709	Asian Restaurant
3673	Corralejos	40.464444	-3.590000	Pizzamascalzone	40.465177	-3.592820	Pizza Place
3674	Corralejos	40.464444	-3.590000	Alimentación, pan, bebida y frutos secos	40.462925	-3.592227	Food & Drink Shop
3675	Corralejos	40.464444	-3.590000	Mercadona	40.465216	-3.592730	Supermarket
3676	Corralejos	40.464444	-3.590000	Farmacia Junquera	40.463779	-3.593902	Pharmacy

3677 rows × 7 columns

[41]:

```
madrid_venues.groupby('Neighbourhood').count()
```

	Neighbourhood Latitude	Neighbourhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
Neighbourhood						
Adelfas	36	36	36	36	36	36
Aeropuerto (Madrid)	19	19	19	19	19	19
Alameda de Osuna	26	26	26	26	26	26
Almagro (Madrid)	75	75	75	75	75	75
Almenara (Madrid)	4	4	4	4	4	4
...	...	...	...	...	...	...
Valverde (Madrid)	2	2	2	2	2	2
Ventas (Madrid)	21	21	21	21	21	21
Villaverde Alto (Madrid)	4	4	4	4	4	4
Vinateros	6	6	6	6	6	6
Zofío	5	5	5	5	5	5

129 rows × 6 columns

We save the dataframe in a file for later use.

[42]:

```
madrid_venues.to_csv('madrid_venues.csv',index=False)
```

Now we write the code to run the above function on each station and create a new dataframe called *madrid_station_venues*.

[43]:

```
madrid_station_venues = getNearbyVenues(names=stations['Name'],
                                   latitudes=stations['latitude'],
                                   longitudes=stations['longitude']
                                  )
```

Punto de recarga de Vehículos Eléctricos
Enchufauto
Recarga eléctrica de coches y motos
Recarga eléctrica de coches
Recarga eléctrica de coches
Punto de Carga para Vehiculo Eléctrico Sta. Engracia- José Abascal
Recarga eléctrica de coches y motos
Punto de Carga Vehículos Eléctricos
Punto de recarga Vehículo Eléctrico

[44]:

```
madrid_station_venues.to_csv('madrid_stations_venues.csv',index=False)
```

[45]:

```
madrid_station_venues.groupby('Neighbourhood').count()
```

	Neighbourhood Latitude	Neighbourhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
Neighbourhood						
Enchufauto	81	81	81	81	81	81
Punto de Carga Vehículos Eléctricos	14	14	14	14	14	14
Punto de Carga para Vehiculo Eléctrico Sta. Engracia- José Abascal	66	66	66	66	66	66
Punto de recarga Vehículo Eléctrico	100	100	100	100	100	100
Punto de recarga de Vehículos Eléctricos	100	100	100	100	100	100
Recarga eléctrica de coches	85	85	85	85	85	85
Recarga eléctrica de coches y motos	165	165	165	165	165	165

We already have all the data we need to carry out our study.

Part 2 : METHODOLOGY

We want to find the best locations for a new electric vehicle charging station using as a reference the similarity between each neighborhood and the environment of the points where a charging station already exists.

Analyzing the dataframe **madrid_station_venues** we will try to locate the parameters that define a good location.

Then, we will apply those parameters to the set of neighborhoods in Madrid (*madrid_venues*) to obtain its classification.

We load the data from the previously generated csv.

[46]:

```
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
%matplotlib inline
```

[47]:

```
madrid_station_venues = pd.read_csv('madrid_stations_venues.csv')
madrid_venues = pd.read_csv('madrid_venues.csv')
print(madrid_venues.head())
print(madrid_station_venues.head())
```

```
     Neighbourhood  Neighbourhood Latitude  Neighbourhood Longitude  \
0  Palacio (Madrid)                  40.415                -3.713333
1  Palacio (Madrid)                  40.415                -3.713333
2  Palacio (Madrid)                  40.415                -3.713333
3  Palacio (Madrid)                  40.415                -3.713333
4  Palacio (Madrid)                  40.415                -3.713333

                                          Venue  Venue Latitude  \
0  Santa Iglesia Catedral de Santa María la Real ...       40.415767
1                              Cervecería La Mayor       40.415218
2                              Plaza de La Almudena       40.416320
3                             Mercado Jamón Iberico       40.415309
4                                 Taberna Rayuela       40.413179

   Venue Longitude    Venue Category
0        -3.714516            Church
1        -3.712194          Beer Bar
2        -3.713777             Plaza
3        -3.711633            Market
4        -3.713496  Tapas Restaurant
                        Neighbourhood  Neighbourhood Latitude  \
0  Punto de recarga de Vehículos Eléctricos                40.43188
1  Punto de recarga de Vehículos Eléctricos                40.43188
2  Punto de recarga de Vehículos Eléctricos                40.43188
3  Punto de recarga de Vehículos Eléctricos                40.43188
4  Punto de recarga de Vehículos Eléctricos                40.43188

   Neighbourhood Longitude                                     Venue  \
0                -3.689155                                     Gucci
1                -3.689155                          Hotel Villa Magna
2                -3.689155  Museo de Escultura al Aire Libre de La Castellana
3                -3.689155                                    Cartier
4                -3.689155                    Supermercado El Corte Inglés

   Venue Latitude  Venue Longitude    Venue Category
```

0	40.430693	-3.687062	Boutique
1	40.429984	-3.687964	Hotel
2	40.433233	-3.688734	Sculpture Garden
3	40.430626	-3.686559	Jewelry Store
4	40.430061	-3.687861	Supermarket

We group the categories of venues to obtain features that define an environment.

[48]:

```
print('There are {} uniques categories in "Train" set.'.format(len(madrid_station_venues['Venue Category'].unique())))
print('There are {} uniques categories in "Prediction" set.'.format(len(madrid_venues['Venue Category'].unique())))
```

There are 139 uniques categories in "Train" set.
There are 274 uniques categories in "Prediction" set.

Analyze Each Charging Station

Next, let's group rows by neighborhood and by taking the mean of the frequency of occurrence of each category.
We also change the Nan by 0.

[49]:

```
station_grouped=madrid_station_venues.groupby(['Neighbourhood','Venue Category']).size().reset_index()
station_grouped.columns=['Neighbourhood','Venue Category','Count']
station_grouped=station_grouped.pivot(index='Neighbourhood', columns='Venue Category', values='Count')
station_grouped=station_grouped.fillna(0)
```

We repeat the same process to obtain the most common venues in each neighborhood.

[50]:

```
neighborhood_grouped=madrid_venues.groupby(['Neighbourhood','Venue Category']).size().reset_index()
neighborhood_grouped.columns=['Neighbourhood','Venue Category','Count']
neighborhood_grouped=neighborhood_grouped.pivot(index='Neighbourhood', columns='Venue Category', values='Count')
neighborhood_grouped=neighborhood_grouped.fillna(0)
```

We want to use the data set of the charging stations as a reference to identify the resulting categories after creating the clusters. To homogenize the results we will use the same columns in both data sets.

[51]:

```
columnsTrain=set(station_grouped.columns)
columnsTest=set(neighborhood_grouped.columns)
columnsComuns=list(columnsTrain.intersection(columnsTest))
print(columnsComuns)
```

['Arcade', 'Sports Club', 'Yoga Studio', 'Bistro', 'Sculpture Garden', 'Thai Restaurant', 'Pool', 'Dessert Shop', 'Exhibit', 'Breakfast Spot', 'Deli / Bodega', 'Building', 'Mexican Restaurant', 'Pub', 'Art Museum', 'Burger Joint', 'Theme Restaurant', 'Multiplex', 'Discount Store', 'Cajun / Creole Restaurant', 'Spanish Restaurant', 'Scenic Lookout', 'Italian Restaurant', 'Playground', 'Monument / Landmark', 'Brazilian Restaurant', 'Museum', 'Wine Bar', 'Seafood Restaurant', 'Hobby Shop', 'Japanese Restaurant', 'Office', 'Park', 'Gastropub', 'Peruvian Restaurant', 'Restaurant', 'Pizza Place', 'Coffee Shop', 'Paper / Office Supplies Store', 'Snack Place', 'Casino', 'Hotel', 'Historic Site', 'Public Art', 'Bookstore', 'Cocktail Bar', 'Steakhouse', 'Turkish Restaurant', 'Supermarket', 'Arepa Restaurant', 'Indian Restaurant', 'Paella Restaurant', 'Mobile Phone Shop', 'Coworking Space', 'Science Museum', 'Falafel Restaurant', 'Garden', 'Cafeteria', 'Pastry Shop', 'Department Store', 'History Museum', 'Shoe Store', 'Electronics Store', 'Optical Shop', 'Modern European Restaurant', 'Event Space', 'Motorcycle Shop', 'Mediterranean Restaurant', 'Juice Bar', 'Miscellaneous Shop', 'Sandwich Place', 'Sushi Restaurant', 'Diner', 'Salon / Barbershop', 'Gym', 'Spa', 'French Restaurant', 'Market', 'Used Bookstore', 'American Restaurant', 'Farmers Market', 'Boutique', 'Nightclub', 'Basketball Stadium', 'Grocery Store', 'Music Venue', 'Korean Restaurant', 'Fast Food Restaurant', 'Cheese Shop', 'Donut Shop', 'Argentinian Restaurant', 'Beer Bar', 'BBQ Joint', 'Church', 'Beer Garden', 'Asian Restaurant', 'Lounge', 'Bar', 'South American Restaurant', 'Art Gallery', 'Bakery', 'Middle Eastern Restaurant', 'Jewelry Store', 'Greek Restaurant', 'Bowling Alley', 'Dive Shop', 'Liquor Store', 'Theater', 'Plaza', 'Hostel', 'Gym / Fitness Center', 'Furniture / Home Store', 'Café', 'Ice Cream Shop', 'Vegetarian / Vegan Restaurant', 'New American Restaurant', 'Gourmet Shop', 'Pie Shop', 'Chinese Restaurant', 'Cosmetics Shop', 'Tapas Restaurant', 'Clothing Store', "Men's Store", 'General Entertainment', 'Brewery', 'Cupcake Shop', 'Gymnastics Gym', 'Burrito Place', 'Tea Room', 'Gift Shop']

[52]:

```
neighborhood_grouped=neighborhood_grouped[columnsComuns]
station_grouped=station_grouped[columnsComuns]
```

We have now two dataframes with identical columns ...

[53]:

```
neighborhood_grouped.head()
```

Venue Category	Arcade	Sports Club	Yoga Studio	Bistro	Sculpture Garden	Thai Restaurant	Pool	Dessert Shop	Exhibit	Breakfast Spot	...	Tapas Restaurant	Clothing Store	Men's Store	General Entertainment	Brewery	Cupcake Shop	Gymnastics Gym	Burrito Place	Tea Room	Gift Shop
Neighbourhood																					
Adelfas	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	2.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0
Aeropuerto (Mad	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Venue Category	Arcade	Sports Club	Yoga Studio	Bistro	Sculpture Garden	Thai Restaurant	Pool	Dessert Shop	Exhibit	Breakfast Spot	...	Tapas Restaurant	Clothing Store	Men's Store	General Entertainment	Brewery	Cupcake Shop	Gymnastics Gym	Burrito Place	Tea Room	Gift Shop
Neighbourhood																					
rid)																					
Alameda de Osuna	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	...	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Almagro (Madrid)	0.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0	...	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
Almenara (Madrid)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 130 columns

[54]:

```
station_grouped.head()
```

Venue Category	Arcade	Sports Club	Yoga Studio	Bistro	Sculpture Garden	Thai Restaurant	Pool	Dessert Shop	Exhibit	Breakfast Spot	...	Tapas Restaurant	Clothing Store	Men's Store	General Entertainment	Brewery	Cupcake Shop	Gymnastics Gym	Burrito Place	Tea Room	Gift Shop
Neighbourhood																					
Enchufauto	0.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0	2.0	...	5.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Punto de Carga Vehículos Eléctricos	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Punto de Carga para Vehicu	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	...	11.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0

Venue Category	Arcade	Sports Club	Yoga Studio	Bistro	Sculpture Garden	Thai Restaurant	Pool	Dessert Shop	Exhibit	Breakfast Spot	...	Tapas Restaurant	Clothing Store	Men's Store	General Entertainment	Brewery	Cupcake Shop	Gymnastics Gym	Burrito Place	Tea Room	Gift Shop
Neighbourhood																					
lo Eléctrico Sta. Engracia-José Abascal																					
Punto de recarga Vehículo Eléctrico	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	...	6.0	1.0	1.0	0.0	1.0	1.0	0.0	0.0	1.0	2.0

Venue Category	Arcade	Sports Club	Yoga Studio	Bistro	Sculpture Garden	Thai Restaurant	Pool	Dessert Shop	Exhibit	Breakfast Spot	...	Tapas Restaurant	Clothing Store	Men's Store	General Entertainment	Brewery	Cupcake Shop	Gymnastics Gym	Burrito Place	Tea Room	Gift Shop
Neighbourhood																					
Punto de recarga de Vehículos Eléctricos	0.0	1.0	0.0	2.0	1.0	1.0	0.0	0.0	0.0	0.0	...	4.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0

5 rows × 130 columns

Comparing reference with the raw data of each neighborhood.

Plot the number of venues as boxplots nearby charging station of electrical vehicle.

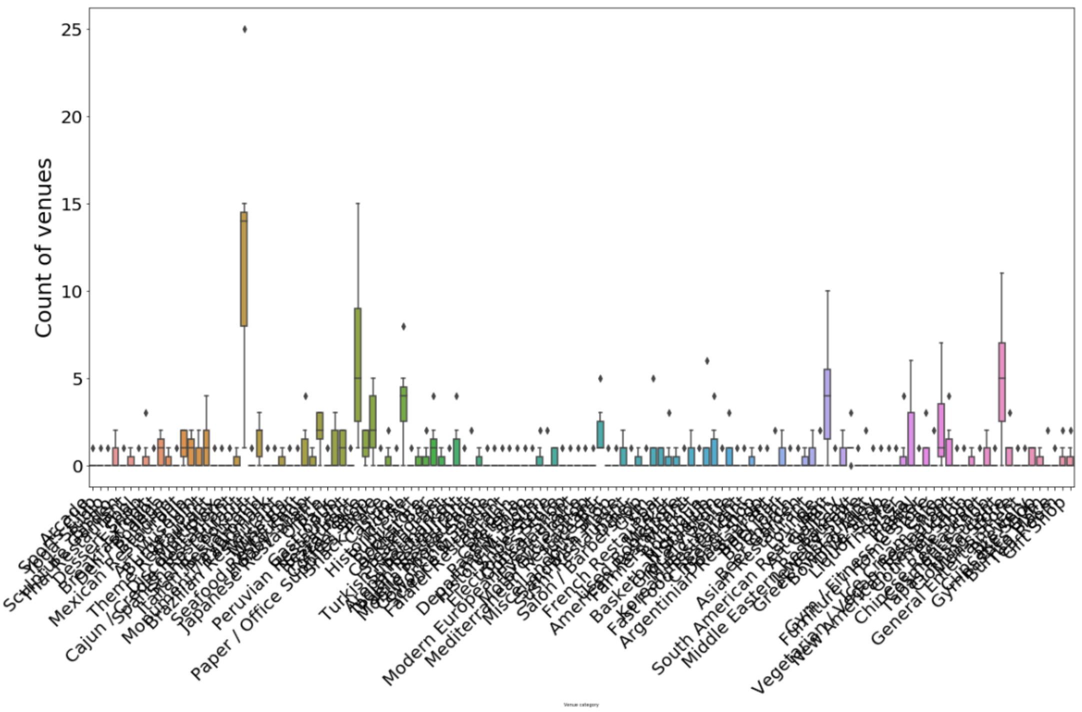
[55]:

```
plt.figure(figsize=(20, 10))
plt.xticks(rotation='vertical')
sns.boxplot

ax = sns.boxplot(data = station_grouped)
ax.set_ylabel('Count of venues', fontsize=25)

ax.set_xlabel('Venue category', fontsize=5)
ax.tick_params(labelsize=20)
plt.xticks(rotation=45, ha='right')
```

```
plt.show()
```



Plot the number of venues as boxplots nearby centroid neighbourhood.

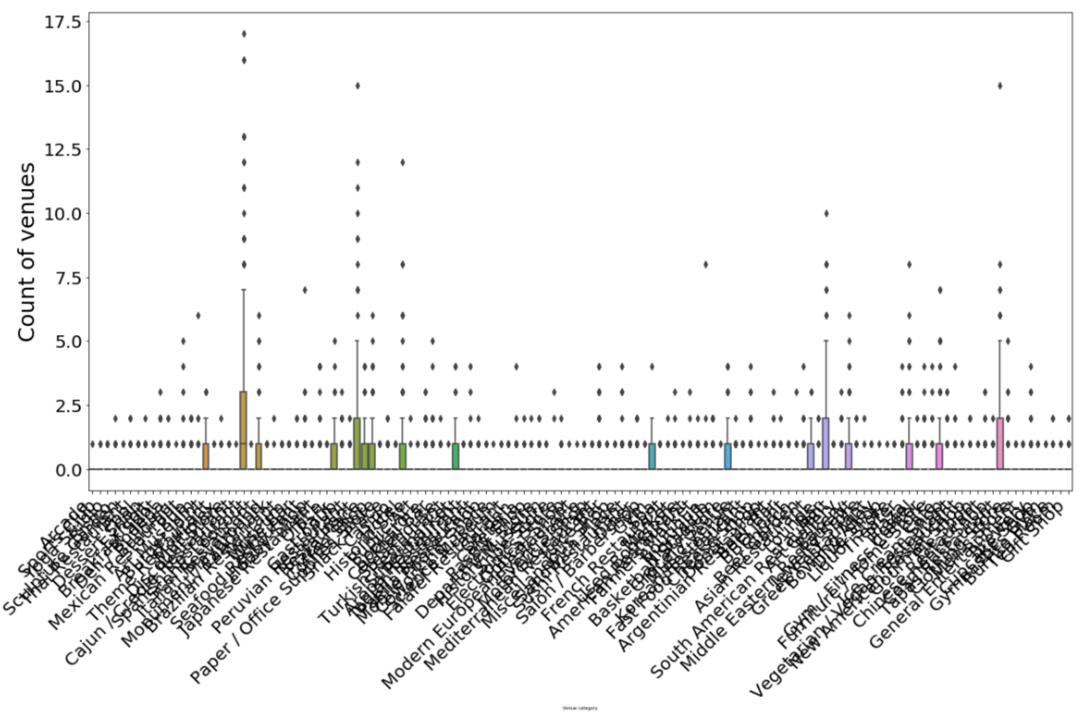
[56]:

```
plt.figure(figsize=(20, 10))
plt.xticks(rotation='vertical')
sns.boxplot

ax = sns.boxplot(data = neighborhood_grouped)
ax.set_ylabel('Count of venues', fontsize=25)

ax.set_xlabel('Venue category', fontsize=5)
ax.tick_params(labelsize=20)
plt.xticks(rotation=45, ha='right')

plt.show()
```



Looking at this second graph, we can see that only 18 categories appear with sufficient frequency and this categories also take the highest values nearby charging stations of electric vehicles.

We select this categories.

We calculate the average of each column and we are left with the 18 highest values.

[57]:

```
station_groupedH=station_grouped.mean(axis = 0, skipna = True).nlargest(18)
#neighborhood_grouped.nlargest(3, neighborhood_grouped.mean(axis = 1, skipna = True))
```

We convert the resulting series into a dataframe and we are left with the name of the most significant columns.

[58]:

```
df = station_groupedH.to_frame().reset_index()

columnasSignificativas=df['Venue Category'].values
```

We select only those columns and look at the resulting graphs, which now provide more detail.

For each charging station:

[59]:

```
station_MoreSignificant=station_grouped[columnasSignificativas]
```

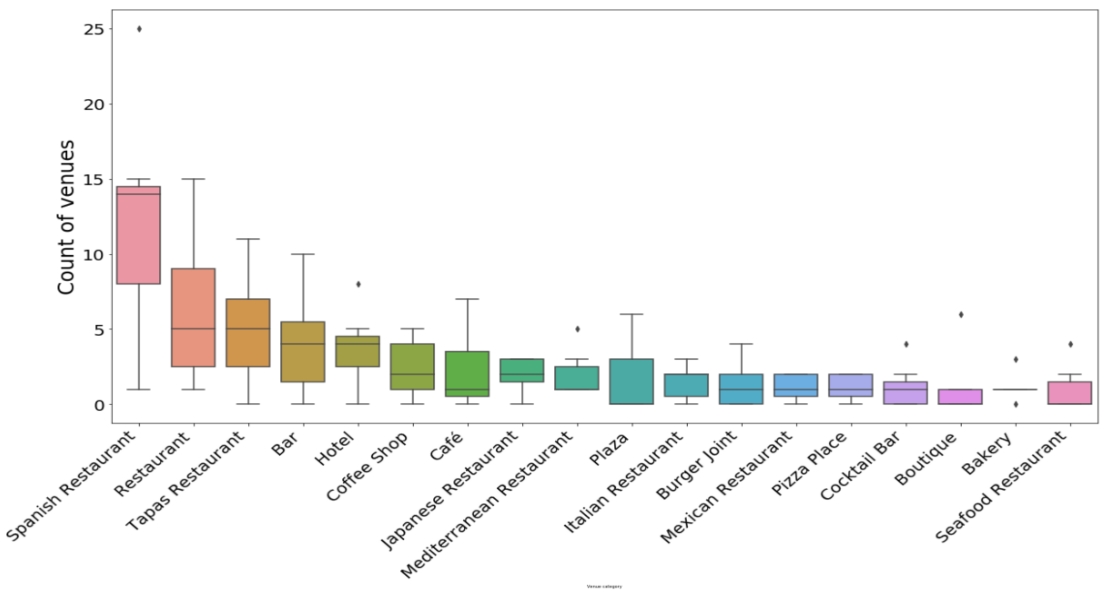
[60]:

```
plt.figure(figsize=(20, 10))
plt.xticks(rotation='vertical')
sns.boxplot

ax = sns.boxplot(data = station_MoreSignificant)
ax.set_ylabel('Count of venues', fontsize=25)

ax.set_xlabel('Venue category', fontsize=5)
ax.tick_params(labelsize=20)
plt.xticks(rotation=45, ha='right')

plt.show()
```



For each neighbourhood:

[61]:

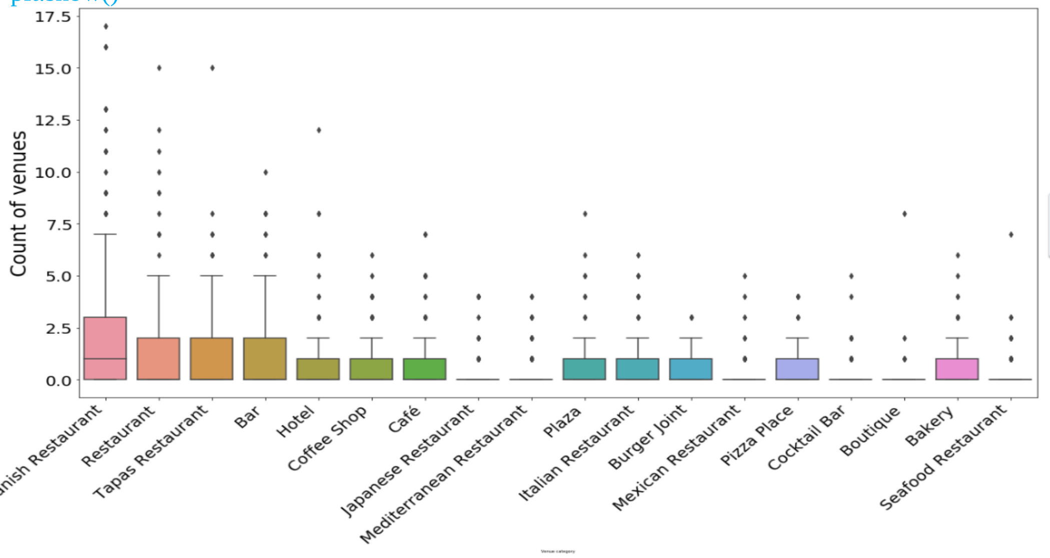
```
neighborhood_MoreSignificant=neighborhood_grouped[columnasSignificativas]
```

[62]:

```
plt.figure(figsize=(20, 10))
plt.xticks(rotation='vertical')
sns.boxplot

ax = sns.boxplot(data = neighborhood_MoreSignificant)
ax.set_ylabel('Count of venues', fontsize=25)
ax.set_xlabel('Venue category', fontsize=5)
ax.tick_params(labelsize=20)
plt.xticks(rotation=45, ha='right')

plt.show()
```



The location of the electric vehicle charging stations seems to be related to a greater presence of leisure and catering places. It seems logical, because near these places is where there is usually a lot of traffic.

Data preparation

Let's normalize the data using MinMaxScaler (scale from 0 to 1). This scales the data and provides an easy to interpret score at the same time.

[63]:

```
from sklearn.preprocessing import MinMaxScaler

X = neighborhood_MoreSignificant.values  #[:,4:]
cluster_dataset = MinMaxScaler().fit_transform(X)
```

[64]:

```
cluster_dataset
cluster_df = pd.DataFrame(cluster_dataset)
#cluster_df.columns = [c[0] for c in categories_list]
cluster_df
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	0.176471	0.066667	0.133333	0.4	0.166667	0.166667	0.142857	0.0	0.0	0.000	0.000000	0.333333	0.0	0.25	0.0	0.0	0.333333	0.0
1	0.117647	0.000000	0.000000	0.0	0.000000	0.333333	0.142857	0.0	0.0	0.000	0.000000	0.000000	0.0	0.00	0.0	0.0	0.000000	0.0
2	0.058824	0.066667	0.133333	0.1	0.166667	0.166667	0.000000	0.0	0.0	0.250	0.166667	0.000000	0.0	0.25	0.2	0.0	0.166667	0.0
3	0.529412	0.666667	0.066667	0.3	0.250000	0.166667	0.285714	0.0	0.5	0.500	0.666667	0.333333	0.0	0.00	0.2	0.0	0.333333	0.0
4	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.0	0.0	0.000	0.000000	0.000000	0.0	0.00	0.0	0.0	0.000000	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
124	0.000000	0.066667	0.000000	0.0	0.000000	0.000000	0.000000	0.0	0.0	0.000	0.000000	0.000000	0.0	0.00	0.0	0.0	0.000000	0.0
125	0.176471	0.000000	0.000000	0.2	0.166667	0.000000	0.000000	0.0	0.5	0.000	0.000000	0.000000	0.0	0.25	0.0	0.0	0.166667	0.0
126	0.000000	0.000000	0.000000	0.1	0.000000	0.000000	0.000000	0.0	0.0	0.000	0.000000	0.000000	0.0	0.00	0.0	0.0	0.000000	0.0
127	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.0	0.0	0.125	0.000000	0.000000	0.0	0.00	0.0	0.0	0.000000	0.0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
128	0.117647	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.0	0.0	0.000	0.000000	0.000000	0.0	0.00	0.0	0.0	0.000000	0.0

129 rows × 18 columns

[65]:

```
print(neighborhood_MoreSignificant.index)
Index(['Adelfas', 'Aeropuerto (Madrid)', 'Alameda de Osuna',
       'Almagro (Madrid)', 'Almenara (Madrid)', 'Almendrales', 'Aluche',
       'Amposta (Madrid)', 'Apóstol Santiago (Madrid)', 'Arapiles (Madrid)',
       ...
       'Valdefuentes (Madrid)', 'Valdemarín', 'Valderrivas', 'Valdezarza',
       'Vallehermoso (Madrid)', 'Valverde (Madrid)', 'Ventas (Madrid)',
       'Villaverde Alto (Madrid)', 'Vinateros', 'Zofío'],
      dtype='object', name='Neighbourhood', length=129)
```

[66]:

```
print(cluster_df.index)
#cluster_df=cluster_df0.reindex_like(neighborhood_MoreSignificant)
```

RangeIndex(start=0, stop=129, step=1)

2.1 Clustering

We'll be using k-means clustering.

[67]:

```
from sklearn.cluster import KMeans
```

We want to classify each neighborhood on a scale that goes from "The best" to "Bad", going through "Good" and "Regular". Therefore, we select 4 for the number of clusters.

[68]:

```
# set number of clusters
kclusters = 4

# run k-means clustering
kmeans = KMeans(n_clusters=kclusters, random_state=0).fit(cluster_df)

kmeans_labels = kmeans.labels_
```

[69]:

```
neighborhood_MoreSignificant_cluster= neighborhood_MoreSignificant.copy()
neighborhood_MoreSignificant_cluster['Cluster'] = kmeans_labels
neighborhood_MoreSignificant_cluster_minmax_df = cluster_df.copy()
neighborhood_MoreSignificant_cluster_minmax_df['Cluster'] = kmeans_labels
neighborhood_MoreSignificant_cluster_minmax_df['Neighbourhood']=neighborhood_MoreSignificant.index
neighborhood_MoreSignificant_cluster_minmax_df
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	Cluster	Neighbourhood
0	0.176471	0.066667	0.133333	0.4	0.166667	0.166667	0.142857	0.0	0.0	0.000	0.000000	0.333333	0.0	0.25	0.0	0.0	0.333333	0.0	1	Adelfas
1	0.117647	0.000000	0.000000	0.0	0.000000	0.333333	0.142857	0.0	0.0	0.000	0.000000	0.000000	0.0	0.00	0.0	0.0	0.000000	0.0	1	Aeropuerto (Madrid)
2	0.058824	0.066667	0.133333	0.1	0.166667	0.166667	0.000000	0.0	0.0	0.250	0.166667	0.000000	0.0	0.25	0.2	0.0	0.166667	0.0	1	Alameda de Osuna
3	0.529412	0.666667	0.066667	0.3	0.250000	0.166667	0.285714	0.0	0.5	0.500	0.666667	0.333333	0.0	0.00	0.2	0.0	0.333333	0.0	0	Almagro (Madrid)
4	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.0	0.0	0.000	0.000000	0.000000	0.0	0.00	0.0	0.0	0.000000	0.0	1	Almenara (Madrid)
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
124	0.000000	0.066667	0.000000	0.0	0.000000	0.000000	0.000000	0.0	0.0	0.000	0.000000	0.000000	0.0	0.00	0.0	0.0	0.000000	0.0	1	Valverde (Madrid)
125	0.176471	0.000000	0.000000	0.2	0.166667	0.000000	0.000000	0.0	0.5	0.000	0.000000	0.000000	0.0	0.25	0.0	0.0	0.166667	0.0	1	Ventas (Madrid)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	Cluster	Neighbourhood
126	0.000000	0.000000	0.000000	0.1	0.000000	0.000000	0.000000	0.0	0.0	0.000	0.000000	0.000000	0.0	0.00	0.0	0.0	0.000000	0.0	1	Villaverde Alto (Madrid)
127	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.0	0.0	0.125	0.000000	0.000000	0.0	0.00	0.0	0.0	0.000000	0.0	1	Vinateros
128	0.117647	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.0	0.0	0.000	0.000000	0.000000	0.0	0.00	0.0	0.0	0.000000	0.0	1	Zofío

129 rows × 20 columns

Visualize the clusters with boxplots

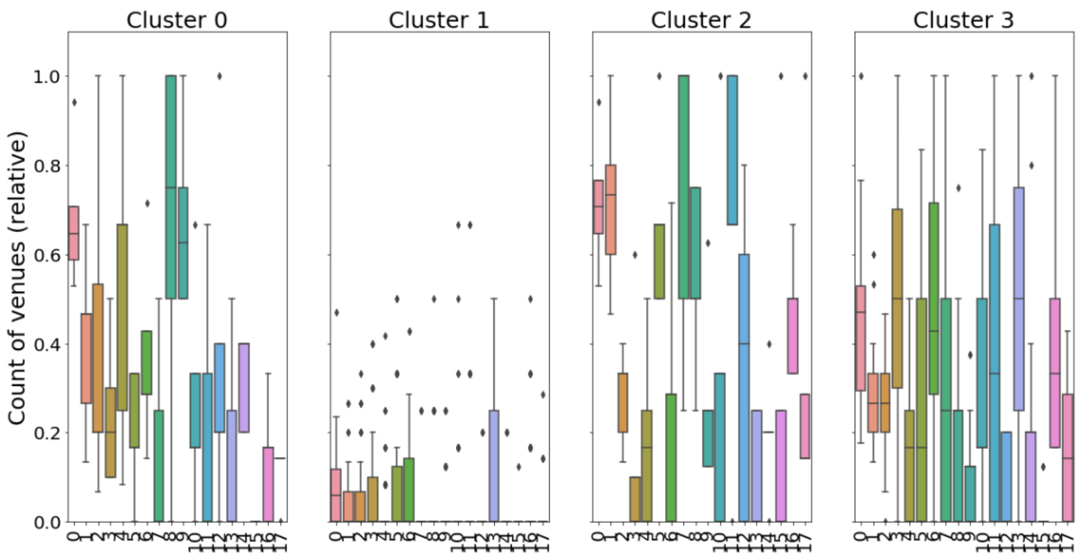
[70]:

```
import matplotlib.ticker as ticker

fig, axes = plt.subplots(1,kclusters, figsize=(20, 10), sharey=True)

axes[0].set_ylabel('Count of venues (relative)', fontsize=25)
#plt.set_xlabel('Venue category', fontsize='x-large')

for k in range(kclusters):
    #Set same y axis limits
    axes[k].set_ylim(0,1.1)
    axes[k].xaxis.set_label_position('top')
    axes[k].set_xlabel('Cluster ' + str(k), fontsize=25)
    axes[k].tick_params(labelsize=20)
    plt.sca(axes[k])
    plt.xticks(rotation='vertical')
    sns.boxplot(data = neighborhood_MoreSignificant_cluster_minmax_df[neighborhood_MoreSignificant_cluster_minmax_df['Cluster'] == k].drop('Cluster',1), ax=axes[k])
plt.show()
```



Comparing with the reference:

[71]:

```
Z = station_MoreSignificant.values  #[:,4:]
reference_dataset = MinMaxScaler().fit_transform(Z)
```

[72]:

```
reference_dataset
reference_df = pd.DataFrame(reference_dataset)
#cluster_df.columns = [c[0] for c in categories_list]
reference_df.head()
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	0.583333	0.428571	0.454545	0.5	0.500	1.0	0.000000	0.333333	0.5	0.000000	0.333333	0.50	0.0	1.0	0.0	0.166667	1.000000	1.0
1	0.000000	0.000000	0.000000	0.0	0.250	0.0	0.000000	0.000000	0.0	0.000000	0.000000	0.00	0.0	0.0	0.0	0.000000	0.000000	0.0
2	0.375000	0.142857	1.000000	0.6	0.000	0.0	0.428571	0.666667	0.0	0.000000	0.666667	0.00	1.0	0.5	0.0	0.000000	0.333333	0.0
3	0.208333	0.071429	0.545455	1.0	0.375	0.4	0.571429	0.666667	0.0	0.833333	0.000000	0.25	0.5	1.0	1.0	0.166667	0.333333	0.0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
4	0.541667	0.714286	0.363636	0.3	0.500	0.4	0.142857	1.000000	1.0	0.166667	0.666667	0.00	1.0	0.5	0.5	1.000000	0.333333	0.0

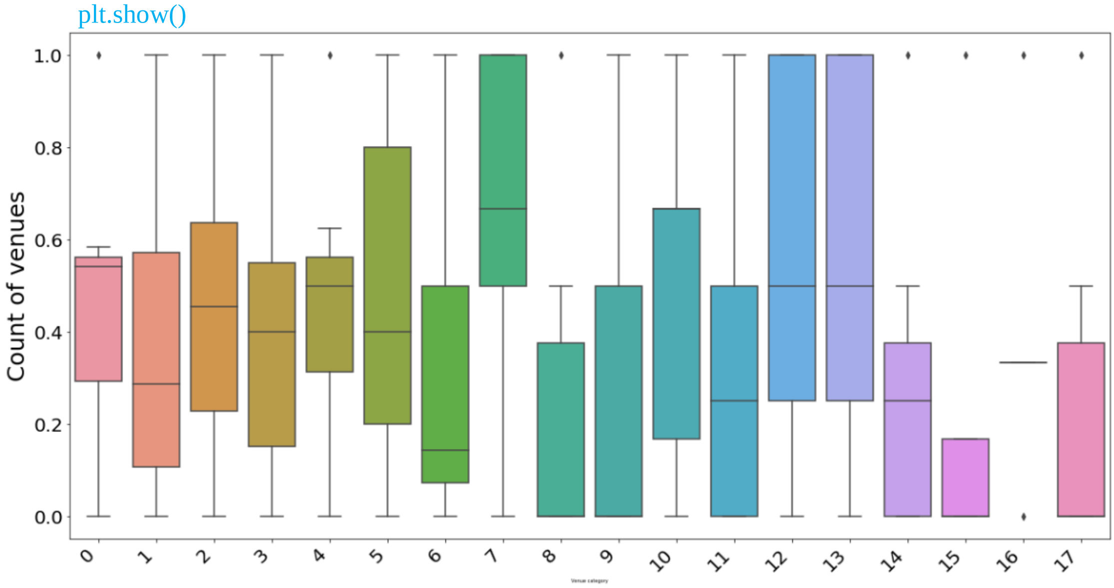
[73]:

```
plt.figure(figsize=(20, 10))
plt.xticks(rotation='vertical')
sns.boxplot

ax = sns.boxplot(data = reference_df)
ax.set_ylabel('Count of venues', fontsize=25)

ax.set_xlabel('Venue category', fontsize=5)
ax.tick_params(labelsize=20)
plt.xticks(rotation=45, ha='right')

plt.show()
```



[74]:

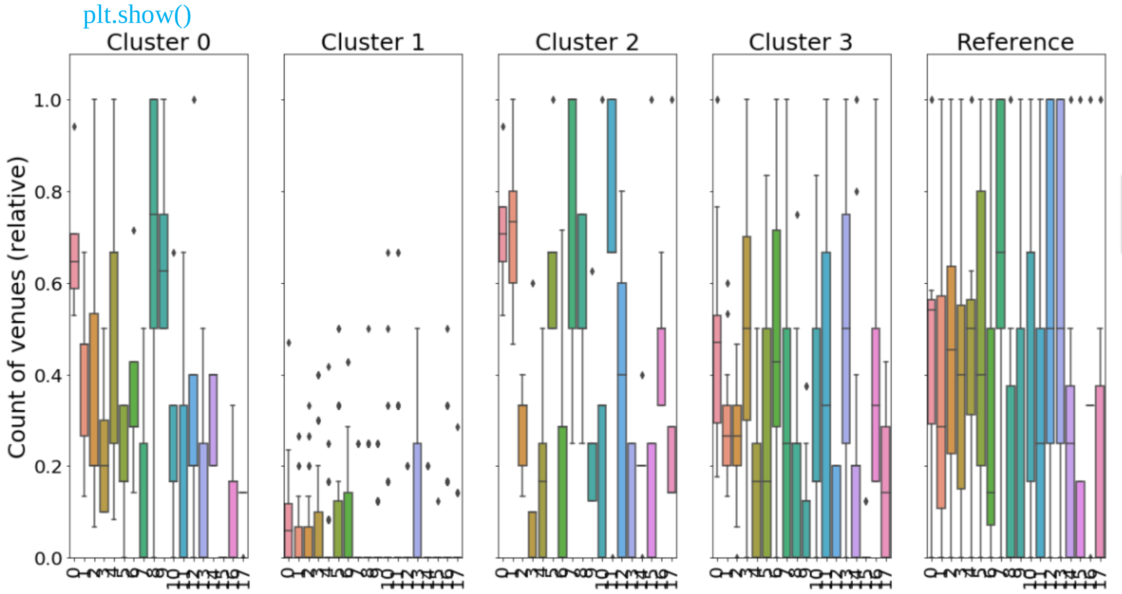
```
fig, axes = plt.subplots(1,kclusters + 1, figsize=(20, 10), sharey=True)

axes[0].set_ylabel('Count of venues (relative)', fontsize=25)
#plt.set_xlabel('Venue category', fontsize='x-large')

axes[4].set_ylim(0,1.1)
axes[4].xaxis.set_label_position('top')
axes[4].set_xlabel('Reference', fontsize=25)
axes[4].tick_params(labelsize=20)
plt.sca(axes[4])
plt.xticks(rotation='vertical')
sns.boxplot(data = reference_df, ax=axes[4])

for k in range(kclusters):
    #Set same y axis limits
    axes[k].set_ylim(0,1.1)
```

```
    axes[k].xaxis.set_label_position('top')
    axes[k].set_xlabel('Cluster ' + str(k), fontsize=25)
    axes[k].tick_params(labelsize=20)
    plt.sca(axes[k])
    plt.xticks(rotation='vertical')
    sns.boxplot(data = neighborhood_MoreSignificant_cluster_minmax_df[neighborhood_MoreSignificant_cluster_minmax_df['Cluster'] == k].drop('Cluster',1), ax=axes[k])
plt.show()
```



Visually we can observe that Cluster 0 is the one that presents the greatest similarity to the Reference. This will be the "Best" Category.

Next, Cluster 2 follows in similarity. This will be the "Good" category.

Cluster 3 has values well below the reference. This will be the "Regular" category.

Finally, Cluster 1 is the one that hardly matches the reference. This will be the "Bad" category

We rearrange the categories with this criterion:

[75]:

```
# Change label numbers so they go from highest scores to lowest
replace_labels = {0:1,1:3,2:0,3:2}
for i in range(len(kmeans_labels)):
    kmeans_labels[i] = replace_labels[kmeans_labels[i]]

neighborhood_MoreSignificant_cluster_minmax_df['Cluster'] = kmeans_labels
```

And we plot again:

[76]:

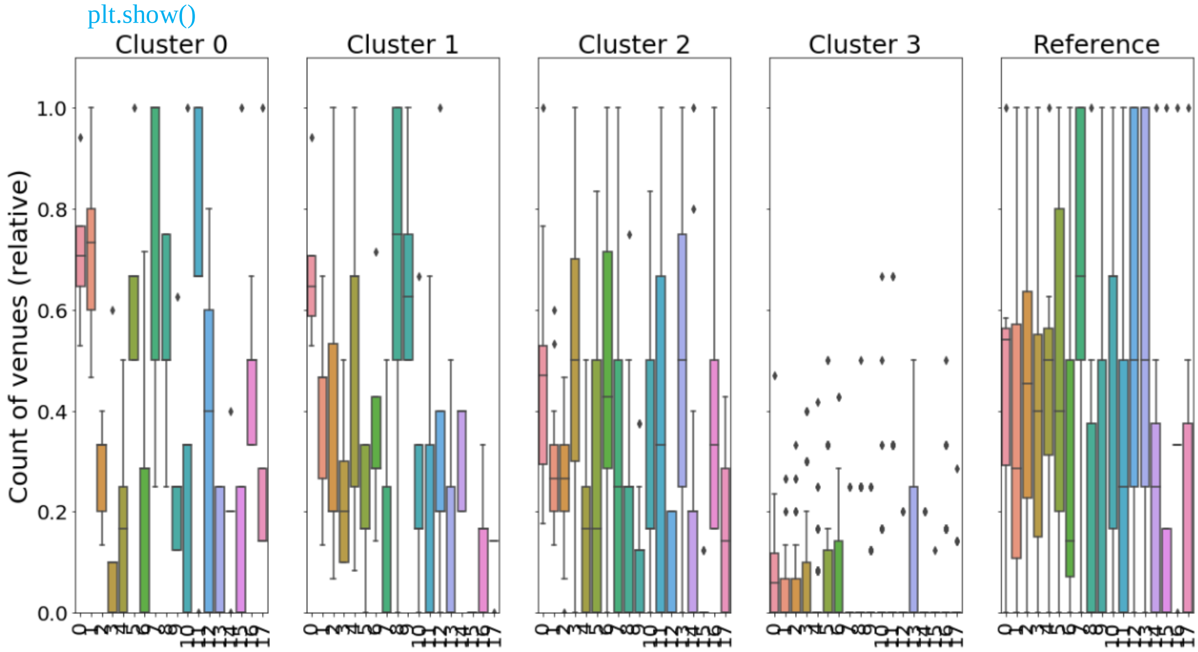
```
fig, axes = plt.subplots(1,kclusters + 1, figsize=(20, 10), sharey=True)

axes[0].set_ylabel('Count of venues (relative)', fontsize=25)
#plt.set_xlabel('Venue category', fontsize='x-large')

axes[4].set_ylim(0,1.1)
axes[4].xaxis.set_label_position('top')
axes[4].set_xlabel('Reference', fontsize=25)
```

```
axes[4].tick_params(labelsize=20)
plt.sca(axes[4])
plt.xticks(rotation='vertical')
sns.boxplot(data = reference_df, ax=axes[4])

for k in range(kclusters):
    #Set same y axis limits
    axes[k].set_ylim(0,1.1)
    axes[k].xaxis.set_label_position('top')
    axes[k].set_xlabel('Cluster ' + str(k), fontsize=25)
    axes[k].tick_params(labelsize=20)
    plt.sca(axes[k])
    plt.xticks(rotation='vertical')
    sns.boxplot(data =
neighborhood_MoreSignificant_cluster_minmax_df[neighborhood_MoreSignificant_cluster_minmax_df['Cluster'] == k].drop('Cluster',1), ax=axes[k])
plt.show()
```



Map the data.

[77]:

```
import folium

latitudMadrid=40.418889
longitudMadrid=-3.691944

#create map of madrid with all charging stations
map_madrid_cluster = folium.Map(location=[latitudMadrid, longitudMadrid], zoom_start=10)
```

[78]:

```
listado = pd.read_csv('barrios.csv')
listado.head()
```

	Name	link	Type	latitude	longitude
0	Centro (Madrid)	/wiki/Centro_(Madrid)	Distrito	40.415347	-3.707371
1	Palacio (Madrid)	/wiki/Palacio_(Madrid)	Barrio	40.415000	-3.713333
2	Embajadores (Madrid)	/wiki/Embajadores_(Madrid)	Barrio	40.408889	-3.699722
3	Cortes (Madrid)	/wiki/Cortes_(Madrid)	Barrio	40.414167	-3.698056
4	Justicia (Madrid)	/wiki/Justicia_(Madrid)	Barrio	40.423889	-3.696389

[79]:

```
listadobarrios=listado[listado['Type']=='Barrio'].reset_index()
listadoDistritos=listado[listado['Type']=='Distrito'].reset_index()
```

[80]:

```
import xml.etree.ElementTree as ET

pathtokml='estacion de carga de vehiculos electrico madrid.kml'
tree=ET.parse(pathtokml)
places=tree.findall('.//{http://www.opengis.net/kml/2.2}Placemark')
places[0].findall('.//{http://www.opengis.net/kml/2.2}coordinates')[0].text.split(',')[0]
places[0].findall('.//{http://www.opengis.net/kml/2.2}name')[0].text.split(',')[0]
elementos=[]
for place in places:

latitud=float(place.findall('.//{http://www.opengis.net/kml/2.2}coordinates')[0].text.split(',')[1])

longitud=float(place.findall('.//{http://www.opengis.net/kml/2.2}coordinates')[0].text.split(',')[0])
    nombre=place.findall('.//{http://www.opengis.net/kml/2.2}name')[0].text.split(',')[0]
    direccion=place.findall('.//{http://www.opengis.net/kml/2.2}address')[0].text.split(',')[0]
    elementos.append([nombre,direccion,latitud,longitud])

stations=pd.DataFrame(elementos)
stations.columns=['Name','Address','latitude','longitude']
```

stations

	Name	Address	latitude	longitude
0	Punto de recarga de Vehículos Eléctricos	28046 Madrid	40.431880	-3.689155
1	Enchufauto	Calle de Diego de León	40.434853	-3.678520
2	Recarga eléctrica de coches y motos	Calle de Alfonso XII	40.419270	-3.688927
3	Recarga eléctrica de coches	Paseo de la Castellana	40.458343	-3.689345
4	Recarga eléctrica de coches	Calle de los Chulapos	40.407397	-3.720168
5	Punto de Carga para Vehiculo Eléctrico Sta. En...	Calle de Sta Engracia	40.438250	-3.700276
6	Recarga eléctrica de coches y motos	Calle de Goya	40.424696	-3.671250
7	Punto de Carga Vehículos Eléctricos	Parking ifema	40.464751	-3.618176
8	Punto de recarga Vehículo Eléctrico	Calle de San Bernardo	40.427178	-3.706448

To represent the clusters on the map, we must first add the latitude and longitude of each neighborhood.

[81]:

neighborhood_MoreSignificant_cluster_minmax_df

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	Cluster	Neighbourhood
0	0.176471	0.066667	0.133333	0.4	0.166667	0.166667	0.142857	0.0	0.0	0.000	0.000000	0.333333	0.0	0.25	0.0	0.0	0.333333	0.0	3	Adelfas
1	0.117647	0.000000	0.000000	0.0	0.000000	0.333333	0.142857	0.0	0.0	0.000	0.000000	0.000000	0.0	0.00	0.0	0.0	0.000000	0.0	3	Aeropuerto (Madrid)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	Cluster	Neighbourhood
2	0.058824	0.066667	0.133333	0.1	0.166667	0.166667	0.000000	0.0	0.0	0.250	0.166667	0.000000	0.0	0.25	0.2	0.0	0.166667	0.0	3	Alameda de Osuna
3	0.529412	0.666667	0.066667	0.3	0.250000	0.166667	0.285714	0.0	0.5	0.500	0.666667	0.333333	0.0	0.00	0.2	0.0	0.333333	0.0	1	Almagro (Madrid)
4	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.0	0.0	0.000	0.000000	0.000000	0.0	0.00	0.0	0.0	0.000000	0.0	3	Almenara (Madrid)
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
124	0.000000	0.066667	0.000000	0.0	0.000000	0.000000	0.000000	0.0	0.0	0.000	0.000000	0.000000	0.0	0.00	0.0	0.0	0.000000	0.0	3	Valverde (Madrid)
125	0.176471	0.000000	0.000000	0.2	0.166667	0.000000	0.000000	0.0	0.5	0.000	0.000000	0.000000	0.0	0.25	0.0	0.0	0.166667	0.0	3	Ventas (Madrid)
126	0.000000	0.000000	0.000000	0.1	0.000000	0.000000	0.000000	0.0	0.0	0.000	0.000000	0.000000	0.0	0.00	0.0	0.0	0.000000	0.0	3	Villaverde Alto (Madrid)
127	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.0	0.0	0.1	0.000000	0.000000	0.0	0.00	0.0	0.0	0.000000	0.0	3	Vinateros

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	Cluster	Neighbourhood
										25										
128	0.117647	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.0	0.0	0.000	0.000000	0.000000	0.0	0.00	0.0	0.0	0.000000	0.0	3	Zofío

129 rows × 20 columns

Merging both dataframes:

[82]:

```
cluster_barrios=pd.merge(left=neighborhood_MoreSignificant_cluster_minmax_df,right=listadobarrios, left_on='Neighbourhood', right_on='Name')
```

We add a new column with a category label corresponding to each cluster.

[83]:

```
cluster_barrios.loc[cluster_barrios['Cluster']==0, 'Group']='The Best'
cluster_barrios.loc[cluster_barrios['Cluster']==1, 'Group']='Good'
cluster_barrios.loc[cluster_barrios['Cluster']==2, 'Group']='Regular'
cluster_barrios.loc[cluster_barrios['Cluster']==3, 'Group']='Bad'
#df1.loc[df1['stream'] == 2, 'feat'] = 10
cluster_barrios
```

	0	1	2	3	4	5	6	7	8	9	...	17	Cluster	Neighbourhood	index	Name	link	Type	latitude	longitude	Group
0	0.176471	0.066667	0.133333	0.4	0.166667	0.166667	0.142857	0.0	0.0	0.000	...	0.0	3	Adelfas	17	Adelfas	/wiki/Adelfas	Barrio	40.400278	-3.670833	Bad
1	0.1176	0.0000	0.0000	0.0	0.0000	0.3333	0.1428	0.0	0.0	0.000	...	0.0	3	Aeropuerto (M	147	Aeropuerto	/wiki/Aeropuerto_(Madrid)	Barr	40.4941	-3.5669	Bad

	0	1	2	3	4	5	6	7	8	9	...	17	Cluster	Neighbourhood	index	Name	link	Type	latitude	longitude	Group
	47	00	00		00	33	57							adrid)		(Madrid)		io	67	44	
2	0.058824	0.066667	0.133333	0.1	0.166667	0.166667	0.000000	0.0	0.0	0.250	...	0.0	3	Alameda de Osuna	146	Alameda de Osuna	/wiki/Alameda_de_Osuna	Barrio	40.457222	-3.587778	Bad
3	0.529412	0.666667	0.066667	0.3	0.250000	0.166667	0.285714	0.0	0.5	0.500	...	0.0	1	Almagro (Madrid)	47	Almagro (Madrid)	/wiki/Almagro_(Madrid)	Barrio	40.431667	-3.694167	Good
4	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.0	0.0	0.000	...	0.0	3	Almenara (Madrid)	40	Almenara (Madrid)	/wiki/Almenara_(Madrid)	Barrio	40.471290	-3.695505	Bad

	0	1	2	3	4	5	6	7	8	9	...	17	Cluster	Neighbourhood	index	Name	link	Type	latitude	longitude	Group
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
124	0.000000	0.066667	0.000000	0.0	0.000000	0.000000	0.000000	0.0	0.0	0.000	...	0.0	3	Valverde (Madrid)	56	Valverde (Madrid)	/wiki/Valverde_(Madrid)	Barrio	40.505833	-3.685833	Bad
125	0.176471	0.000000	0.000000	0.2	0.166667	0.000000	0.000000	0.0	0.5	0.000	...	0.0	3	Ventas (Madrid)	105	Ventas (Madrid)	/wiki/Ventas_(Madrid)	Barrio	40.428333	-3.657500	Bad
126	0.000000	0.000000	0.000000	0.1	0.000000	0.000000	0.000000	0.0	0.0	0.000	...	0.0	3	Villaverde Alto (Madrid)	122	Villaverde Alto (Madrid)	/wiki/Villaverde_Alto_(Madrid)	Barrio	40.347778	-3.711667	Bad
127	0.000	0.000	0.000	0.0	0.000	0.000	0.000	0.0	0.0	0.1	...	0.0	3	Vinateros	103	Vinater	/wiki/Vinateros	Barr	40.403	-3.640	Bad

	0	1	2	3	4	5	6	7	8	9	...	17	Cluster	Neighbourhood	index	Name	link	Type	latitude	longitude	Group
	000	000	000		000	000	000			25						os		io	056	278	
128	0.117647	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.0	0.0	0.000	...	0.0	3	Zofío	88	Zofío	/wiki/Zof%C3%ADo	Barrio	40.379167	-3.715278	Bad

129 rows × 27 columns

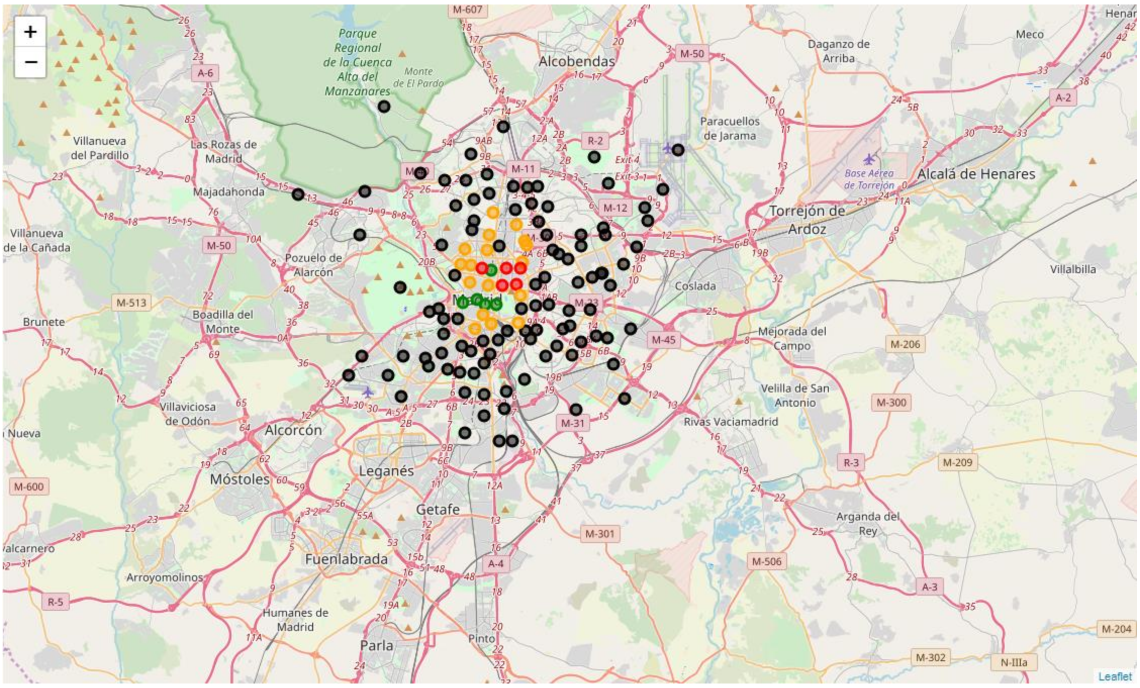
And finally, we place each point on the map with a color code:

1. The Best: Red
2. Good: Green
3. Regular: orange
4. Bad: black

[84]:

```
#add markers
for i, Neighbourhood, latitude, longitude, cluster, Group in zip(cluster_barrios.index,
                                    cluster_barrios['Neighbourhood'],
                                    cluster_barrios['latitude'],
                                    cluster_barrios['longitude'],
                                    cluster_barrios['Cluster'],
                                    cluster_barrios['Group']):
  #colors = sns.color_palette(None, kclusters).as_hex()
   colors=['red','green','orange','black']
   station_series = cluster_barrios.iloc[i]
   popup='<b>{}</b><br>Cluster {}'.format(
       Neighbourhood,
       Group
       )
   folium.CircleMarker(
       location=[latitude,longitude],
       fill=True,
       fill_opacity=0.5,
       popup=folium.Popup(popup, max_width = 300),
       radius=5,
       color=colors[cluster]
   ).add_to(map_madrid_cluster)


map_madrid_cluster
```



Results

Here is how we can characterize the clusters by looking at venue scores

- Cluster 0 (Red) has consistently high scores for all venue categories as seen in the reference pattern of the existing charging stations.
- Cluster 1 (Green) has lower score for all venue categories than previos cluster.
- Cluster 2 (Orange) has lowest marks for all venue categories.
- Cluster 3 (Red) barely coincide with the selected venue categories.

Cluster 0 is the best option because it its the most similar to the current locations of the selected charging stations.

Plotting the clusters on a map shows us that

- Cluster 0 corresponds to the large commercial districts of the city's commercial center.
- Group 1 is also in the center but in neighborhoods with smaller capacity roads.
- Groups 2 and 3 correspond to more peripheral neighborhoods.

Discussion

To be fair, Foursquare data does not cover everything. The largest number of places are in the categories of Food, shops and services.

In a more complete study, other indicators of the movement of people and vehicles should be included, but as a first approximation the results appear to be consistent.

Conclusion

In this study, we have taken data on the environment of electric vehicle charging stations. They have been used as a pattern of comparison with the neighborhoods of Madrid, where there are no facilities of this type to establish a division of the city in areas where it is more interesting to install a new charging station.

Four groups have been established to classify the neighborhoods from the best area to the worst, and the geographic results show a great correlation with the current location of the existing facilities, so the validity of this method for locating suitable locations is accepted because wecan answer the questions we had raised:

- What is best location in Madrid for stations charging electrical vehicles? The Cluster 0 o "The Best" locations.
- Which areas have potential station charging electrical vehicles market? The Cluster 1 or "Good" locations.

Additional socio-economic factors that could better characterize neighborhoods have not been taken into account. Future developments in this field may provide more accurate results.