IBM Data Science Course

Coursera Capstone Project

The battle of neighborhoods

Stefano Magarotto

Step 0: install required libraries

In [152]:

```
import numpy as np # library to handle data in a vectorized manner
import time
import pandas as pd # library for data analsysis
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
import json # library to handle JSON files
import requests # library to handle requests
from pandas.io.json import json_normalize # tranform JSON file into a pandas dataframe
from geopy.geocoders import Nominatim # convert an address into latitude and longitude values
import folium # map rendering library
import folium # map rendering library
from folium import plugins

# Matplotlib and associated plotting modules
import matplotlib.cm as cm
import matplotlib.colors as colors

import seaborn as sns

# import k-means from clustering stage
from sklearn.cluster import KMeans

print('All libraries imported.')
```

```
All libraries imported.
```

Step 1: Padova/Padua

Padova/Padua: residence location and venues (source: FourSquare)

In [153]:

```
# Via Manzoni, Padova, Italy
address = 'Via Manzoni, Padova, Italy'
geolocator = Nominatim(user_agent="thebofn")
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude
neighborhood_latitude=45.3932529
neighborhood_longitude=11.8799238
print('The geograpical coordinates of my Padova home are {}, {}.'.format(neighborhood_latitude, neighborhood_longitude))
```

```
The geograpical coordinates of my Padova home are 45.3932529, 11.8799238.
```

1.1 We start with parks and playground

In [154]:

```
CLIENT_ID = 'DHQVKXTSH4R2LAKLZTI3NS5ZGHUQOBXLPD4SSEYJYSUYJ5KF' # your Foursquare ID
CLIENT_SECRET = 'WM3FT0GHS5SFU2FME1JCFJARPJQJLXXEJP2LVUMDKLUX0PHB' # your Foursquare Secret
VERSION = '20190801' # Foursquare API version
LIMIT = 100 # limit of number of venues returned by Foursquare API
radius = 1000 # define radius
# CATEGORIES I AM INTERESTED ARE (TO FILTER THE SEARCH)
# PARK 4bf58dd8d48988d163941735,5bae9231bedf3950379f89d0,52e81612bcbc57f1066b7a21,52e81612bcbc57f1066b7a13,4bf58dd8d48988d162941735
# PLAYGROUND 4bf58dd8d48988d1e7941735
categoryId='4bf58dd8d48988d163941735,5bae9231bedf3950379f89d0,52e81612bcbc57f1066b7a21,52e81612bcbc57f1066b7a13,4bf58dd8d48988d162941735'

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

Out[154]:

```
'https://api.foursquare.com/v2/venues/explore?&client_id=DHQVKXTSH4R2LAKLZTI3N
S5ZGHUQOBXLPD4SSEYJYSUYJ5KF&client_secret=WM3FT0GHS5SFU2FME1JCFJARPJQJLXXEJP2L
VUMDKLUX0PHB&v=20190801&ll=45.3932529,11.8799238&categoryId=4bf58dd8d48988d163
941735,5bae9231bedf3950379f89d0,52e81612bcbc57f1066b7a21,52e81612bcbc57f1066b7
a13,4bf58dd8d48988d162941735&radius=1000&limit=100'
```

In [155]:

```
results = requests.get(url).json()
#results (to keep clean I commented it)
```

In [156]:

```
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']
```

In [157]:

```
venues = results['response']['groups'][0]['items']
PDnearby_venues = json_normalize(venues) # flatten JSON
# filter columns
filtered_columns = ['venue.name', 'venue.categories', 'venue.location.lat', 'venue.location.lng']
PDnearby_venues = PDnearby_venues.loc[:, filtered_columns]
# filter the category for each row
PDnearby_venues['venue.categories'] = PDnearby_venues.apply(get_category_type, axis=1)
# clean columns
PDnearby_venues.columns = [col.split(".")[-1] for col in PDnearby_venues.columns]

PDnearby_venues.shape
```

Out[157]:

```
(5, 4)
```

In [158]:

```
# Venues near my current Padova/Padua home
PDnearby_venues['categories'].value_counts()
```

Out[158]:

```
Other Great Outdoors    2
Park                    1
Garden                  1
Plaza                   1
Name: categories, dtype: int64
```

In [176]:

```
PDnearby_venues
```

Out[176]:

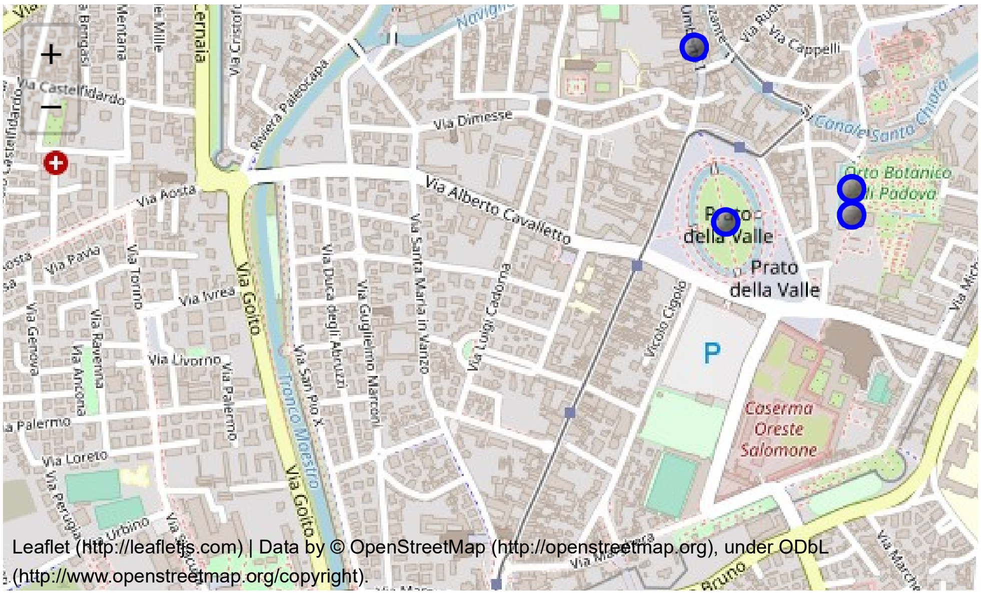
	name	categories	lat	lng
0	Prato della Valle	Plaza	45.398420	11.876521
1	Orto Botanico (Orto botanico di Padova)	Garden	45.399007	11.879603
2	Centro Giovanile Antonianum	Other Great Outdoors	45.398539	11.879618
3	Giardini Santa Rita	Park	45.391702	11.888398
4	Via Umberto I	Other Great Outdoors	45.401423	11.875735

Map of my Padova/Padua home with parks and playgrounds in Neighborhood

In [177]:

```
map_pd = folium.Map(location=[neighborhood_latitude, neighborhood_longitude], z
oom_start=15)
# add markers to map
for lat, lng, label in zip(PDnearby_venues['lat'], PDnearby_venues['lng'], PDne
arby_venues['name']):
    label = folium.Popup(label, parse_html=True)
    folium.RegularPolygonMarker(
        [lat, lng],
        number_of_sides=30,
        radius=7,
        popup=label,
        color='blue',
        fill_color='#0f0f0f',
        fill_opacity=0.6,
    ).add_to(map_pd)

map_pd
```

Out[177]:



Leaflet (http://leafletjs.com) | Data by © OpenStreetMap (http://openstreetmap.org), under ODbL (http://www.openstreetmap.org/copyright).

1.2 We continue with schools (primary, middle and high schools)

In [161]:

```
CLIENT_ID = 'DHQVKXTSH4R2LAKLZTI3NS5ZGHUQOBXLPD4SSEYJYSUYJ5KF' # your Foursquar
e ID
CLIENT_SECRET = 'WM3FT0GHS5SFU2FME1JCFJARPJQJLXXEJP2LVUMDKLUX0PHB' # your Fours
quare Secret
VERSION = '20190801' # Foursquare API version
LIMIT = 100 # limit of number of venues returned by Foursquare API
radius = 1000 # define radius
# CATEGORIES I AM INTERESTED ARE (TO FILTER THE SEARCH)
# SCHOOLS Primary 4f4533804b9074f6e4fb0105, Middle 4f4533814b9074f6e4fb0106, Hi
gh 4bf58dd8d48988d13d941735
categoryId='4f4533804b9074f6e4fb0105,4f4533814b9074f6e4fb0106,4bf58dd8d48988d13
d941735'

# create URL
url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secre
t={}&v={}&ll={},{}&categoryId={}&radius={}&limit={}'.format(
    CLIENT_ID,
    CLIENT_SECRET,
    VERSION,
    neighborhood_latitude,
    neighborhood_longitude,
    categoryId,
    radius,
    LIMIT)
url # display URL
```

Out[161]:

```
'https://api.foursquare.com/v2/venues/explore?&client_id=DHQVKXTSH4R2LAKLZTI3N
S5ZGHUQOBXLPD4SSEYJYSUYJ5KF&client_secret=WM3FT0GHS5SFU2FME1JCFJARPJQJLXXEJP2L
VUMDKLUX0PHB&v=20190801&ll=45.3932529,11.8799238&categoryId=4f4533804b9074f6e4
fb0105,4f4533814b9074f6e4fb0106,4bf58dd8d48988d13d941735&radius=1000&limit=100
'
```

In [162]:

```
results = requests.get(url).json()
#results (to keep clean I commented it)
venues = results['response']['groups'][0]['items']
PDnearby_venues2 = json_normalize(venues) # flatten JSON
# filter columns
filtered_columns = ['venue.name', 'venue.categories', 'venue.location.lat', 've
nue.location.lng']
PDnearby_venues2 = PDnearby_venues2.loc[:, filtered_columns]
# filter the category for each row
PDnearby_venues2['venue.categories'] = PDnearby_venues2.apply(get_category_typ
e, axis=1)
# clean columns
PDnearby_venues2.columns = [col.split(".")[-1] for col in PDnearby_venues2.colu
mns]

PDnearby_venues2.shape
```

Out[162]:

```
(3, 4)
```

In [163]:

```
PDnearby_venues2
```

Out[163]:

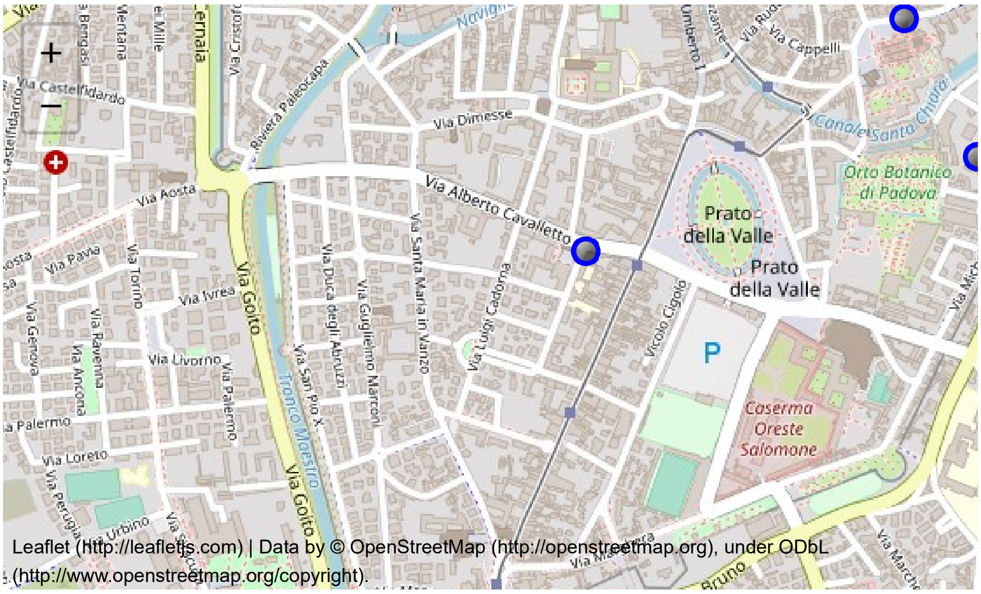
	name	categories	lat	lng
0	Itis Ruzza	High School	45.399522	11.882700
1	Liceo Scientifico E. Fermi	High School	45.397916	11.873107
2	ITC Calvi	High School	45.401920	11.880880

Map of my Padova/Padua home with schools in Neighborhood

In [164]:

```
map_pd = folium.Map(location=[neighborhood_latitude, neighborhood_longitude], z
oom_start=15)
# add markers to map
for lat, lng, label in zip(PDnearby_venues2['lat'], PDnearby_venues2['lng'], PD
nearby_venues2['name']):
    label = folium.Popup(label, parse_html=True)
    folium.RegularPolygonMarker(
        [lat, lng],
        number_of_sides=30,
        radius=7,
        popup=label,
        color='blue',
        fill_color='#0f0f0f',
        fill_opacity=0.6,
    ).add_to(map_pd)

map_pd
```

Out[164]:



1.3 We continue with shopping venues

In [165]:

```
CLIENT_ID = 'DHQVKXTSH4R2LAKLZTI3NS5ZGHUQOBXLPD4SSEYJYSUYJ5KF' # your Foursquare ID
CLIENT_SECRET = 'WM3FT0GHS5SFU2FME1JCFJARPJQJLXXEJP2LVUMDKLUX0PHB' # your Foursquare Secret
VERSION = '20190801' # Foursquare API version
LIMIT = 100 # limit of number of venues returned by Foursquare API
radius = 1000 # define radius
# CATEGORIES I AM INTERESTED ARE (TO FILTER THE SEARCH)
# SHOPPING 4d4b7105d754a06378d81259
categoryId='4d4b7105d754a06378d81259'

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

Out[165]:

```
'https://api.foursquare.com/v2/venues/explore?&client_id=DHQVKXTSH4R2LAKLZTI3NS5ZGHUQOBXLPD4SSEYJYSUYJ5KF&client_secret=WM3FT0GHS5SFU2FME1JCFJARPJQJLXXEJP2LVUMDKLUX0PHB&v=20190801&ll=45.3932529,11.8799238&categoryId=4d4b7105d754a06378d81259&radius=1000&limit=100'
```

In [166]:

```
results = requests.get(url).json()
#results (to keep clean I commented it)
venues = results['response']['groups'][0]['items']
PDnearby_venues3 = json_normalize(venues) # flatten JSON
# filter columns
filtered_columns = ['venue.name', 'venue.categories', 'venue.location.lat', 'venue.location.lng']
PDnearby_venues3 = PDnearby_venues3.loc[:, filtered_columns]
# filter the category for each row
PDnearby_venues3['venue.categories'] = PDnearby_venues3.apply(get_category_type, axis=1)
# clean columns
PDnearby_venues3.columns = [col.split(".")[-1] for col in PDnearby_venues3.columns]

PDnearby_venues3.shape
```

Out[166]:

```
(14, 4)
```

In [178]:
```
PDnearby_venues3
```

Out[178]:

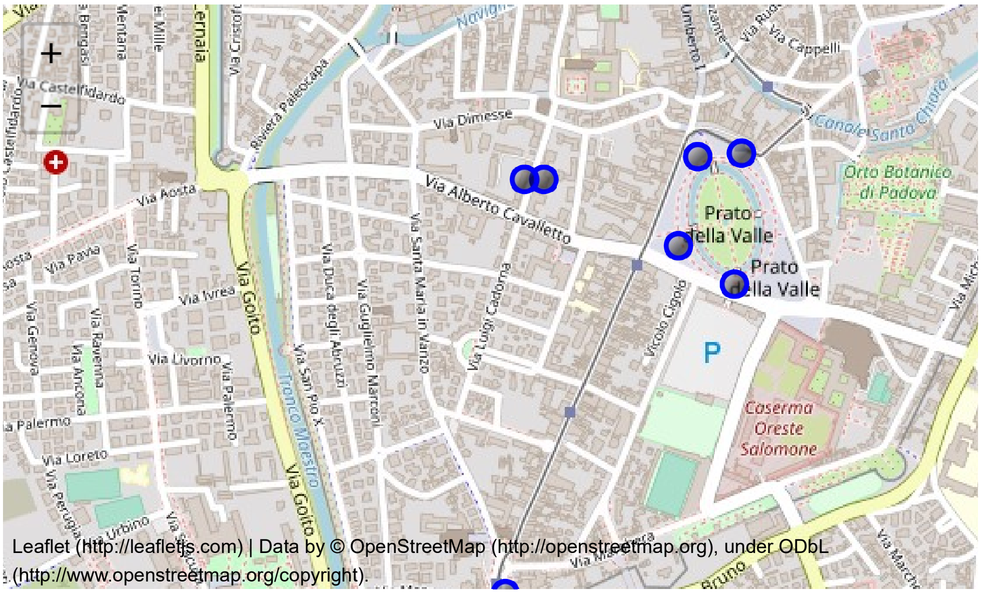
	name	categories	lat	lng
0	Mercato di Prato della Valle	Market	45.397334	11.876730
1	3Store	Mobile Phone Shop	45.389453	11.880090
2	Farmacia Ciato	Pharmacy	45.389220	11.880160
3	DESPAR Vergerio	Grocery Store	45.395713	11.885286
4	Gioielleria Cattelan	Jewelry Store	45.388406	11.880976
5	Bottega Angoli di Mondo Coop. Sociale	Thrift / Vintage Store	45.389103	11.875822
6	Europa in Prato	Food & Drink Shop	45.397989	11.875354
7	Calore Piante	Flower Shop	45.388128	11.884997
8	Pam Panorama	Supermarket	45.391990	11.871113
9	Drogheria Preti	Liquor Store	45.399604	11.876897
10	Mercato Antiquariato Prato Della Valle	Flea Market	45.399544	11.875821
11	Conad City	Supermarket	45.393849	11.890307
12	Despar	Food & Drink Shop	45.399133	11.872046
13	Sara Assicurazioni - Agenzia di Padova Sud	Insurance Office	45.399148	11.871606

Map of my Padova/Padua home with shopping venues in Neighborhood

In [168]:

```
map_pd = folium.Map(location=[neighborhood_latitude, neighborhood_longitude], z
oom_start=15)
# add markers to map
for lat, lng, label in zip(PDnearby_venues3['lat'], PDnearby_venues3['lng'], PD
nearby_venues3['name']):
    label = folium.Popup(label, parse_html=True)
    folium.RegularPolygonMarker(
        [lat, lng],
        number_of_sides=30,
        radius=7,
        popup=label,
        color='blue',
        fill_color='#0f0f0f',
        fill_opacity=0.6,
    ).add_to(map_pd)

map_pd
```

Out[168]:



1.4 Finally arts and entertainment venues

In [169]:

```
CLIENT_ID = 'DHQVKXTSH4R2LAKLZTI3NS5ZGHUQOBXLPD4SSEYJYSUYJ5KF' # your Foursquare ID
CLIENT_SECRET = 'WM3FT0GHS5SFU2FME1JCFJARPJQJLXXEJP2LVUMDKLUX0PHB' # your Foursquare Secret
VERSION = '20190801' # Foursquare API version
LIMIT = 100 # limit of number of venues returned by Foursquare API
radius = 1000 # define radius
# CATEGORIES I AM INTERESTED ARE (TO FILTER THE SEARCH)
# # ARTS AND THEATRES 4d4b7104d754a06370d81259
categoryId='4d4b7104d754a06370d81259'

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

Out[169]:

```
'https://api.foursquare.com/v2/venues/explore?&client_id=DHQVKXTSH4R2LAKLZTI3NS5ZGHUQOBXLPD4SSEYJYSUYJ5KF&client_secret=WM3FT0GHS5SFU2FME1JCFJARPJQJLXXEJP2LVUMDKLUX0PHB&v=20190801&ll=45.3932529,11.8799238&categoryId=4d4b7104d754a06370d81259&radius=1000&limit=100'
```

In [170]:

```
results = requests.get(url).json()
#results (to keep clean I commented it)
venues = results['response']['groups'][0]['items']
PDnearby_venues4 = json_normalize(venues) # flatten JSON
# filter columns
filtered_columns = ['venue.name', 'venue.categories', 'venue.location.lat', 'venue.location.lng']
PDnearby_venues4 = PDnearby_venues4.loc[:, filtered_columns]
# filter the category for each row
PDnearby_venues4['venue.categories'] = PDnearby_venues4.apply(get_category_type, axis=1)
# clean columns
PDnearby_venues4.columns = [col.split(".")[-1] for col in PDnearby_venues4.columns]

PDnearby_venues4.shape
```

Out[170]:

```
(5, 4)
```

In [179]:

```
PDnearby_venues4
```

Out[179]:

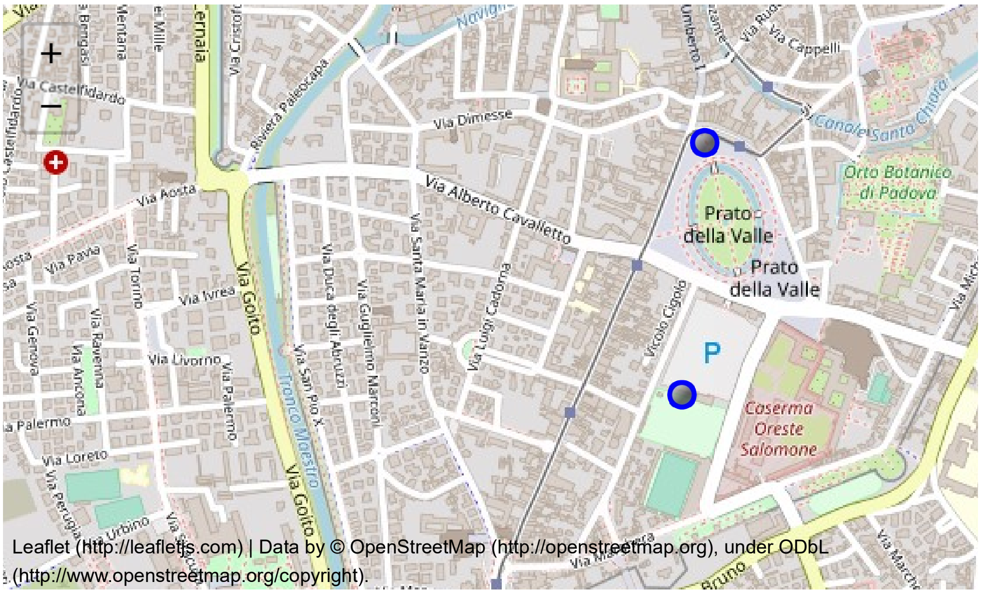
	name	categories	lat	lng
0	Cinema Lux	Movie Theater	45.395431	11.875456
1	Museo Del Precinema Palazzo Angeli	Museum	45.399790	11.875993
2	Giardini Sospesi	Concert Hall	45.390996	11.868909
3	Bastione Alicorno	Performing Arts Venue	45.391014	11.868284
4	Cinema Rex	Movie Theater	45.391082	11.891885

Map of my Padova/Padua home with arts and entertainment venues in Neighborhood

In [172]:

```
map_pd = folium.Map(location=[neighborhood_latitude, neighborhood_longitude], z
oom_start=15)
# add markers to map
for lat, lng, label in zip(PDnearby_venues4['lat'], PDnearby_venues4['lng'], PD
nearby_venues4['name']):
    label = folium.Popup(label, parse_html=True)
    folium.RegularPolygonMarker(
        [lat, lng],
        number_of_sides=30,
        radius=7,
        popup=label,
        color='blue',
        fill_color='#0f0f0f',
        fill_opacity=0.6,
    ).add_to(map_pd)

map_pd
```

Out[172]:



Leaflet (http://leafletjs.com) | Data by © OpenStreetMap (http://openstreetmap.org), under ODbL (http://www.openstreetmap.org/copyright).

1.5 Finally I merge all dataframes in a single one PDnearby_venues

In [173]:

```
PDnearby_venues_tot = pd.concat([PDnearby_venues, PDnearby_venues2, PDnearby_ve
nues3, PDnearby_venues4])
PDnearby_venues_tot.shape
```

Out[173]:

```
(27, 4)
```

In [174]:

```
PDnearby_venues.head()
```

Out[174]:

	name	categories	lat	lng
0	Prato della Valle	Plaza	45.398420	11.876521
1	Orto Botanico (Orto botanico di Padova)	Garden	45.399007	11.879603
2	Centro Giovanile Antonianum	Other Great Outdoors	45.398539	11.879618
3	Giardini Santa Rita	Park	45.391702	11.888398
4	Via Umberto I	Other Great Outdoors	45.401423	11.875735

In [175]:

```
PDnearby_venues_tot.tail()
```

Out[175]:

	name	categories	lat	lng
0	Cinema Lux	Movie Theater	45.395431	11.875456
1	Museo Del Precinema Palazzo Angeli	Museum	45.399790	11.875993
2	Giardini Sospesi	Concert Hall	45.390996	11.868909
3	Bastione Alicorno	Performing Arts Venue	45.391014	11.868284
4	Cinema Rex	Movie Theater	45.391082	11.891885

Step 2: Las Vegas

2.1 Las Vegas neighborhoods dataset

I searched a lot but I haven't found a clear dataset of Las Vegas neighborhoods. Moreover there are different "neighborhoods" for different sources.
Anyway I found a pretty good list of Las Vegas, North Las Vegas and Henderson neighborhoods that I added to the Census Designated Places to have a complete coverage of the different "neighborhoods" of Las Vegas.
Moreover I added Sloan and Boulder City.
I tried to get the neighborhood centroid automatically with code, but it was not realiable so I had to correct manually from Google Maps.
The result is an Excel file with City/CDP, Neighbordhoos, Latitude and Longitude that I exported as csv and imported as a panda dataframe.

In [77]:

```
lvneighborhoods = pd.read_csv("LV_neighborhoods.csv",sep=';',usecols=['City/CDP
', 'Neighborhood', 'Latitude', 'Longitude'] )
lvneighborhoods.head()
```

Out[77]:

	City/CDP	Neighborhood	Latitude	Longitude
0	Blue Diamond	Blue Diamond	36.046451	-115.405274
1	Boulder city	Boulder city	35.968495	-114.840873
2	Enterprise	Enterprise	36.025391	-115.208344
3	Henderson	Anthem	35.969019	-115.097036
4	Henderson	Black Mountain	36.018061	-114.981108

I use geopy library to get the latitude and longitude values of Las Vegas.

In [78]:

```
address = 'Las Vegas, NV'

geolocator = Nominatim(user_agent="lv_explorer")
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude
print('The geograpical coordinate of Las Vegas are {}, {}.'.format(latitude, lon
gitude))
```

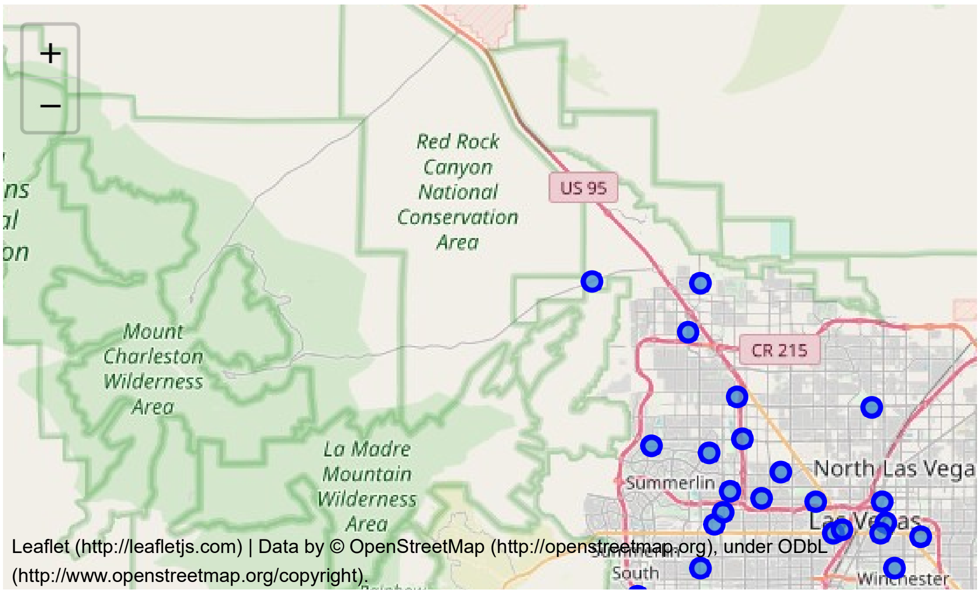
```
The geograpical coordinate of Las Vegas are 36.1662859, -115.149225.
```

Now I create a map of Las Vegas with neighborhoods superimposed on top.

In [91]:

```
# create map of Las Vegas using latitude and longitude values
map_lv = folium.Map(location=[latitude, longitude], zoom_start=10)

# add markers to map
for lat, lng, citycdp, neighborhood in zip(lvneighborhoods['Latitude'], lvneighb
orhoods['Longitude'], lvneighborhoods['City/CDP'], lvneighborhoods['Neighborhood
']):
    label = '{}, {}'.format(neighborhood, citycdp)
    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_lv)

map_lv
```

Out[91]:


+
−
Red Rock Canyon National Conservation Area
US 95
Mount Charleston Wilderness Area
La Madre Mountain Wilderness Area
CR 215
Summerlin
North Las Vega
Las Vegas
South
Winchester
Leaflet (http://leafletjs.com) | Data by © OpenStreetMap (http://openstreetmap.org), under ODbL (http://www.openstreetmap.org/copyright).

2.2 Las Vegas neighborhoods venues (source Foursquare API)

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

In [2]:

```
CLIENT_ID = 'DHQVKXTSH4R2LAKLZTI3NS5ZGHUQOBXLPD4SSEYJYSUYJ5KF' # my Foursquare I
D
CLIENT_SECRET = 'WM3FT0GHS5SFU2FME1JCFJARPJQJLXXEJP2LVUMDKLUX0PHB' # my Foursqua
re Secret
VERSION = '20190730' # Foursquare API version

print('My credentails:')
print('CLIENT_ID: ' + CLIENT_ID)
print('CLIENT_SECRET:' + CLIENT_SECRET)
```

```
My credentails:
CLIENT_ID: DHQVKXTSH4R2LAKLZTI3NS5ZGHUQOBXLPD4SSEYJYSUYJ5KF
CLIENT_SECRET:WM3FT0GHS5SFU2FME1JCFJARPJQJLXXEJP2LVUMDKLUX0PHB
```

From the Foursquare lab, 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.

In [3]:

```
# 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']
```

2.2.1 Let's create a function to extract Park and Playground venues for all the neighborhoods in Las Vegas

In [4]:

```
def getNearbyVenues(names, latitudes, longitudes, radius=500):

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

        # CATEGORIES I AM INTERESTED ARE (TO FILTER THE SEARCH)
        # PARK 4bf58dd8d48988d163941735,5bae9231bedf3950379f89d0,52e81612bcbc57f1066b7a21,52e81612bcbc57f1066b7a13,4bf58dd8d48988d162941735
        # PLAYGROUND 4bf58dd8d48988d1e7941735
        categoryId='4bf58dd8d48988d163941735,5bae9231bedf3950379f89d0,52e81612bcbc57f1066b7a21,52e81612bcbc57f1066b7a13,4bf58dd8d48988d162941735'

        LIMIT = 100 # limit of number of venues returned by Foursquare API

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

    return(nearby_venues)
```

In [23]:

```
lv_venues = getNearbyVenues(names=lvneighborhoods['Neighborhood'],
                                   latitudes=lvneighborhoods['Latitude'],
                                   longitudes=lvneighborhoods['Longitude']
                                  )
print(lv_venues.shape)
lv_venues.head()
```

```
Blue Diamond
Boulder city
Enterprise
Anthem
Black Mountain
Calico Ridge
Carver Park
Foothills
Gibson Springs
Green Valley North
Green Valley Ranch
Green Valley South
Highland Hills
Lake Las Vegas
Macdonald Ranch
McCullough Hills
Midway
Mission Hills
Paradise Hills
Pittman
River Mountain
Townsite
Valley View
Whitney Ranch
Angel Park Lindell
Buffalo
Centennial Hills
Charleston Heights
Cultural Corridor
Desert Shores
Downtown
Huntridge
Kyle Canyon
Lone Mountain
Meadows Village
Michael Way
Pioneer Park
Rancho Charleston
Silverado Ranch
Sun City Summerlin
Sunrise
The Lakes
Tule Springs
Twin Lakes
Umc
North Las Vegas
Paradise
Sloan
Spring Valley
Summerlin South
Sunrise Manor
Whitney
Winchester
(30, 7)
```

Out[23]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Blue Diamond	36.046451	-115.405274	Blue Diamond Park	36.048139	-115.408030	Park
1	Blue Diamond	36.046451	-115.405274	Blue Diamond Skate Park	36.048207	-115.406572	Skate Park
2	Boulder city	35.968495	-114.840873	Whalen Field	35.970052	-114.836850	Baseball Field
3	Anthem	35.969019	-115.097036	Anthem Gate	35.969818	-115.095146	Other Great Outdoors
4	Calico Ridge	36.080467	-114.958250	Tuscany Park	36.084794	-114.959327	Park

2.2.2 Let's create a function to extract Schools venues for all the neighborhoods in Las Vegas

In [24]:

```
def getNearbyVenues(names, latitudes, longitudes, radius=500):

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

        # CATEGORIES I AM INTERESTED ARE (TO FILTER THE SEARCH)
        # SCHOOLS Primary 4f4533804b9074f6e4fb0105, Middle 4f4533814b9074f6e4fb0106, High 4bf58dd8d48988d13d941735
        categoryId='4f4533804b9074f6e4fb0105,4f4533814b9074f6e4fb0106,4bf58dd8d48988d13d941735'
        LIMIT = 100 # limit of number of venues returned by Foursquare API

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

    return(nearby_venues)
```

In [25]:

```
lv_venues2 = getNearbyVenues(names=lvneighborhoods['Neighborhood'],
                                   latitudes=lvneighborhoods['Latitude'],
                                   longitudes=lvneighborhoods['Longitude']
                                  )
print(lv_venues2.shape)
lv_venues2.head()
```

```
Blue Diamond
Boulder city
Enterprise
Anthem
Black Mountain
Calico Ridge
Carver Park
Foothills
Gibson Springs
Green Valley North
Green Valley Ranch
Green Valley South
Highland Hills
Lake Las Vegas
Macdonald Ranch
McCullough Hills
Midway
Mission Hills
Paradise Hills
Pittman
River Mountain
Townsite
Valley View
Whitney Ranch
Angel Park Lindell
Buffalo
Centennial Hills
Charleston Heights
Cultural Corridor
Desert Shores
Downtown
Huntridge
Kyle Canyon
Lone Mountain
Meadows Village
Michael Way
Pioneer Park
Rancho Charleston
Silverado Ranch
Sun City Summerlin
Sunrise
The Lakes
Tule Springs
Twin Lakes
Umc
North Las Vegas
Paradise
Sloan
Spring Valley
Summerlin South
Sunrise Manor
Whitney
Winchester
(26, 7)
```

Out[25]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Blue Diamond	36.046451	-115.405274	blue diamond elementry	36.045538	-115.405004	Elementary School
1	Boulder city	35.968495	-114.840873	Martha P King Elementary School	35.970122	-114.837087	Elementary School
2	Calico Ridge	36.080467	-114.958250	Josh Stevens Elementary School	36.083612	-114.960514	Elementary School
3	Carver Park	36.045670	-114.978136	Lake Mead Christian Academy	36.043982	-114.973944	High School
4	Gibson Springs	36.043857	-115.039418	Kesterson elementary school	36.044616	-115.037027	Elementary School

2.2.3 Let's create a function to extract Shopping venues for all the neighborhoods in Las Vegas

In [27]:

```
def getNearbyVenues(names, latitudes, longitudes, radius=500):

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

        # CATEGORIES I AM INTERESTED ARE (TO FILTER THE SEARCH)
        # SHOPPING 4d4b7105d754a06378d81259
        categoryId='4d4b7105d754a06378d81259'
        LIMIT = 100 # limit of number of venues returned by Foursquare API

        # create the API request URL
        url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client
_secret={}&v={}&ll={},{}&categoryId={}&radius={}&limit={}'.format(
            CLIENT_ID,
            CLIENT_SECRET,
            VERSION,
            lat,
            lng,
            categoryId,
            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 = ['Neighborhood',
                  'Neighborhood Latitude',
                  'Neighborhood Longitude',
                  'Venue',
                  'Venue Latitude',
                  'Venue Longitude',
                  'Venue Category']

    return(nearby_venues)
```

In [28]:

```
lv_venues3 = getNearbyVenues(names=lvneighborhoods['Neighborhood'],
                                   latitudes=lvneighborhoods['Latitude'],
                                   longitudes=lvneighborhoods['Longitude']
                                  )
print(lv_venues3.shape)
lv_venues3.head()
```

```
Blue Diamond
Boulder city
Enterprise
Anthem
Black Mountain
Calico Ridge
Carver Park
Foothills
Gibson Springs
Green Valley North
Green Valley Ranch
Green Valley South
Highland Hills
Lake Las Vegas
Macdonald Ranch
McCullough Hills
Midway
Mission Hills
Paradise Hills
Pittman
River Mountain
Townsite
Valley View
Whitney Ranch
Angel Park Lindell
Buffalo
Centennial Hills
Charleston Heights
Cultural Corridor
Desert Shores
Downtown
Huntridge
Kyle Canyon
Lone Mountain
Meadows Village
Michael Way
Pioneer Park
Rancho Charleston
Silverado Ranch
Sun City Summerlin
Sunrise
The Lakes
Tule Springs
Twin Lakes
Umc
North Las Vegas
Paradise
Sloan
Spring Valley
Summerlin South
Sunrise Manor
Whitney
Winchester
(560, 7)
```

Out[28]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Blue Diamond	36.046451	-115.405274	420 fo sho	36.045619	-115.404860	Pharmacy
1	Boulder city	35.968495	-114.840873	Superior Tennis Courts	35.972070	-114.842970	Construction & Landscaping
2	Enterprise	36.025391	-115.208344	Laundry Room	36.028512	-115.211134	Video Store
3	Enterprise	36.025391	-115.208344	Upstairs Loft	36.028539	-115.211155	Gift Shop
4	Enterprise	36.025391	-115.208344	SC At Caprock	36.028700	-115.210930	Garden Center

2.2.4 Let's create a function to extract Arts and entertainment venues for all the neighborhoods in Las Vegas

In [29]:

```
def getNearbyVenues(names, latitudes, longitudes, radius=500):

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

        # CATEGORIES I AM INTERESTED ARE (TO FILTER THE SEARCH)
        # ARTS AND THEATRES 4d4b7104d754a06370d81259
        categoryId='4d4b7104d754a06370d81259'
        LIMIT = 100 # limit of number of venues returned by Foursquare API

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

    return(nearby_venues)
```

In [30]:
```
lv_venues4 = getNearbyVenues(names=lvneighborhoods['Neighborhood'],
                                   latitudes=lvneighborhoods['Latitude'],
                                   longitudes=lvneighborhoods['Longitude']
                                  )
print(lv_venues4.shape)
lv_venues4.head()
```

```
Blue Diamond
Boulder city
Enterprise
Anthem
Black Mountain
Calico Ridge
Carver Park
Foothills
Gibson Springs
Green Valley North
Green Valley Ranch
Green Valley South
Highland Hills
Lake Las Vegas
Macdonald Ranch
McCullough Hills
Midway
Mission Hills
Paradise Hills
Pittman
River Mountain
Townsite
Valley View
Whitney Ranch
Angel Park Lindell
Buffalo
Centennial Hills
Charleston Heights
Cultural Corridor
Desert Shores
Downtown
Huntridge
Kyle Canyon
Lone Mountain
Meadows Village
Michael Way
Pioneer Park
Rancho Charleston
Silverado Ranch
Sun City Summerlin
Sunrise
The Lakes
Tule Springs
Twin Lakes
Umc
North Las Vegas
Paradise
Sloan
Spring Valley
Summerlin South
Sunrise Manor
Whitney
Winchester
(45, 7)
```

Out[30]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Blue Diamond	36.046451	-115.405274	Vannila Gorilla	36.047680	-115.404705	Comedy Club
1	Green Valley Ranch	36.015370	-115.083006	Henderson Pavilion	36.019116	-115.081312	Concert Hall
2	Green Valley Ranch	36.015370	-115.083006	Henderson Symphony Orchestra	36.017552	-115.080299	Music Venue
3	Green Valley Ranch	36.015370	-115.083006	The Gallery at Liberty Point	36.018524	-115.079437	Art Gallery
4	Lake Las Vegas	36.113338	-114.924954	The Event Dance Competition	36.114233	-114.923357	Dance Studio

2.2.5 Let's merge all venues toegether for all the neighborhoods in Las Vegas

In [31]:

```
lv_venues_tot = pd.concat([lv_venues, lv_venues2, lv_venues3, lv_venues4])
lv_venues_tot.shape
```

Out[31]:

```
(661, 7)
```

2.2.6 Let's check how many venues were returned for each neighborhood

In [32]:

```
lv_venues_tot.groupby('Neighborhood').count()
```

Out[32]:

	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
Neighborhood						
Angel Park Lindell	5	5	5	5	5	5
Anthem	5	5	5	5	5	5
Black Mountain	1	1	1	1	1	1
Blue Diamond	5	5	5	5	5	5
Boulder city	3	3	3	3	3	3
Buffalo	18	18	18	18	18	18
Calico Ridge	5	5	5	5	5	5
Carver Park	8	8	8	8	8	8
Centennial Hills	35	35	35	35	35	35
Charleston Heights	3	3	3	3	3	3
Cultural Corridor	64	64	64	64	64	64
Desert Shores	14	14	14	14	14	14
Downtown	31	31	31	31	31	31
Enterprise	3	3	3	3	3	3
Foothills	1	1	1	1	1	1
Gibson Springs	9	9	9	9	9	9
Green Valley North	8	8	8	8	8	8
Green Valley Ranch	6	6	6	6	6	6
Green Valley South	7	7	7	7	7	7
Highland Hills	7	7	7	7	7	7
Huntridge	63	63	63	63	63	63
Lake Las Vegas	6	6	6	6	6	6
Lone Mountain	12	12	12	12	12	12
Macdonald Ranch	2	2	2	2	2	2
McCullough Hills	2	2	2	2	2	2
Meadows Village	24	24	24	24	24	24
Michael Way	3	3	3	3	3	3
Midway	18	18	18	18	18	18
Mission Hills	1	1	1	1	1	1
North Las Vegas	9	9	9	9	9	9
Paradise	1	1	1	1	1	1
Paradise Hills	6	6	6	6	6	6
Pioneer Park	5	5	5	5	5	5
Pittman	11	11	11	11	11	11
Rancho Charleston	44	44	44	44	44	44
River Mountain	3	3	3	3	3	3
Silverado Ranch	5	5	5	5	5	5
Spring Valley	2	2	2	2	2	2

Let's find out how many unique categories can be curated from all the returned venues

```
In [33]: print('There are {} uniques categories.'.format(len(lv_venues['Venue Category'].
         unique())))
```

```
There are 14 uniques categories.
```

2.3 Now I analyze each Las Vegas neighborhood

```
In [34]: # one hot encoding
         lv_onehot = pd.get_dummies(lv_venues[['Venue Category']], prefix="", prefix_se
         p="")

         # add neighborhood column back to dataframe
         lv_onehot['Neighborhood'] = lv_venues['Neighborhood']

         # move neighborhood column to the first column
         fixed_columns = [lv_onehot.columns[-1]] + list(lv_onehot.columns[:-1])
         lv_onehot = lv_onehot[fixed_columns]

         lv_onehot.head()
```

Out[34]:

	Neighborhood	Baseball Field	Building	Convention Center	Food Truck	Funeral Home	Health & Beauty Service	History Museum	Non-Profit	Other Great Outdoors	Parl
0	Blue Diamond	0	0	0	0	0	0	0	0	0	1
1	Blue Diamond	0	0	0	0	0	0	0	0	0	0
2	Boulder city	1	0	0	0	0	0	0	0	0	0
3	Anthem	0	0	0	0	0	0	0	0	1	0
4	Calico Ridge	0	0	0	0	0	0	0	0	0	1

```
In [35]: lv_onehot.shape
```

```
Out[35]: (30, 15)
```

Next, let's group rows by neighborhood and by taking the mean of the frequency of occurrence of each category

In [36]:
```
lv_grouped = lv_onehot.groupby('Neighborhood').mean().reset_index()
lv_grouped
```

Out[36]:

	Neighborhood	Baseball Field	Building	Convention Center	Food Truck	Funeral Home	Health & Beauty Service	History Museum	Non-Profit	Other Great Outdoors	Pa
0	Anthem	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0
1	Blue Diamond	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
2	Boulder city	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
3	Buffalo	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1
4	Calico Ridge	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1
5	Carver Park	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
6	Charleston Heights	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1
7	Cultural Corridor	0.0	0.0	0.5	0.0	0.0	0.0	0.5	0.0	0.0	0
8	Desert Shores	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0
9	Downtown	0.0	0.0	0.0	0.5	0.0	0.0	0.0	0.5	0.0	0
10	Gibson Springs	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1
11	Green Valley North	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
12	Green Valley South	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1
13	Highland Hills	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
14	Huntridge	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.0	0
15	Lake Las Vegas	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1
16	Lone Mountain	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0
17	Pioneer Park	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1
18	River Mountain	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1
19	Summerlin South	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
20	Townsite	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1
21	Tule Springs	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1
22	Twin Lakes	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
23	Whitney Ranch	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1

In [37]:
```
lv_grouped.shape
```

Out[37]:
```
(24, 15)
```

Let's print each neighborhood along with the top 5 most common venues

In [38]:

```
num_top_venues = 5

for hood in lv_grouped['Neighborhood']:
    print("----"+hood+"----")
    temp = lv_grouped[lv_grouped['Neighborhood'] == hood].T.reset_index()
    temp.columns = ['venue','freq']
    temp = temp.iloc[1:]
    temp['freq'] = temp['freq'].astype(float)
    temp = temp.round({'freq': 2})
    print(temp.sort_values('freq', ascending=False).reset_index(drop=True).head(num_top_venues))
    print('\n')
```

```
----Anthem----
                  venue  freq
0  Other Great Outdoors   1.0
1        Baseball Field   0.0
2              Building   0.0
3     Convention Center   0.0
4            Food Truck   0.0


----Blue Diamond----
               venue  freq
0               Park   0.5
1         Skate Park   0.5
2     Baseball Field   0.0
3           Building   0.0
4  Convention Center   0.0


----Boulder city----
               venue  freq
0     Baseball Field   1.0
1           Building   0.0
2  Convention Center   0.0
3         Food Truck   0.0
4       Funeral Home   0.0


----Buffalo----
               venue  freq
0               Park   1.0
1     Baseball Field   0.0
2           Building   0.0
3  Convention Center   0.0
4         Food Truck   0.0


----Calico Ridge----
               venue  freq
0               Park   1.0
1     Baseball Field   0.0
2           Building   0.0
3  Convention Center   0.0
4         Food Truck   0.0


----Carver Park----
               venue  freq
0           Building   1.0
1     Baseball Field   0.0
2  Convention Center   0.0
3         Food Truck   0.0
4       Funeral Home   0.0


----Charleston Heights----
               venue  freq
0               Park   1.0
1     Baseball Field   0.0
2           Building   0.0
3  Convention Center   0.0
4         Food Truck   0.0


----Cultural Corridor----
               venue  freq
0  Convention Center   0.5
1     History Museum   0.5
2     Baseball Field   0.0
3           Building   0.0
```

Let's put that into a *panda* dataframe

First, let's write a function to sort the venues in descending order.

In [39]:
```
def return_most_common_venues(row, num_top_venues):
    row_categories = row.iloc[1:]
    row_categories_sorted = row_categories.sort_values(ascending=False)

    return row_categories_sorted.index.values[0:num_top_venues]
```

Now let's create the new dataframe and display the top 10 venues for each neighborhood.

In [40]:
```
num_top_venues = 10

indicators = ['st', 'nd', 'rd']

# create columns according to number of top venues
columns = ['Neighborhood']
for ind in np.arange(num_top_venues):
    try:
        columns.append('{}{} Most Common Venue'.format(ind+1, indicators[ind]))
    except:
        columns.append('{}th Most Common Venue'.format(ind+1))

# create a new dataframe
neighborhoods_venues_sorted = pd.DataFrame(columns=columns)
neighborhoods_venues_sorted['Neighborhood'] = lv_grouped['Neighborhood']

for ind in np.arange(lv_grouped.shape[0]):
    neighborhoods_venues_sorted.iloc[ind, 1:] = return_most_common_venues(lv_gro
uped.iloc[ind, :], num_top_venues)

neighborhoods_venues_sorted.head()
```

Out[40]:

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue	9tl Co
0	Anthem	Other Great Outdoors	Skate Park	Residential Building (Apartment / Condo)	Real Estate Office	Playground	Park	Non-Profit	History Museum	He I S
1	Blue Diamond	Skate Park	Park	Residential Building (Apartment / Condo)	Real Estate Office	Playground	Other Great Outdoors	Non-Profit	History Museum	He I S
2	Boulder city	Baseball Field	Skate Park	Residential Building (Apartment / Condo)	Real Estate Office	Playground	Park	Other Great Outdoors	Non-Profit	I M
3	Buffalo	Park	Skate Park	Residential Building (Apartment / Condo)	Real Estate Office	Playground	Other Great Outdoors	Non-Profit	History Museum	He I S
4	Calico Ridge	Park	Skate Park	Residential Building (Apartment / Condo)	Real Estate Office	Playground	Other Great Outdoors	Non-Profit	History Museum	He I S

2.4 Cluster Las Vegas neighborhood

Run k-means to cluster the neighborhood into 5 clusters.

In [42]:

```
# set number of clusters
kclusters = 5

lv_grouped_clustering = lv_grouped.drop('Neighborhood', 1)

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

# check cluster labels generated for each row in the dataframe
kmeans.labels_[0:10]
```

Out[42]:

```
array([2, 1, 0, 1, 1, 2, 1, 2, 2, 2])
```

Let's create a new dataframe that includes the cluster as well as the top 10 venues for each neighborhood.

In [61]:

```
# add clustering labels
neighborhoods_venues_sorted.insert(0, 'Cluster Labels', kmeans.labels_)

lv_merged = lvneighborhoods

# merge lv_grouped with lv_data to add latitude/longitude for each neighborhood
lv_merged = lv_merged.join(neighborhoods_venues_sorted.set_index('Neighborhood'), on='Neighborhood')

lv_merged = lv_merged.dropna().astype({"Cluster Labels": int})

lv_merged.head() # check the last columns!
```

Out[61]:

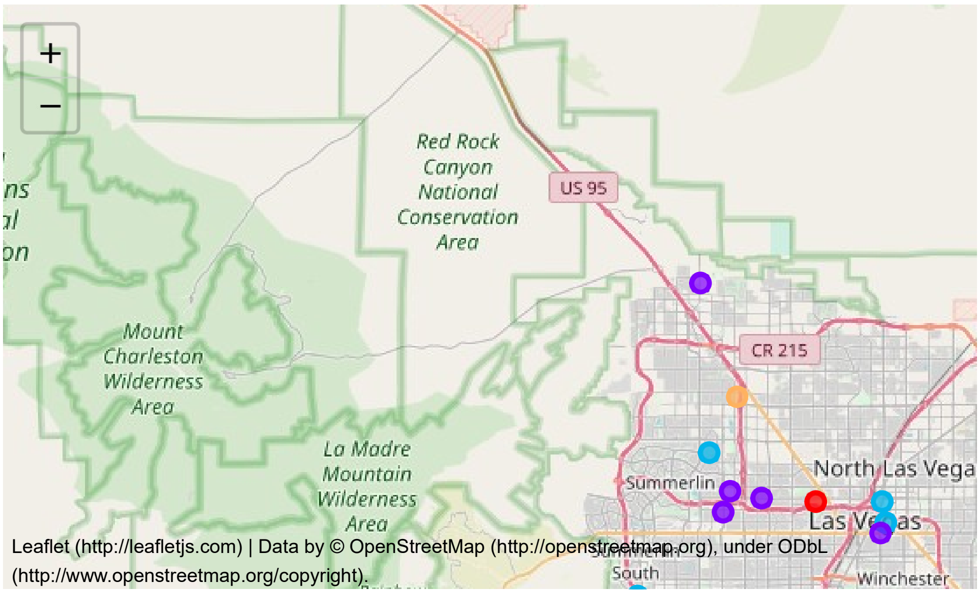
	City/CDP	Neighborhood	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	(
0	Blue Diamond	Blue Diamond	36.046451	-115.405274	1	Skate Park	Park	Residential Building (Apartment / Condo)	Real Estate Office	Pl
1	Boulder city	Boulder city	35.968495	-114.840873	0	Baseball Field	Skate Park	Residential Building (Apartment / Condo)	Real Estate Office	Pl
3	Henderson	Anthem	35.969019	-115.097036	2	Other Great Outdoors	Skate Park	Residential Building (Apartment / Condo)	Real Estate Office	Pl
5	Henderson	Calico Ridge	36.080467	-114.958250	1	Park	Skate Park	Residential Building (Apartment / Condo)	Real Estate Office	Pl
6	Henderson	Carver Park	36.045670	-114.978136	2	Building	Skate Park	Residential Building (Apartment / Condo)	Real Estate Office	Pl

Finally, let's visualize the resulting clusters

In [64]:

```
# create map
map_clusters = folium.Map(location=[latitude, longitude], zoom_start=10)

# set color scheme for the clusters
x = np.arange(kclusters)
ys = [i + x + (i*x)**2 for i in range(kclusters)]
colors_array = cm.rainbow(np.linspace(0, 1, len(ys)))
rainbow = [colors.rgb2hex(i) for i in colors_array]

# add markers to the map
markers_colors = []
for lat, lon, poi, cluster in zip(lv_merged['Latitude'], lv_merged['Longitude'],
lv_merged['Neighborhood'], lv_merged['Cluster Labels']):
    label = folium.Popup(str(poi) + ' Cluster ' + str(cluster), parse_html=True)
    folium.CircleMarker(
        [lat, lon],
        radius=5,
        popup=label,
        color=rainbow[cluster-1],
        fill=True,
        fill_color=rainbow[cluster-1],
        fill_opacity=0.7).add_to(map_clusters)

map_clusters
```

Out[64]:


+
−
Red Rock Canyon National Conservation Area
US 95
CR 215
Mount Charleston Wilderness Area
La Madre Mountain Wilderness Area
Summerlin
North Las Vega
Las Vegas
Summerlin South
Winchester

Leaflet (http://leafletjs.com) | Data by © OpenStreetMap (http://openstreetmap.org), under ODbL (http://www.openstreetmap.org/copyright).

2.5 Examine the clusters in Las Vegas neighborhood

Now, I can examine each cluster and determine the discriminating venue categories that distinguish each cluster. Based on the defining categories, I can then assign a name to each cluster.

Cluster 1

In [72]:
```
lv_merged.loc[lv_merged['Cluster Labels'] == 0, lv_merged.columns[[1] + list(ran
ge(5, lv_merged.shape[1]))]]
```

Out[72]:

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue
1	Boulder city	Baseball Field	Skate Park	Residential Building (Apartment / Condo)	Real Estate Office	Playground	Park	Other Great Outdoors	Non-Profit
12	Highland Hills	Baseball Field	Skate Park	Residential Building (Apartment / Condo)	Real Estate Office	Playground	Park	Other Great Outdoors	Non-Profit
43	Twin Lakes	Park	Baseball Field	Skate Park	Residential Building (Apartment / Condo)	Real Estate Office	Playground	Other Great Outdoors	Non-Profit

Cluster 2

In [73]:

```
lv_merged.loc[lv_merged['Cluster Labels'] == 1, lv_merged.columns[[1] + list(range(5, lv_merged.shape[1]))]]
```

Out[73]:

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue
0	Blue Diamond	Skate Park	Park	Residential Building (Apartment / Condo)	Real Estate Office	Playground	Other Great Outdoors	Non-Profit	History Museum
5	Calico Ridge	Park	Skate Park	Residential Building (Apartment / Condo)	Real Estate Office	Playground	Other Great Outdoors	Non-Profit	History Museum
8	Gibson Springs	Park	Skate Park	Residential Building (Apartment / Condo)	Real Estate Office	Playground	Other Great Outdoors	Non-Profit	History Museum
11	Green Valley South	Park	Skate Park	Residential Building (Apartment / Condo)	Real Estate Office	Playground	Other Great Outdoors	Non-Profit	History Museum
13	Lake Las Vegas	Park	Skate Park	Residential Building (Apartment / Condo)	Real Estate Office	Playground	Other Great Outdoors	Non-Profit	History Museum
20	River Mountain	Park	Skate Park	Residential Building (Apartment / Condo)	Real Estate Office	Playground	Other Great Outdoors	Non-Profit	History Museum
21	Townsite	Park	Skate Park	Residential Building (Apartment / Condo)	Real Estate Office	Playground	Other Great Outdoors	Non-Profit	History Museum
23	Whitney Ranch	Park	Skate Park	Residential Building (Apartment / Condo)	Real Estate Office	Playground	Other Great Outdoors	Non-Profit	History Museum
25	Buffalo	Park	Skate Park	Residential Building (Apartment / Condo)	Real Estate Office	Playground	Other Great Outdoors	Non-Profit	History Museum
27	Charleston Heights	Park	Skate Park	Residential Building (Apartment / Condo)	Real Estate Office	Playground	Other Great Outdoors	Non-Profit	History Museum
31	Huntridge	Park	Non-Profit	Skate Park	Residential Building (Apartment / Condo)	Real Estate Office	Playground	Other Great Outdoors	History Museum
36	Pioneer Park	Park	Skate Park	Residential Building (Apartment / Condo)	Real Estate Office	Playground	Other Great Outdoors	Non-Profit	History Museum
42	Tule Springs	Park	Skate Park	Residential Building (Apartment / Condo)	Real Estate Office	Playground	Other Great Outdoors	Non-Profit	History Museum

Cluster 3

In [74]:
```
lv_merged.loc[lv_merged['Cluster Labels'] == 2, lv_merged.columns[[1] + list(ran
ge(5, lv_merged.shape[1]))]]
```

Out[74]:

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue
3	Anthem	Other Great Outdoors	Skate Park	Residential Building (Apartment / Condo)	Real Estate Office	Playground	Park	Non-Profit	History Museum
6	Carver Park	Building	Skate Park	Residential Building (Apartment / Condo)	Real Estate Office	Playground	Park	Other Great Outdoors	Non-Profit
28	Cultural Corridor	History Museum	Convention Center	Skate Park	Residential Building (Apartment / Condo)	Real Estate Office	Playground	Park	Other Great Outdoors
29	Desert Shores	Health & Beauty Service	Skate Park	Residential Building (Apartment / Condo)	Real Estate Office	Playground	Park	Other Great Outdoors	Non-Profit
30	Downtown	Non-Profit	Food Truck	Skate Park	Residential Building (Apartment / Condo)	Real Estate Office	Playground	Park	Other Great Outdoors
49	Summerlin South	Real Estate Office	Playground	Skate Park	Residential Building (Apartment / Condo)	Park	Other Great Outdoors	Non-Profit	History Museum

Cluster 4

In [75]:
```
lv_merged.loc[lv_merged['Cluster Labels'] == 3, lv_merged.columns[[1] + list(ran
ge(5, lv_merged.shape[1]))]]
```

Out[75]:

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue	9th Co
9	Green Valley North	Residential Building (Apartment / Condo)	Skate Park	Real Estate Office	Playground	Park	Other Great Outdoors	Non-Profit	History Museum	He I S

Cluster 5

In [76]:
```
lv_merged.loc[lv_merged['Cluster Labels'] == 4, lv_merged.columns[[1] + list(ran
ge(5, lv_merged.shape[1]))]]
```

Out[76]:

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue	9 C
33	Lone Mountain	Funeral Home	Skate Park	Residential Building (Apartment / Condo)	Real Estate Office	Playground	Park	Other Great Outdoors	Non-Profit	M

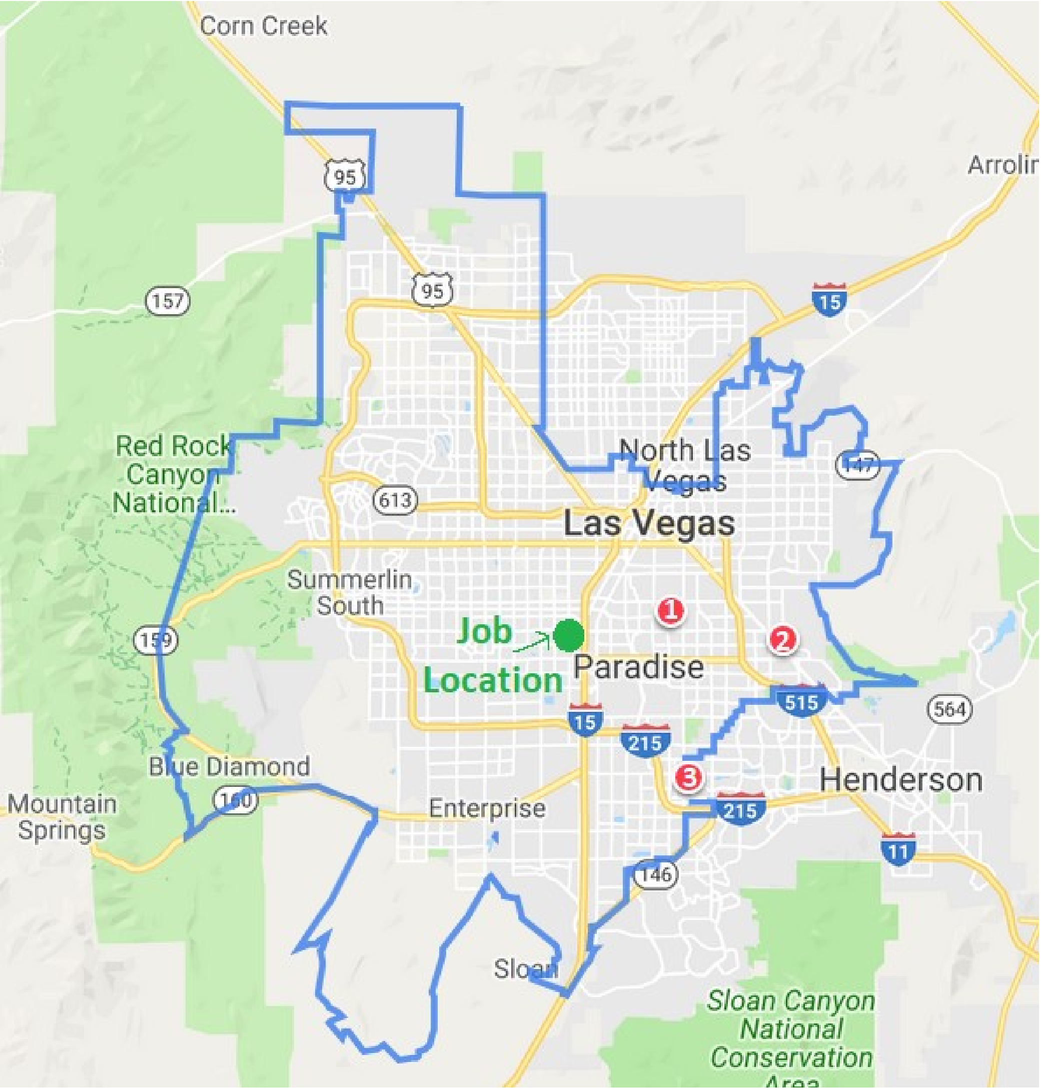
2.6 Las Vegas potential houses

Now that I have evaluated the venues and the clusters in the Las Vegas area, I look on Zillow to find potential houses that respect the family requisites (parks and schools for children, shopping venues for my wife and arts and entertainment for me).
Last but not least, I consider the distance from the job location.

So I looked on Zillow and I filtered the results based on our family requirements:

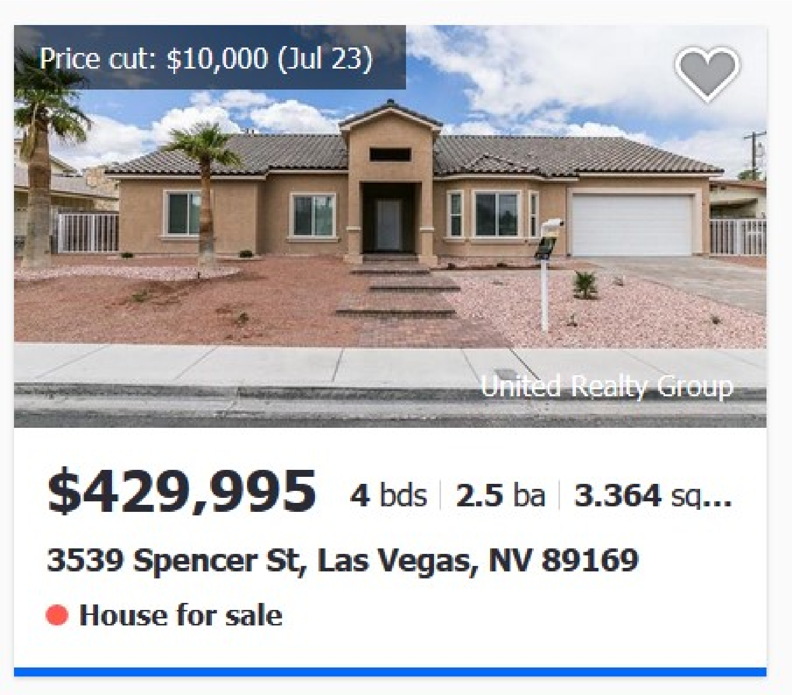
- single house
- min 2 bedrooms
- min 2 bathrooms
- a garage
- a garden
- swimming pool
- no HOA fees
- build after 2009
- budget of $500k
- nearby primary, middle and high schools
- maximum distance from Caesars Palace Casino (job location): 10 miles

This is the query result from Zillow website:



As you can see there are only 3 options (#1 and #2 are in the budget, while #3 is just over the budget)

House #1 (3.6mi from house to job location) - Winchester neighborhood

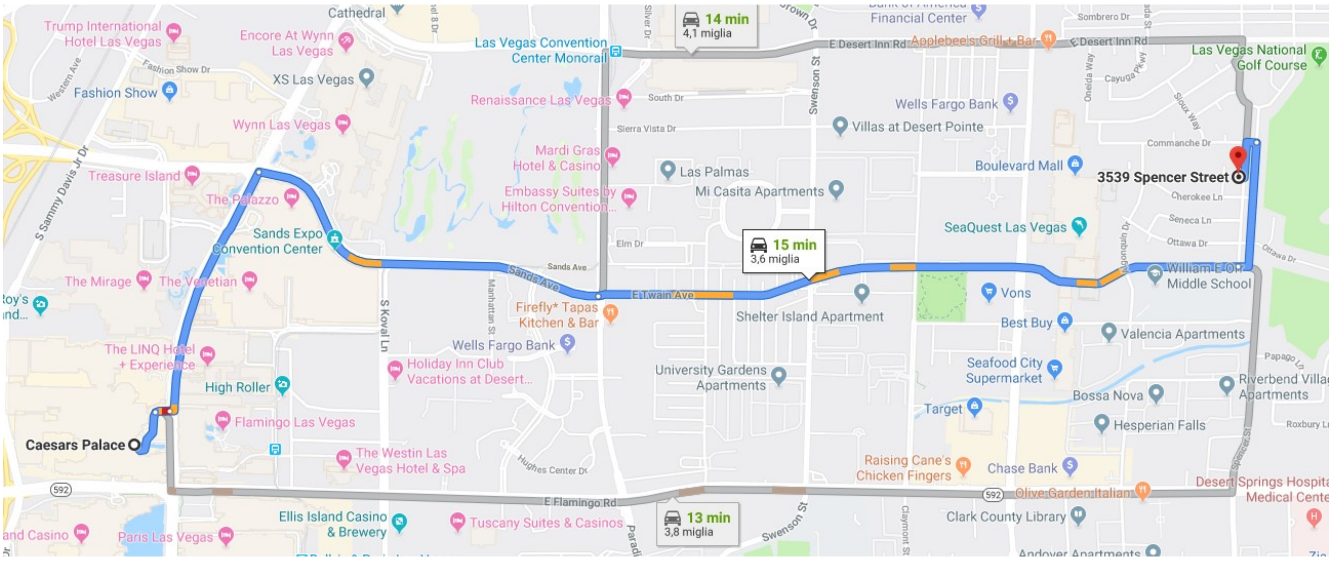


Price cut: $10,000 (Jul 23)

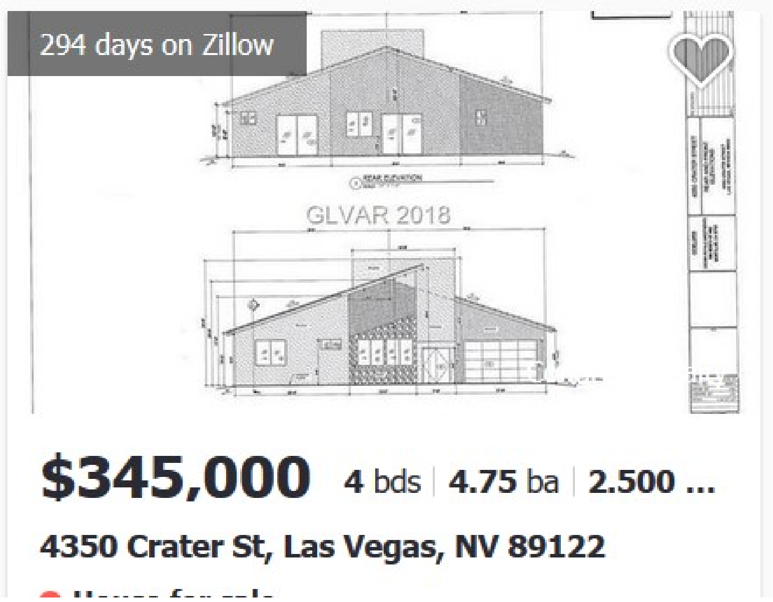
$429,995 4 bds | 2.5 ba | 3.364 sq...

3539 Spencer St, Las Vegas, NV 89169

● House for sale



House #2 (7.1mi from house to job location) - Whitney neighborhood



294 days on Zillow

$345,000 4 bds | 4.75 ba | 2.500 ...

4350 Crater St, Las Vegas, NV 89122

● House for sale

Let's recheck these 3 neighborhoods, now that I have also the latitude and longitude of the 3 potential houses.
The radius of 1000 it is the same used for Padova/Padua.

In [89]:
```
lvhouses = pd.read_csv("LV_Houses.csv",sep=';')
lvhouses.head()
```

Out[89]:

	ID_House	Address	Latitude	Longitude	Bds	Ba	sqft	Price	Job distance (mi)
0	1	3539 Spencer St., Las Vegas, NV 89169	36.125213	-115.130470	4	3	3364	429995	3,6
1	2	4530 Crater St., Las Vegas, NV 89122	36.109992	-115.058833	4	5	2500	345000	7,1
2	3	8576 Hogan Falls Cir., Las Vegas, NV 89123	36.033679	-115.125374	3	3	2981	519900	8

2.6.1 Parks and playgrounds

In [148]:

```python
def getNearbyVenues(names, latitudes, longitudes, radius=1000):
    
    venues_list=[]
    for name, lat, lng in zip(names, latitudes, longitudes):
        print(name)
        
        # CATEGORIES I AM INTERESTED ARE (TO FILTER THE SEARCH)
        # PARK 4bf58dd8d48988d163941735,5bae9231bedf3950379f89d0,52e81612bcbc57f1066b7a21,52e81612bcbc57f1066b7a13,4bf58dd8d48988d162941735
        # PLAYGROUND 4bf58dd8d48988d1e7941735
        categoryId='4bf58dd8d48988d163941735,5bae9231bedf3950379f89d0,52e81612bcbc57f1066b7a21,52e81612bcbc57f1066b7a13,4bf58dd8d48988d162941735'
        
        LIMIT = 100 # limit of number of venues returned by Foursquare API
        
        # create the API request URL
        url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={},{}&categoryId={}&radius={}&limit={}'.format(
            CLIENT_ID, 
            CLIENT_SECRET, 
            VERSION, 
            lat, 
            lng, 
            categoryId,
            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 = ['House', 
                  'House Latitude', 
                  'House Longitude', 
                  'Venue', 
                  'Venue Latitude', 
                  'Venue Longitude', 
                  'Venue Category']
    
    return(nearby_venues)
```

In [149]:

```
lv_h_venues = getNearbyVenues(names=lvhouses['ID_House'],
                                   latitudes=lvhouses['Latitude'],
                                   longitudes=lvhouses['Longitude']
                                  )
lv_h_venues.sort_values(by=['House'])
```

```
1
2
3
```

Out[149]:

	House	House Latitude	House Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	1	36.125213	-115.130470	Maryland Park Aparments	36.125145	-115.139727	Residential Building (Apartment / Condo)
1	1	36.125213	-115.130470	Pinapple Park	36.125766	-115.135175	Dessert Shop
2	2	36.109992	-115.058833	Dog Fancier's Park	36.108859	-115.048490	Dog Run
3	2	36.109992	-115.058833	Horsemans park	36.111641	-115.049927	Park
4	2	36.109992	-115.058833	Sam's Town RV Park	36.109906	-115.058996	RV Park
5	2	36.109992	-115.058833	Mystic Falls Park	36.112765	-115.062435	Other Nightlife
6	2	36.109992	-115.058833	Dr Peter Park, Optometrist.	36.109218	-115.063352	Doctor's Office
7	3	36.033679	-115.125374	Park Warrior Arena	36.035020	-115.135818	Athletics & Sports
8	3	36.033679	-115.125374	Park Animal Hospital	36.036644	-115.117877	Veterinarian

Parks and playgrounds:

- House #1: no parks
- House #2: 3 parks (two parks and one dog run)
- House #3: 1 park

2.6.2 Schools (primary, middle and high schools)

In [114]:

```
def getNearbyVenues(names, latitudes, longitudes, radius=1000):

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

        # CATEGORIES I AM INTERESTED ARE (TO FILTER THE SEARCH)
        # SCHOOLS Primary 4f4533804b9074f6e4fb0105, Middle 4f4533814b9074f6e4fb0106, High 4bf58dd8d48988d13d941735
        categoryId='4f4533804b9074f6e4fb0105,4f4533814b9074f6e4fb0106,4bf58dd8d48988d13d941735'

        LIMIT = 100 # limit of number of venues returned by Foursquare API

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

    return(nearby_venues)
```

In [115]:

```
lv_h2_venues = getNearbyVenues(names=lvhouses['ID_House'],
                                   latitudes=lvhouses['Latitude'],
                                   longitudes=lvhouses['Longitude']
                                  )
lv_h2_venues.sort_values(by=['House'])
```

```
1
2
3
```

Out[115]:

	House	House Latitude	House Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	1	36.125213	-115.130470	Ruby Thomas Elementary School	36.124459	-115.131157	Elementary School
1	1	36.125213	-115.130470	William E. Orr Middle School	36.121590	-115.130925	Middle School
2	1	36.125213	-115.130470	Dean Peterson Elementary	36.123455	-115.140581	Elementary School
3	2	36.109992	-115.058833	Bailey Elementary	36.106451	-115.051305	Elementary School
4	2	36.109992	-115.058833	Cunningham Elementary	36.114395	-115.051705	Elementary School
5	2	36.109992	-115.058833	J.M. Ullom Elementary School	36.104142	-115.067164	Elementary School

Schools:

- House #1: both elementary (primary) and middle school
- House #2: only elementary schools
- House #3: no schools in the radius

2.6.3 Shopping venues

In [116]:

```
def getNearbyVenues(names, latitudes, longitudes, radius=1000):

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

        # CATEGORIES I AM INTERESTED ARE (TO FILTER THE SEARCH)
        # SHOPPING 4d4b7105d754a06378d81259
        categoryId='4d4b7105d754a06378d81259'

        LIMIT = 100 # limit of number of venues returned by Foursquare API

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

    return(nearby_venues)
```

In [121]:

```
lv_h3_venues = getNearbyVenues(names=lvhouses['ID_House'],
                                   latitudes=lvhouses['Latitude'],
                                   longitudes=lvhouses['Longitude']
                                  )
lv_h3_venues.House.value_counts()
```

```
1
2
3
```

Out[121]:

```
1    100
3     80
2     27
Name: House, dtype: int64
```

Shopping venues:

- House #1: plenty of shopping venues
- House #2: plenty of shopping venues
- House #3: not so many shopping venues

2.6.4 Arts and entertainment venues

In [123]:

```
def getNearbyVenues(names, latitudes, longitudes, radius=1000):

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

        # CATEGORIES I AM INTERESTED ARE (TO FILTER THE SEARCH)
        # ARTS AND THEATRES 4d4b7104d754a06370d81259
        categoryId='4d4b7104d754a06370d81259'

        LIMIT = 100 # limit of number of venues returned by Foursquare API

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

    return(nearby_venues)
```

In [124]:

```
lv_h4_venues = getNearbyVenues(names=lvhouses['ID_House'],
                                   latitudes=lvhouses['Latitude'],
                                   longitudes=lvhouses['Longitude']
                                  )
lv_h4_venues.sort_values(by=['House'])
```

```
1
2
3
```

Out[124]:

	House	House Latitude	House Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	1	36.125213	-115.130470	Galaxy Theatre Blvd	36.124081	-115.134654	Movie Theater
1	1	36.125213	-115.130470	Band-Aid Booking	36.124840	-115.134530	Rock Club
2	1	36.125213	-115.130470	Rex Center	36.123038	-115.134738	Laser Tag
3	1	36.125213	-115.130470	Hispanic Museum Of Nevada	36.123868	-115.135471	Art Gallery
4	1	36.125213	-115.130470	Stanislav Georgiev	36.124842	-115.136536	Art Gallery
5	1	36.125213	-115.130470	Golf House OXYGEN	36.120880	-115.127118	Mini Golf
6	1	36.125213	-115.130470	Exoticdancers	36.125302	-115.137182	Concert Hall
7	1	36.125213	-115.130470	Henna Art	36.122747	-115.137085	Art Gallery
8	1	36.125213	-115.130470	JB Dance Studio	36.122690	-115.137056	Salsa Club
14	2	36.109992	-115.058833	Century Riverside 12	36.112895	-115.062353	Movie Theater
13	2	36.109992	-115.058833	Sam's Town Showroom (holiday Show!)	36.112808	-115.062289	Music Venue
12	2	36.109992	-115.058833	Theater 17	36.113361	-115.060700	Movie Theater
9	2	36.109992	-115.058833	Century 18 Sam's Town	36.112024	-115.061304	Movie Theater
10	2	36.109992	-115.058833	Sam's Town Live!	36.112532	-115.061577	Rock Club
11	2	36.109992	-115.058833	Theater 16	36.111906	-115.060839	Movie Theater
17	3	36.033679	-115.125374	River Church	36.038618	-115.131620	Music Venue
15	3	36.033679	-115.125374	Entertainment Unlimited	36.032110	-115.118319	Concert Hall
16	3	36.033679	-115.125374	Saccardi manner	36.036130	-115.117253	Music Venue
18	3	36.033679	-115.125374	Mail Again Envelopes	36.039905	-115.118353	Art Gallery

Arts and entertainment venues:

- House #1: plenty of arts and entertainment venues
- House #2: average of arts and entertainment venues
- House #3: some arts and entertainment venues

Step 3: Results

Let's consolidate all the results in a single map

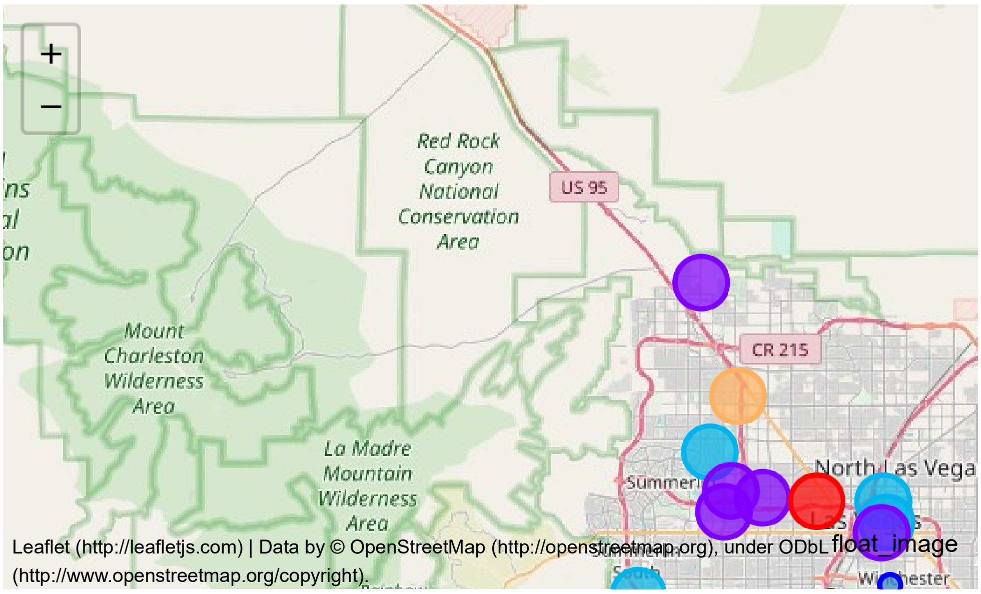
Legend of the map:

- the houses are blue circles
- the job location is the green circle
- the clusters of venues are the bubbles

In [144]:

```
# create map of Las Vegas
latitude= 36.1662859
longitude= -115.149225

map_lv_one = folium.Map(location=[latitude, longitude], zoom_start=10)

# add houses markers to map
for lat, lng, label in zip(lvhouses['Latitude'], lvhouses['Longitude'],lvhouses
['Address']):
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius=6,
        popup=label,
        color='blue',
        fill=True,
        fill_color='#3186cc',
        fill_opacity=0.7,
        parse_html=False).add_to(map_lv_one)

data = [['Caesars Palace Casino', '36.1161685', '-115.1766877']]
lv_job_location = pd.DataFrame(data, columns = ['Name', 'Latitude', 'Longitude
'])

# add job location (Caesars palace casino) marker to map
for lat, lng, label in zip(lv_job_location['Latitude'], lv_job_location['Longit
ude'],lv_job_location['Name']):
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius=6,
        popup=label,
        color='green',
        fill=True,
        fill_color='#6bef7f',
        fill_opacity=0.7,
        parse_html=False).add_to(map_lv_one)

# set color scheme for the clusters
kclusters=5
x = np.arange(kclusters)
ys = [i+x+(i*x)**2 for i in range(kclusters)]
colors_array = cm.rainbow(np.linspace(0, 1, len(ys)))
rainbow = [colors.rgb2hex(i) for i in colors_array]

# add markers to the map
markers_colors = []
for lat, lon, poi, cluster in zip(lv_merged['Latitude'], lv_merged['Longitude
'], lv_merged['Neighborhood'], lv_merged['Cluster Labels']):
    label = folium.Popup(str(poi) + ' Cluster ' + str(cluster), parse_html=Tru
e)
    folium.CircleMarker(
        [lat, lon],
        radius=15,
        popup=label,
        color=rainbow[cluster-1],
        fill=True,
        fill_color=rainbow[cluster-1],
        fill_opacity=0.7).add_to(map_lv_one)

# adds tool to the top right
from folium.plugins import MeasureControl
map_lv_one.add_child(MeasureControl())

# measurement ruler icon tool to measure distances in map
from folium.plugins import FloatImage
url = ('https://media.licdn.com/mpr/mpr/shrinknp_100_100/AAEAAQAAAAAAAAlgAAAAJG
E3OTA4YTdlLTkzZjUtNDFjYy1iZThlLWQ5OTNkYzlhNzM4OQ.jpg')
FloatImage(url, bottom=5, left=85).add_to(map_lv_one)
```

Out[144]:


+
−
Red Rock Canyon National Conservation Area
US 95
CR 215
Mount Charleston Wilderness Area
La Madre Mountain Wilderness Area
Summerlin
North Las Vega
float_image
Winchester
Leaflet (http://leafletjs.com) | Data by © OpenStreetMap (http://openstreetmap.org), under ODbL (http://www.openstreetmap.org/copyright).

Step 4: Problem resolution

Using the above map I am going to evaluate:

- for House #1: Huntridge - cluster 1
- for House #2: Whitney Ranch - cluster 1
- for House #3: Green Valley South - cluster 1

In [145]:

```
## cn is the cluster number to explore
cn = 1
lv_merged.loc[lv_merged['Cluster Labels'] == cn, lv_merged.columns[[1] + list(r
ange(5, lv_merged.shape[1]))]]
```

Out[145]:

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue
0	Blue Diamond	Skate Park	Park	Residential Building (Apartment / Condo)	Real Estate Office	Playground	Other Great Outdoors	Non-Profit	History Museum
5	Calico Ridge	Park	Skate Park	Residential Building (Apartment / Condo)	Real Estate Office	Playground	Other Great Outdoors	Non-Profit	History Museum
8	Gibson Springs	Park	Skate Park	Residential Building (Apartment / Condo)	Real Estate Office	Playground	Other Great Outdoors	Non-Profit	History Museum
11	Green Valley South	Park	Skate Park	Residential Building (Apartment / Condo)	Real Estate Office	Playground	Other Great Outdoors	Non-Profit	History Museum
13	Lake Las Vegas	Park	Skate Park	Residential Building (Apartment / Condo)	Real Estate Office	Playground	Other Great Outdoors	Non-Profit	History Museum
20	River Mountain	Park	Skate Park	Residential Building (Apartment / Condo)	Real Estate Office	Playground	Other Great Outdoors	Non-Profit	History Museum
21	Townsite	Park	Skate Park	Residential Building (Apartment / Condo)	Real Estate Office	Playground	Other Great Outdoors	Non-Profit	History Museum
23	Whitney Ranch	Park	Skate Park	Residential Building (Apartment / Condo)	Real Estate Office	Playground	Other Great Outdoors	Non-Profit	History Museum
25	Buffalo	Park	Skate Park	Residential Building (Apartment / Condo)	Real Estate Office	Playground	Other Great Outdoors	Non-Profit	History Museum
27	Charleston Heights	Park	Skate Park	Residential Building (Apartment / Condo)	Real Estate Office	Playground	Other Great Outdoors	Non-Profit	History Museum
31	Huntridge	Park	Non-Profit	Skate Park	Residential Building (Apartment / Condo)	Real Estate Office	Playground	Other Great Outdoors	History Museum
36	Pioneer Park	Park	Skate Park	Residential Building (Apartment / Condo)	Real Estate Office	Playground	Other Great Outdoors	Non-Profit	History Museum
42	Tule Springs	Park	Skate Park	Residential Building (Apartment / Condo)	Real Estate Office	Playground	Other Great Outdoors	Non-Profit	History Museum

Step 5. Discussion / House Selection

I used the above map to evaluate the different potential houses to buy.

- *House #1*: the price USD429,995 is on the budget and it is only 3.6mi from job location (15 min). There are no parks, but both elementary and middle schools. From parents point of view (pov) there are plenty of shopping and arts and entertainment venues.
- *House #2*: the price USD345,000 is pretty low and much bigger than House #1, it is 7.1mi from job location but around the same time(19 min). There are 3 parks, but only elementary schools. From parents pov there are plenty of shopping venues and average arts and entertainment venues.
- *House #3*: the price USD519,900 is just a little bit over budget and around the same size like House #1. Distance from job location is 8mi but around the same time (18 min). There is 1 park and no schools nearby. Moreover not so many shopping and arts and entertainment venues.

The venues for all houses are as of Cluster 1 so no real difference.

House #3 is immediately out because an higher price tag gives less comfort and venues.
House #2 is comparable to House #1 even somehow better and the price tag is much lower so it is my favourite choice.

Comparing House #2 with Padova/Padua venues, Padova/Padua has more parks and playgrounds, some high schools but no elementary or middle schools around and the same level of arts and entertainment. Shopping venues are less that in House #2 area.

Step 6. Conclusions

I can conclude that venues in House #2 area are pretty much the same like in my Padova/Padua home area even more shopping venues for my wife happiness. It is not far from job location and in a good neighborhood.
House #2 is a perfect fit for me and my family.

```
In [ ]:
```