Investigating the Feasibilty of Opening Video Game Stores in St. Louis, MO, USA

Applied Data Science Capstone Week 5

Taher Hajilounezhad

November 2019

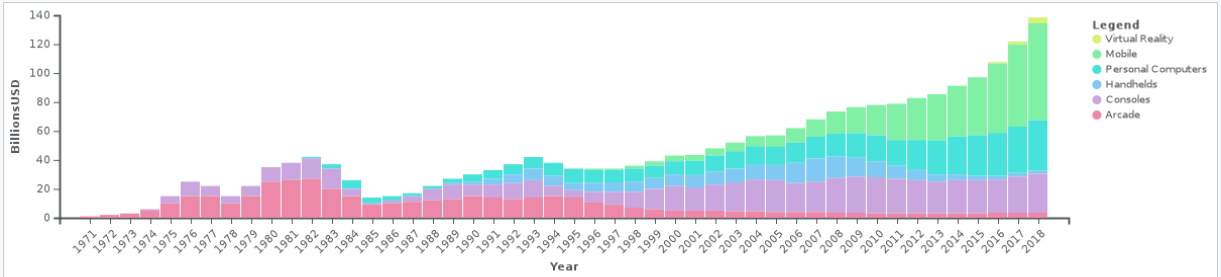
Introduction

St. Louisis an independent city (city not in a county) in the U.S. state of Missouri. It is the second largest city in the state of Missouri behind Kansas City. It is situated along the western bank of the Mississippi River, which forms the state line between Illinois and Missouri. The estimated 2018 population of the city proper was 302,838 and the bi-state metropolitan area was 2,804,724. Greater St. Louis is the largest metropolitan area in Missouri, second-largest in Illinois, 7th-largest in the Great Lakes Megalopolis, and the 20th-largest in the United

States.The architecture of St. Louis exhibits a variety of commercial, residential, and monumental architecture. St. Louis is known for the Gateway Arch, the tallest monument constructed in the United States at 630 feet (190 m). The Arch pays homage to Thomas Jefferson and St. Louis's position as the gateway to the West.

The computer and video-game industry has grown from focused markets to mainstream. They took in about 9.5 billion USD in the US in 2007, 11.7 billion in 2008, and 25.1 billion in 2010 (ESA annual report) and is estimated to be around 140 billion USD in 2018. Although, The games industry's shift from brick and mortar retail to digital downloads led to a severe sales decline at video game retailers, but retail game stores diversified its services by purchasing chains that repair wireless devices and expanding its trade-in program through which customers trade used games for credit towards new games. Other chain stores revamped its stores so customers would spend time playing games there. . It built a gaming arena for events and tournaments.

Although there are a few video game stores, but the increased interest in video gaming specially among teenagers or even younger childer opens a huge door to start new video game stores. A video game retail chain stores is interested to investigate possible locations for new video game stores.



Global revenues of the video game industry from 1971 to 2018, not adjusted for inflation.[40]

Business Problem

This capstone project aims to study the feasible locations in St. Louis, MO to open new video game stores. Various data science techniques like clustering by machine learning is employed to provide a comprehensive study to suggest to the city hall what locations are best candidates for the video game stores.

The Target Audience

This project is primarily defined to look for mutiple alternatives the city of St. Louis has to open new department stores. This study can also be used by other chain retail video gaing companies to evaluate the investment opportunities.

Data

Following data are used to answer business questions:

- List of neighborhouds in St. Louis, MO. This determins the scope of project in the heart of Missouri

- Latitude and longitude of the neighborhouds to plot the map and to get the venue data

- Venue data specially data related to other local businesses. It will be uised for clustering the neigborhoods.

Source of Data

This page on wikipedia "https://en.wikipedia.org/wiki/List_of_neighborhoods_of_St._Louis" (https://en.wikipedia.org/wiki/List_of_neighborhoods_of_St._Louis") provides the list of neighborhoods in St. Louis, MO. Web scraping techniques will be employed to extract data from this page, using Python requests and Beautifulsoup package. Later, Python Geocoder will be used to obtain the geographical coordinates of the neighborhoods. Then, the Foursquare API is used to get the venue data. It will provide various categories of venue data including department stores. This will help us tackle our business problem. After that, data preparation and wrangling will be applied to finally buid a clustering machine learning model (K-means for this proeject). Finally, Folium will be used for map visualization.

Methodology

At first, the list of neighborhoods in St. Luois, MO, USA is obtained. I have used (https://en.wikipedia.org/wiki/List_of_neighborhoods_of_St._Louis#List_of_neighborhoods (https://en.wikipedia.org/wiki/List_of_neighborhoods_of_St._Louis#List_of_neighborhoods)) to access data. Later, web scrapping is performed by Pytho nrequests and beautiful soup packages to extract the list of neighborhood data. In order to be able to use Foursquare API, we need to get the geographical coordinates as latitude and longitude. Accordingly, Geocoder package is employed that enables converting addresses into coordinates. Then, data is populated into a dataframe. Folium package is used to visualize the neighborhoods in map. This is a double check of the accuracy of geographical data extracted by Geocoder.

Then, Foursquare API is used again to get the top 100 venues within the radius of 2000 meters. API calls are utilized to to Foursquare passing in the geographical coordinates of the neighborhoods within a loop. Foursquare will return the venue data in JSON format from which venue name, category, latitude and longitude is extracted. The the number of returned venues for each neighborhood is checked and the. The number of unique categories is also determined. Neighborhoods are grouped and the mean frequency of occurance of each unique venue category is calculated.

K-means clustering algorithm is applied on data with three number of clusters. It is a common unsupervised clustering technique and is a good fit to our type of project. NEighborhoods are clustered into 3 clusters based on their frequency of occurance for "Video Game Store". IT is clear from the results which neighborhoods have higher concentration of video game stores. These results are used to answer the buisness questions.

Results

K-means clustering results reveal that we can divide neighborhoods into three clusters based on the frequency of "video game store" occurance:

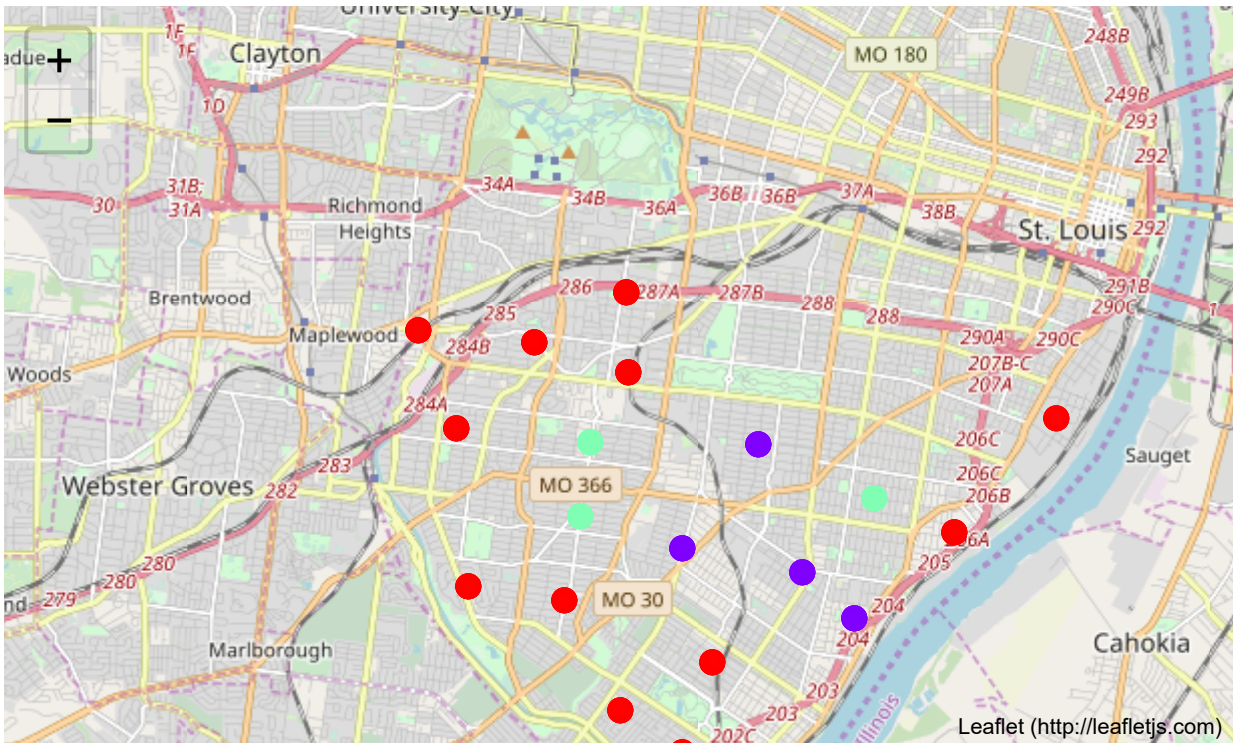
*** Clsuter 0: Neighborhoods with low number or without video game stores (red points)**

*** Clsuter 1: Neighborhoods with highest number of video game stores (purple points)**

*** Clsuter 2: Neighborhoods with medium number of video game stores (green points)**

In [263]:

Out[263]:



Discussion

Only there are 28 video game stores in St. Louis. They are mainly located in the center area of St. Louis. All video game stores are located in cluster 1& 2. Surprisingly, there is no video game stores in cluster 0. This is a great opportunity to open new department stores specially in cluster 0 where there are 51 neighborhoods. Nevertheless,

cluster 1 which is in the heart of the city and marked by purple clustering points, and spceially cluster 2 represented by green points also need some additional video game stores.

Conlcusion

This study represents that there is a huge investment success chance in cluster zero that occupies about 65% of the total neighborhoods of St. Louis and investers and chain video game stores are highly recommended to seriously investigate the feasibilty of opening new stores here.

1. Importing libraries

In [3]:
```
!pip install geocoder
```

```
Collecting geocoder
  Downloading https://files.pythonhosted.org/packages/4f/6b/13166c909ad2f2d76b929a4227c952630ebaf0d729f6317eb09cbceccbab/geocoder-1.38.1-py2.py3-none-any.whl (https://files.pythonhosted.org/packages/4f/6b/13166c909ad2f2d76b929a4227c952630ebaf0d729f6317eb09cbceccbab/geocoder-1.38.1-py2.py3-none-any.whl) (98kB)
Collecting ratelim (from geocoder)
  Downloading https://files.pythonhosted.org/packages/f2/98/7e6d147fd16a10a5f821db6e25f192265d6ecca3d82957a4fdd592cad49c/ratelim-0.1.6-py2.py3-none-any.whl (https://files.pythonhosted.org/packages/f2/98/7e6d147fd16a10a5f821db6e25f192265d6ecca3d82957a4fdd592cad49c/ratelim-0.1.6-py2.py3-none-any.whl)
Requirement already satisfied: six in c:\users\thnrf\anaconda3\lib\site-packages (from geocoder) (1.12.0)
Requirement already satisfied: requests in c:\users\thnrf\anaconda3\lib\site-packages (from geocoder) (2.21.0)
Requirement already satisfied: future in c:\users\thnrf\anaconda3\lib\site-packages (from geocoder) (0.17.1)
Requirement already satisfied: click in c:\users\thnrf\anaconda3\lib\site-packages (from geocoder) (7.0)
Requirement already satisfied: decorator in c:\users\thnrf\anaconda3\lib\site-packages (from ratelim->geocoder) (4.4.0)
Requirement already satisfied: urllib3<1.25,>=1.21.1 in c:\users\thnrf\anaconda3\lib\site-packages (from requests->geocoder) (1.24.1)
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in c:\users\thnrf\anaconda3\lib\site-packages (from requests->geocoder) (3.0.4)
Requirement already satisfied: idna<2.9,>=2.5 in c:\users\thnrf\anaconda3\lib\site-packages (from requests->geocoder) (2.8)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\thnrf\anaconda3\lib\site-packages (from requests->geocoder) (2019.6.16)
Installing collected packages: ratelim, geocoder
Successfully installed geocoder-1.38.1 ratelim-0.1.6
```

In [4]:
```
import numpy as np # library to handle data in a vectorized manner

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 geocoder # to get coordinates
from geopy.geocoders import Nominatim # convert an address into latitude and long

import requests # library to handle requests
from bs4 import BeautifulSoup # library to parse HTML and XML documents

from pandas.io.json import json_normalize # tranform JSON file into a pandas data

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

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

import folium # map rendering library

print("Libraries imported.")
```

```
Libraries imported.
```

2. Scraping data from Wikipedia page into a DataFrame

In [51]:
```
# send the GET request
data = requests.get("https://en.wikipedia.org/wiki/List_of_neighborhoods_of_St._
```

In [52]:
```
type(data)
```

Out[52]:
```
requests.models.Response
```

In [53]:
```
#page = requests.get("https://en.wikipedia.org/wiki/List_of_neighborhoods_of_St._
soup = BeautifulSoup(data.content, 'html.parser')
table = soup.find('tbody')
rows = table.select('tr')
row = [r.get_text() for r in rows]
```

In [54]:
```
df = pd.DataFrame(row)
df1 = df[0].str.split('\n', expand=True)
```

In [104]:
```
df2=[df1[3], df1[7], df1[11], df1[15]]
df3 = pd.concat(df2)
```

In [105]:
```
#results1=pd.DataFrame({'Neighborhood':[result]})
df4=pd.Series.to_frame(df3)
df4.head()
```

Out[105]:

	0
0	Carondelet
1	Patch
2	Holly Hills
3	Boulevard Heights
4	Bevo Mill

In [109]:
```
df4.columns=['Neighborhood']
df4.head()
```

Out[109]:

	Neighborhood
0	Carondelet
1	Patch
2	Holly Hills
3	Boulevard Heights
4	Bevo Mill

In [110]:
```
df4.shape
```

Out[110]: (80, 1)

3. Get the geographical coordinates

In [111]:
```
# define a function to get coordinates
def get_latlng(neighborhood):
    # initialize your variable to None
    lat_lng_coords = None
    # loop until you get the coordinates
    while(lat_lng_coords is None):
        g = geocoder.arcgis('{}, St. Luois, USA'.format(neighborhood))
        lat_lng_coords = g.latlng
    return lat_lng_coords
```

In [113]:
```
# call the function to get the coordinates, store in a new list using list compre
coords = [ get_latlng(neighborhood) for neighborhood in df4["Neighborhood"].toli
```

In [114]:
```
coords
```

Out[114]:
```
[[38.55639000000008, -90.26642999999996],
 [38.54353000000003, -90.26149999999996],
 [38.568520000000035, -90.26137999999997],
 [38.562020000000075, -90.27722999999997],
 [38.58386000000007, -90.26637999999997],
 [38.57680000000005, -90.28652999999997],
 [38.58824000000004, -90.28400999999997],
 [38.57871000000006, -90.30328999999995],
 [38.60009000000008, -90.30512999999996],
 [38.613060000000075, -90.31180999999998],
 [38.61140000000006, -90.29169999999993],
 [38.618310000000065, -90.27602999999993],
 [38.60752000000008, -90.27559999999994],
 [38.598190000000045, -90.28206999999998],
 [38.59769000000006, -90.25330999999994],
 [38.58063000000004, -90.24566999999996],
 [38.574520000000064, -90.23691999999994],
 [38.58598000000006, -90.21979999999996],
 [38.59063000000003, -90.23359999999997],
 [38.601320000000044, -90.20205999999996],
 [38.60418000000004, -90.20828999999998],
 [38.59962000000007, -90.21887999999996],
 [38.61038000000008, -90.21721999999994],
 [38.60890000000006, -90.22569999999996],
 [38.60385000000008, -90.23671999999999],
 [38.61298000000005, -90.23625999999996],
 [38.61217000000005, -90.24917999999997],
 [38.620960000000025, -90.25062999999994],
 [38.623860000000036, -90.24093999999997],
 [38.597660000000076, -90.23096999999996],
 [38.62018000000006, -90.22923999999995],
 [38.617560000000026, -90.21449999999999],
 [38.61573000000004, -90.20733999999999],
 [38.61534000000006, -90.20233999999994],
 [38.62549000000007, -90.19029999999998],
 [38.62975000000006, -90.20624999999995],
 [38.62736000000007, -90.22266999999994],
 [38.64170000000007, -90.25031999999999],
 [38.62695000000008, -90.25708999999995],
 [38.626130000000046, -90.27135999999996],
 [38.62733000000003, -90.27975999999995],
 [38.626990000000035, -90.29080999999996],
 [38.62217000000004, -90.30392999999998],
 [38.62985000000003, -90.30265999999995],
 [38.63954000000007, -90.30426999999997],
 [38.651940000000025, -90.29404999999997],
 [38.649470000000065, -90.27808999999996],
 [38.66110000000003, -90.28764999999999],
 [38.65662000000003, -90.27560999999997],
 [38.677410000000066, -90.27033999999998],
 [38.6772799233944, -90.50661997388784],
 [38.67318000000006, -90.25921999999997],
 [38.65787000000006, -90.25941999999998],
 [38.65417000000008, -90.25063999999998],
 [38.668330000000026, -90.25406999999996],
```

```
 [38.66266000000007, -90.23174999999998],
 [38.65930000000003, -90.24080999999995],
 [38.65005000000008, -90.24131999999997],
 [38.65215000000006, -90.21940999999998],
 [38.65148000000005, -90.20535999999998],
 [38.63909000000007, -90.19946999999996],
 [38.63691000000006, -90.18941999999998],
 [38.64892000000003, -90.19513999999998],
 [38.658250000000066, -90.18924999999996],
 [38.66176000000007, -90.20348999999999],
 [38.674000000000035, -90.20877999999999],
 [38.66745000000003, -90.21775999999994],
 [38.675560000000075, -90.22441999999995],
 [38.67864000000003, -90.23934999999994],
 [38.68772000000007, -90.26709999999997],
 [38.688860000000034, -90.24193999999994],
 [38.69856000000004, -90.25052999999997],
 [38.71572000000003, -90.24633999999998],
 [38.70553000000007, -90.23000999999994],
 [43.40261291448154, -84.61745256080366],
 [38.70607000000007, -90.25645999999995],
 [38.568702197298705, -90.40883688285439],
 [38.66876000000008, -90.27992999999998],
 [38.69925000000006, -90.21882999999997],
 [38.55803950607682, -90.29774340264048]]
```

In [115]:
```
# create temporary dataframe to populate the coordinates into Latitude and Longi
df_coords = pd.DataFrame(coords, columns=['Latitude', 'Longitude'])
```

In [116]:
```
# merge the coordinates into the original dataframe
df4['Latitude'] = df_coords['Latitude']
df4['Longitude'] = df_coords['Longitude']
```

In [118]:
```
# check the neighborhoods and the coordinates
print(df4.shape)
df4.head()
```

```
(80, 3)
```

Out[118]:

	Neighborhood	Latitude	Longitude
0	Carondelet	38.55639	-90.26643
1	Patch	38.54353	-90.26150
2	Holly Hills	38.56852	-90.26138
3	Boulevard Heights	38.56202	-90.27723
4	Bevo Mill	38.58386	-90.26638

In [119]:
```
# save the DataFrame as CSV file
df4.to_csv("df4.csv", index=False)
```

In [120]:
```
type(df4)
```

Out[120]:
```
pandas.core.frame.DataFrame
```

4. Create a map of St. Louis with neighborhoods superimposed on top

In [129]:
```
# get the coordinates of St. Louis
address = 'St. Louis, MO,  USA'

#geolocator = Nominatim(user_agent="my-application")
geolocator = Nominatim(user_agent="foursquare_agent")
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude
print('The geograpical coordinate of St. Louis, Missouri {}, {}.'.format(latitud
```

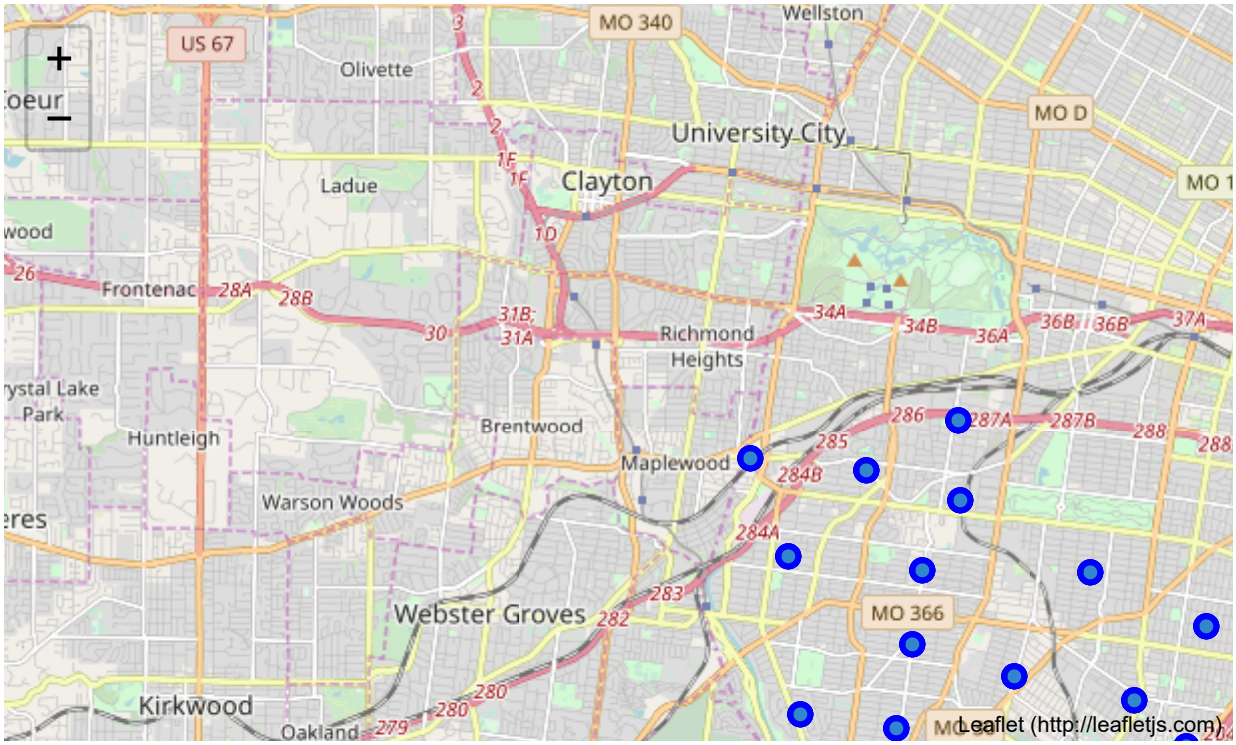
```
The geograpical coordinate of St. Louis, Missouri 38.6268039, -90.1994097.
```

In [131]:
```
# create map of St. Louis using latitude and longitude values
map_stl = folium.Map(location=[latitude, longitude], zoom_start=11)

# add markers to map
for lat, lng, neighborhood in zip(df4['Latitude'], df4['Longitude'], df4['Neighb
    label = '{}'.format(neighborhood)
    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).add_to(map_stl)

map_stl
```

Out[131]:



In [132]:
```
# save the map as HTML file
map_stl.save('map_stl.html')
```

5. Use the Foursquare API to explore the neighborhoods¶

In [231]:
```
CLIENT_ID = 'your Foursquare ID' # your Foursquare ID
CLIENT_SECRET = 'your Foursquare Secret' # your Foursquare Secret
VERSION = '20180605' # Foursquare API version

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

```
Your credentails:
CLIENT_ID: your Foursquare ID
CLIENT_SECRET:your Foursquare Secret
```

Now, let's get the top 100 venues that are within a radius of 2000 meters.

In [150]:
```
radius = 2000
LIMIT = 100

venues = []

for lat, long, neighborhood in zip(df4['Latitude'], df4['Longitude'], df4['Neighb

    # create the API request URL
    url = "https://api.foursquare.com/v2/venues/explore?client_id={}&client_secre
        CLIENT_ID,
        CLIENT_SECRET,
        VERSION,
        lat,
        long,
        radius,
        LIMIT)

    results = requests.get(url).json()["response"]['groups'][0]['items']

    # return only relevant information for each nearby venue
    for venue in results:
        venues.append((
            neighborhood,
            lat,
            long,
            venue['venue']['name'],
            venue['venue']['location']['lat'],
            venue['venue']['location']['lng'],
            venue['venue']['categories'][0]['name']))
```

In [152]:
```
# convert the venues list into a new DataFrame
venues_df = pd.DataFrame(venues)

# define the column names
venues_df.columns = ['Neighborhood', 'Latitude', 'Longitude', 'VenueName', 'Venue

print(venues_df.shape)
venues_df.head()
```

```
(7252, 7)
```

Out[152]:

	Neighborhood	Latitude	Longitude	VenueName	VenueLatitude	VenueLongitude	VenueCategory
0	Carondelet	38.55639	-90.26643	Starbucks	38.557545	-90.262024	Coffee Shop
1	Carondelet	38.55639	-90.26643	Stacked STL	38.549079	-90.262306	Burger Joint
2	Carondelet	38.55639	-90.26643	Carondelet Park Rec Complex	38.560436	-90.265931	Gym
3	Carondelet	38.55639	-90.26643	Perennial Artisan Ales	38.545582	-90.265769	Brewery
4	Carondelet	38.55639	-90.26643	Planet Fitness	38.564495	-90.259522	Gym / Fitness Center

Let's check how many venues were returned for each neighorhood

In [153]:
```
venues_df.groupby(["Neighborhood"]).count()
```

Out[153]:

Neighborhood	Latitude	Longitude	VenueName	VenueLatitude	VenueLongitude	VenueCa
Academy	100	100	100	100	100	
Baden	100	100	100	100	100	
Benton Park	64	64	64	64	64	
Benton Park West	100	100	100	100	100	
Bevo Mill	100	100	100	100	100	
Botanical Heights	100	100	100	100	100	
Boulevard Heights	80	80	80	80	80	
Carondelet	76	76	76	76	76	
Carr Square	76	76	76	76	76	
Central West End	74	74	74	74	74	

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

In [154]:
```
print('There are {} uniques categories.'.format(len(venues_df['VenueCategory'].u
```

```
There are 172 uniques categories.
```

In [158]:
```
# print out the list of categories
venues_df['VenueCategory'].unique()
```

Out[158]:
```
array(['Coffee Shop', 'Burger Joint', 'Gym', 'Brewery',
       'Gym / Fitness Center', 'Chinese Restaurant', 'Theater', 'Bakery',
       'Hardware Store', 'American Restaurant', 'Salon / Barbershop',
       'Pizza Place', 'Bar', 'Mexican Restaurant', 'Clothing Store',
       'ATM', 'Gay Bar', 'Video Store', 'Mobile Phone Shop',
       'Supermarket', 'Dive Bar', 'Park', 'Discount Store', 'Pub',
       'Fast Food Restaurant', 'Convenience Store', 'Skating Rink',
       'Grocery Store', 'Automotive Shop', 'Pharmacy', 'Gas Station',
       'Pet Store', 'Dog Run', 'Deli / Bodega', 'Intersection',
       'Baseball Field', 'Smoke Shop', 'Bus Station',
       'Construction & Landscaping', 'Rental Car Location',
       'Middle Eastern Restaurant', 'Diner', 'Karaoke Bar', 'Sports Bar',
       'Nightclub', 'Locksmith', 'Casino', 'Lounge', 'Cocktail Bar',
       'Gastropub', 'Hotel', 'Steakhouse', 'Italian Restaurant',
       'Breakfast Spot', 'Trail', 'Museum', 'Taco Place', 'Arcade',
       'Ice Cream Shop', 'Food', 'Restaurant', 'Zoo', 'German Restaurant',
       'Jewelry Store', 'Factory', 'Greek Restaurant', 'Wings Joint',
       'Café', 'BBQ Joint', 'Miscellaneous Shop', 'Gift Shop',
       'Sandwich Place', 'Beach', 'Eastern European Restaurant',
       'Indian Restaurant', 'Shoe Store', 'Cosmetics Shop', 'Butcher',
       'Massage Studio', 'Vietnamese Restaurant', 'Art Gallery',
       'Video Game Store', 'Big Box Store', 'Fried Chicken Joint',
       'Department Store', 'Optical Shop', 'Spa',
       'Paper / Office Supplies Store', 'Cajun / Creole Restaurant',
       'Thrift / Vintage Store', 'Bistro', 'Record Shop', 'Tanning Salon',
       'Donut Shop', 'Noodle House', 'Yoga Studio', 'Supplement Shop',
       "Women's Store", 'Furniture / Home Store', 'Playground',
       'Garden Center', 'Asian Restaurant', 'Antique Shop',
       'Smoothie Shop', 'Scenic Lookout', 'Dessert Shop',
       'Weight Loss Center', 'Bowling Alley', 'Arts & Crafts Store',
       'Beer Bar', 'Food & Drink Shop', 'Candy Store', 'Gourmet Shop',
       'Concert Hall', 'Boutique', 'Pet Café', 'New American Restaurant',
       'Pool Hall', 'Electronics Store', 'Pie Shop', 'Hobby Shop',
       'Sushi Restaurant', 'Accessories Store', 'Garden',
       'Farmers Market', 'Fabric Shop', 'Pool',
       'Mediterranean Restaurant', 'Warehouse Store', 'Tapas Restaurant',
       'Liquor Store', 'Science Museum', 'Movie Theater', 'Planetarium',
       'Botanical Garden', 'Food Truck', 'Wine Bar', 'Water Park',
       'Sculpture Garden', 'Southern / Soul Food Restaurant', 'Lake',
       'Golf Course', 'Tea Room', 'Beer Garden',
       'Vegetarian / Vegan Restaurant', 'Ethiopian Restaurant',
       'Japanese Restaurant', 'Thai Restaurant', 'Filipino Restaurant',
       'Hot Dog Joint', 'Market', 'Turkish Restaurant',
       'Seafood Restaurant', 'Toy / Game Store', 'Insurance Office',
       'Boat or Ferry', 'Rock Club', 'Bed & Breakfast', 'Whisky Bar',
       'Irish Pub', 'Stables', 'Beer Store', 'Event Space',
       'Brazilian Restaurant', 'Home Service', 'Music Venue',
       'Latin American Restaurant', 'Public Art', 'Plaza', 'Piano Bar',
       'Wine Shop', 'Outdoor Sculpture'], dtype=object)
```

In [259]:
```
# check if the results contain "Shopping Mall"
"Video Game Store" in venues_df['VenueCategory'].unique()
```

Out[259]:
```
True
```

6. Analyze Each Neighborhood

In [165]:
```
# one hot encoding
stl_onehot = pd.get_dummies(venues_df[['VenueCategory']], prefix="", prefix_sep='

# add neighborhood column back to dataframe
stl_onehot['Neighborhoods'] = venues_df['Neighborhood']

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

print(stl_onehot.shape)
stl_onehot.head()
```

```
(7252, 173)
```

Out[165]:

	Neighborhoods	ATM	Accessories Store	American Restaurant	Antique Shop	Arcade	Art Gallery	Arts & Crafts Store	Asian Restaurant	Au
0	Carondelet	0	0	0	0	0	0	0	0	
1	Carondelet	0	0	0	0	0	0	0	0	
2	Carondelet	0	0	0	0	0	0	0	0	
3	Carondelet	0	0	0	0	0	0	0	0	
4	Carondelet	0	0	0	0	0	0	0	0	

In [230]:
```
print('There are totally {} Department Stores in St. Louis.'.format(sum(stl_oneh
```

```
There are totally 8 Department Stores in St. Louis.
```

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

In [166]:
```
stl_grouped = stl_onehot.groupby(["Neighborhoods"]).mean().reset_index()

print(stl_grouped.shape)
stl_grouped
```

0000	0.020000	0.00	0.000000	0.030000	0.02	0.030000	0.00	0.02	0.020000	0.02000
0000	0.000000	0.00	0.000000	0.020000	0.01	0.010000	0.01	0.00	0.000000	0.02000
5625	0.031250	0.00	0.015625	0.062500	0.00	0.000000	0.00	0.00	0.031250	0.00000
0000	0.000000	0.00	0.000000	0.020000	0.01	0.010000	0.01	0.00	0.000000	0.02000
0000	0.020000	0.00	0.000000	0.010000	0.03	0.020000	0.00	0.01	0.000000	0.01000
0000	0.050000	0.00	0.000000	0.020000	0.00	0.000000	0.00	0.00	0.010000	0.01000
0000	0.020000	0.00	0.000000	0.020000	0.01	0.000000	0.00	0.00	0.010000	0.05000
0000	0.020000	0.00	0.000000	0.020000	0.01	0.000000	0.00	0.00	0.010000	0.05000
0000	0.020000	0.01	0.000000	0.010000	0.00	0.010000	0.00	0.01	0.020000	0.03000
0000	0.020000	0.00	0.000000	0.010000	0.03	0.020000	0.00	0.01	0.000000	0.01000
0000	0.050000	0.00	0.000000	0.010000	0.00	0.000000	0.00	0.00	0.000000	0.00000
0000	0.030000	0.00	0.000000	0.040000	0.02	0.030000	0.00	0.00	0.010000	0.01000
0000	0.012500	0.00	0.000000	0.037500	0.00	0.000000	0.00	0.00	0.000000	0.02500

In [232]:
```
len(stl_grouped[stl_grouped["Video Game Store"] > 0])
```

Out[232]: 28

Create a new DataFrame for Department Store data only

In [233]:
```
stl_dep = stl_grouped[["Neighborhoods","Video Game Store"]]
stl_dep.head()
```

Out[233]:

	Neighborhoods	Video Game Store
0	Academy	0.00
1	Baden	0.01
2	Benton Park	0.00
3	Benton Park West	0.00
4	Bevo Mill	0.03

7. Cluster Neighborhoods

Run k-means to cluster the neighborhoods in St. Louis into 3 clusters.

In [241]:
```
# set number of clusters
kclusters = 3

stl_clustering = stl_dep.drop(["Neighborhoods"], 1)

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

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

Out[241]:
```
array([0, 2, 0, 0, 1, 0, 0, 0, 0, 0])
```

In [242]:
```
# create a new dataframe that includes the cluster as well as the top 10 venues j
stl_merged = stl_dep.copy()

# add clustering labels
stl_merged["Cluster Labels"] = kmeans.labels_
```

In [243]:
```
stl_merged.rename(columns={"Neighborhoods": "Neighborhood"}, inplace=True)
stl_merged.head()
```

Out[243]:

	Neighborhood	Video Game Store	Cluster Labels
0	Academy	0.00	0
1	Baden	0.01	2
2	Benton Park	0.00	0
3	Benton Park West	0.00	0
4	Bevo Mill	0.03	1

In [244]:
```
# merge stl_grouped with df4 to add latitude/longitude for each neighborhood
stl_merged = stl_merged.join(df4.set_index("Neighborhood"), on="Neighborhood")

print(stl_merged.shape)
stl_merged.head() # check the last columns!
```

```
(79, 5)
```

Out[244]:

	Neighborhood	Video Game Store	Cluster Labels	Latitude	Longitude
0	Academy	0.00	0	38.61140	-90.29170
1	Baden	0.01	2	38.59819	-90.28207
2	Benton Park	0.00	0	38.54353	-90.26150
3	Benton Park West	0.00	0	38.61306	-90.31181
4	Bevo Mill	0.03	1	38.58386	-90.26638

In [245]:
```
# sort the results by Cluster Labels
print(stl_merged.shape)
stl_merged.sort_values(["Cluster Labels"], inplace=True)
stl_merged
```

42	Lindenwood Park	0.000000	0	38.60009	-90.30513
43	Marine Villa	0.000000	0	38.58598	-90.21980
44	Mark Twain	0.000000	0	38.61140	-90.29170
45	Mark Twain/I-70 Industrial	0.000000	0	38.61306	-90.31181
46	McKinley Heights	0.000000	0	38.56852	-90.26138
58	Princeton Heights	0.000000	0	38.57680	-90.28653
49	Near North Riverfront	0.000000	0	38.56202	-90.27723
57	Penrose	0.000000	0	38.60009	-90.30513
51	North Point	0.000000	0	38.60752	-90.27560
56	Peabody/Darst/Webbe	0.000000	0	38.60752	-90.27560
53	O'Fallon	0.000000	0	38.57871	-90.30329
61	Skinker/DeBaliviere	0.000000	0	38.57680	-90.28653
54	Old North St. Louis	0.000000	0	38.56852	-90.26138

Finally, let's visualize the resulting clusters

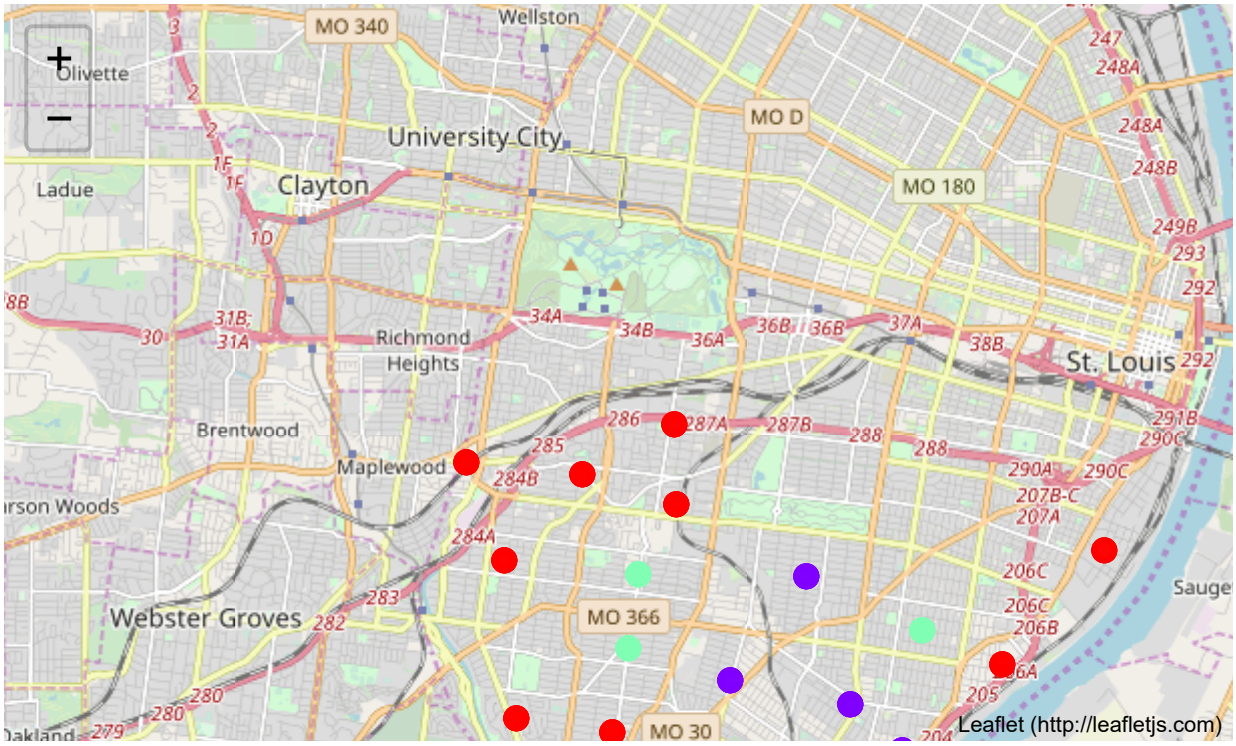
In [246]:
```
# create map
map_clusters = folium.Map(location=[latitude, longitude], zoom_start=11)

# 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(stl_merged['Latitude'], stl_merged['Longitude'
    label = folium.Popup(str(poi) + ' - Cluster ' + str(cluster), parse_html=Tru
    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[246]:



In [212]:
```
# save the map as HTML file
map_clusters.save('map_clusters.html')
```

8. Examine Clusters

Cluster 0

In [247]:
```
stl_merged.loc[stl_merged['Cluster Labels'] == 0]
```

Out[247]:

	Neighborhood	Video Game Store	Cluster Labels	Latitude	Longitude
0	Academy	0.0	0	38.61140	-90.29170
66	St. Louis Place	0.0	0	38.60132	-90.20206
30	Hamilton Heights	0.0	0	38.58598	-90.21980
31	Hi-Pointe	0.0	0	38.56202	-90.27723
32	Holly Hills	0.0	0	38.56852	-90.26138
65	St. Louis Hills	0.0	0	38.57871	-90.30329
64	Southwest Garden	0.0	0	38.60752	-90.27560
35	Kings Oak	0.0	0	38.60132	-90.20206
62	Soulard	0.0	0	38.55639	-90.26643
37	Kingsway West	0.0	0	38.61831	-90.27603
38	Kosciusko	0.0	0	38.60132	-90.20206
77	West End	0.0	0	38.57871	-90.30329
67	The Hill	0.0	0	38.61831	-90.27603
40	Lafayette Square	0.0	0	38.61831	-90.27603
42	Lindenwood Park	0.0	0	38.60009	-90.30513
43	Marine Villa	0.0	0	38.58598	-90.21980
44	Mark Twain	0.0	0	38.61140	-90.29170
45	Mark Twain/I-70 Industrial	0.0	0	38.61306	-90.31181
46	McKinley Heights	0.0	0	38.56852	-90.26138
58	Princeton Heights	0.0	0	38.57680	-90.28653
49	Near North Riverfront	0.0	0	38.56202	-90.27723
57	Penrose	0.0	0	38.60009	-90.30513
51	North Point	0.0	0	38.60752	-90.27560
56	Peabody/Darst/Webbe	0.0	0	38.60752	-90.27560
53	O'Fallon	0.0	0	38.57871	-90.30329
61	Skinker/DeBaliviere	0.0	0	38.57680	-90.28653
54	Old North St. Louis	0.0	0	38.56852	-90.26138
55	Patch	0.0	0	38.54353	-90.26150
25	Franz Park	0.0	0	38.56852	-90.26138
76	Wells/Goodfellow	0.0	0	38.61306	-90.31181
2	Benton Park	0.0	0	38.54353	-90.26150
3	Benton Park West	0.0	0	38.61306	-90.31181
5	Botanical Heights	0.0	0	38.57871	-90.30329
6	Boulevard Heights	0.0	0	38.56202	-90.27723

	Neighborhood	Video Game Store	Cluster Labels	Latitude	Longitude
7	Carondelet	0.0	0	38.55639	-90.26643
8	Carr Square	0.0	0	38.55639	-90.26643
9	Central West End	0.0	0	38.58598	-90.21980
10	Cheltenham	0.0	0	38.55639	-90.26643
11	Clayton/Tamm	0.0	0	38.54353	-90.26150
26	Gate District	0.0	0	38.61140	-90.29170
12	Clifton Heights	0.0	0	38.61140	-90.29170
14	Columbus Square	0.0	0	38.54353	-90.26150
15	Compton Heights	0.0	0	38.57680	-90.28653
74	Walnut Park East	0.0	0	38.61831	-90.27603
73	Visitation Park	0.0	0	38.60009	-90.30513
72	Vandeventer	0.0	0	38.58598	-90.21980
20	Ellendale	0.0	0	38.61306	-90.31181
69	Tiffany	0.0	0	38.60009	-90.30513
23	Fountain Park	0.0	0	38.60752	-90.27560
24	Fox Park	0.0	0	38.56202	-90.27723
13	College Hill	0.0	0	38.57680	-90.28653

In [248]:
```
len(stl_merged.loc[stl_merged['Cluster Labels'] == 0])
```

Out[248]: 51

Cluster 1

In [249]:
```
stl_merged.loc[stl_merged['Cluster Labels'] == 1]
```

Out[249]:

	Neighborhood	Video Game Store	Cluster Labels	Latitude	Longitude
71	Tower Grove South	0.020000	1	38.59769	-90.25331
59	Riverview	0.020000	1	38.59769	-90.25331
70	Tower Grove East	0.030000	1	38.58386	-90.26638
75	Walnut Park West	0.020000	1	38.58063	-90.24567
68	The Ville	0.021277	1	38.57452	-90.23692
78	Wydown/Skinker	0.030000	1	38.58386	-90.26638
48	Mount Pleasant	0.021277	1	38.57452	-90.23692
47	Midtown	0.021277	1	38.57452	-90.23692
4	Bevo Mill	0.030000	1	38.58386	-90.26638
36	Kingsway East	0.020000	1	38.59769	-90.25331
17	Downtown	0.020000	1	38.59769	-90.25331
33	Hyde Park	0.030000	1	38.58386	-90.26638
29	Greater Ville	0.020000	1	38.58063	-90.24567
18	Downtown West	0.020000	1	38.58063	-90.24567
27	Grand Center	0.021277	1	38.57452	-90.23692
19	Dutchtown	0.020000	1	38.58063	-90.24567

In [250]:
```
len(stl_merged.loc[stl_merged['Cluster Labels'] == 1])
```

Out[250]: 16

Cluster 2

In [254]:
```
stl_merged.loc[stl_merged['Cluster Labels'] == 2]
```

Out[254]:

	Neighborhood	Video Game Store	Cluster Labels	Latitude	Longitude
50	North Hampton	0.01	2	38.59819	-90.28207
1	Baden	0.01	2	38.59819	-90.28207
16	DeBaliviere Place	0.01	2	38.58824	-90.28401
28	Gravois Park	0.01	2	38.59063	-90.23360
22	Forest Park Southeast	0.01	2	38.59063	-90.23360
52	North Riverfront	0.01	2	38.59063	-90.23360
34	JeffVanderLou	0.01	2	38.59063	-90.23360
63	Southampton	0.01	2	38.58824	-90.28401
41	Lewis Place	0.01	2	38.59819	-90.28207
60	Shaw	0.01	2	38.58824	-90.28401
21	Fairground	0.01	2	38.58824	-90.28401
39	LaSalle Park	0.01	2	38.59819	-90.28207

In [258]:
```
len(stl_merged.loc[stl_merged['Cluster Labels'] == 2])
```

Out[258]: 12

In [ ]:
```

```