The Battle of Neighborhoods

Applied Data Science Capstone by IBM/Coursera

==========

Finding the best neighborhood in Houston to open a bar

==========

Author: Wenqian Shan

Date: 10/23/2021

---

0. Table of contents

1. Introduction

1.1. Background

Houston is the most populous city in the U.S. state of Texas, fourth-most populous city in the United States, most populous city in the Southern United States. Houston is also one of the most multicultural cities in U.S., making life in Houston a wonderful multicultural experience for all. Houston is also well-known for its great food and drinks.

As a young adult in Houston, I really enjoy hanging out with friends in different bars. Of course, there are hundreds of great bars here in Houston. It always be one of my dreams, that someday I can have my own bar opened here in the city. If so, where should I open it?

1.2. Problem Description

The objective of this project is to find the best neighborhood in Houston to open a bar using Foursquare location data. We will use the data science and machine learning tools we know to analyze the data and get a conclusion for the stakeholder to make a decision.

1.3. Target Audience

- Small business owners who want to open their own bars;
- Tourist who is looking for a bar in the city;
- Investors who want to invest a bar in Houston.

2. Data

The data we need:

1. A list of Houston neighborhood data
Data Source: https://en.wikipedia.org/wiki/List_of_neighborhoods_in_Houston
Description: This Wikipedia page contains a list of neighborhoods in Houston. We will scrape and clean the data, and read it into a pandas data frame.

2. Geographical coordinates of each neighborhoods
Data Source: using Google Geoencoding API
Description: The second data provides the geographical coordinates of each neighborhood in Houston.

3. Venue Data using Foursquare API
Data Source: https://foursquare.com/developers/apps
Description: From Foursquare API we can get the name, category, latitude, longitude for each venue.

3. Methodology

3.1 Installing packages and libraries

The packages I need are as follow:

```
# install requests to gain access to an URL
!pip install requests

# install beautifulsoup4 for web scraping
!pip install beautifulsoup4

# install folium for visualization
!pip install folium

# install sklearn
!pip install -U scikit-learn

# install kmeans for clustering
!pip install kmeans
```

The related libraries are here listed below:

```
# import all necessary libraries
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)
pd.set_option('precision', 9)

import json # library to handle JSON files

from bs4 import BeautifulSoup
import requests # library to handle requests
from pandas.io.json import json_normalize # tranform JSON file into a pandas dataframe

# Matplotlib
import matplotlib.pyplot as plt

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

# !conda install -c conda-forge folium=0.5.0 --yes
import folium # map rendering library
from folium.features import DivIcon
```

```
from IPython.display import display_html
```

3.2 Data Scraping

First, I scraped the Houston neighborhoods data using beautiful soup from a Wikipedia URL (data source 1 mentioned in 2. Data section) It includes neighborhood names and locations in the Houston city. There are 88 neighborhoods in Houston.

	Neighborhood	Location
1	Willowbrook	Northwest
2	Greater Greenspoint	North
3	Carverdale	Northwest
4	Fairbanks / Northwest Crossing	Northwest
5	Greater Inwood	Northwest
6	Acres Home	Northwest
7	Hidden Valley	North
8	Westbranch	West
9	Addicks / Park Ten	West
10	Spring Branch West	West

Second, I need to get the coordinates for each neighborhood.

I was trying the geopy library to get coordinates for each neighborhood and print them out the map. It turns out geopy couldn't get all the coordinates, and some of the coordinates are not at where they are supposed to be on map according to the "Location" from the data I grabbed.

So I decided to get all coordinates using Google Geoencoding API. Below is the function I implemented to get the coordinates for each neighborhood based on its name (name adding "Houston, TX", then becomes address).

```
def google_get_coordinates(api_key, address, verbose=False):
    try:
        url = 'https://maps.googleapis.com/maps/api/geocode/json?address={}&key={}'.format(address, api_key)
        response = requests.get(url).json()
        if verbose:
            print('Google Maps API JSON result =>', response)
        results = response['results']
        geographical_data = results[0]['geometry']['location'] # get geographical coordinates
        lat = geographical_data['lat']
        lon = geographical_data['lng']
        return [lat, lon]
    except:
        return [None, None]
```

3.3 Data Cleaning

It looks like, some of the neighborhoods have different names separated by special characters such as "/", "(" or ")". So I decided to extract the different names and get the coordinates from the average coordinates of different names (addresses). For example, the name "Mid-West (formerly Woodlake/Briarmeadow)", I got three names "Mid-West", "Woodlake" and "Briarmeadow". Then, the final coordinate of this neighborhood will be the average coordinate of these three addresses.

Below is the function I wrote to deal with the multiple name issue:

```
## average the coordinates with multiple names
def get_avg_coordinates(row_index=36, print_out=False):
    neighborhood = df.loc[row_index, 'Neighborhood']
    names = mySeperator(neighborhood)
    lats, lngs = [], []
    for name in names:
        address = neighborhood + ", Houston, TX"
        lat, lng = google_get_coordinates(api_key, address)
        lats.append(lat)
        lngs.append(lng)
    fn_lat = sum(lats)/len(lats)
    fn_lng = sum(lngs)/len(lngs)
    if print_out:
        print(neighborhood, fn_lat, fn_lng)
    return fn_lat, fn_lng
```

After looping around all 88 neighborhood I got from the data source 1, here is first ten rows of the updated data frame with coordinates for each neighborhood.

There was still one more problem in data. It seems two neighborhoods are overlapped on map, which means they shared the same coordinates.

10, Spring Branch West, West, 29.790847, -95.544630
86, Spring Branch East, Northwest, 29.790847, -95.544630

After double-checking the coordinates according to their locations and trying to put the direction word in front, I decided to manually replace the coordinate of "Spring Branch East" with the coordinate got from "East Spring Branch".

OK, here is the first 10 rows of the final data frame, which includes neighborhood names, locations, latitude and longitude for the center of each neighborhood.

	Neighborhood	Location	Latitude	Longitude
1	Willowbrook	Northwest	29.9558598	-95.5459787
2	Greater Greenspoint	North	29.9406789	-95.4139056
3	Carverdale	Northwest	29.8587078	-95.5451620
4	Fairbanks / Northwest Crossing	Northwest	29.8509122	-95.5154068
5	Greater Inwood	Northwest	29.8657387	-95.4804344
6	Acres Home	Northwest	29.8707190	-95.4365430
7	Hidden Valley	North	29.8896157	-95.4181974
8	Westbranch	West	29.8382464	-95.5520192
9	Addicks / Park Ten	West	29.8132687	-95.6454759
10	Spring Branch West	West	29.8028554	-95.4754992

3.4 Saving and Reading Intermediate Dataset¶

After all the data cleaning, now the dataset is ready for further analysis. In order to avoid to restart the previous data scraping and cleaning all over again, I saved the clean data into an CSV file “Houston_Neighborhood_Data.csv”. Later I can read this csv file into data frame to start the further analysis.

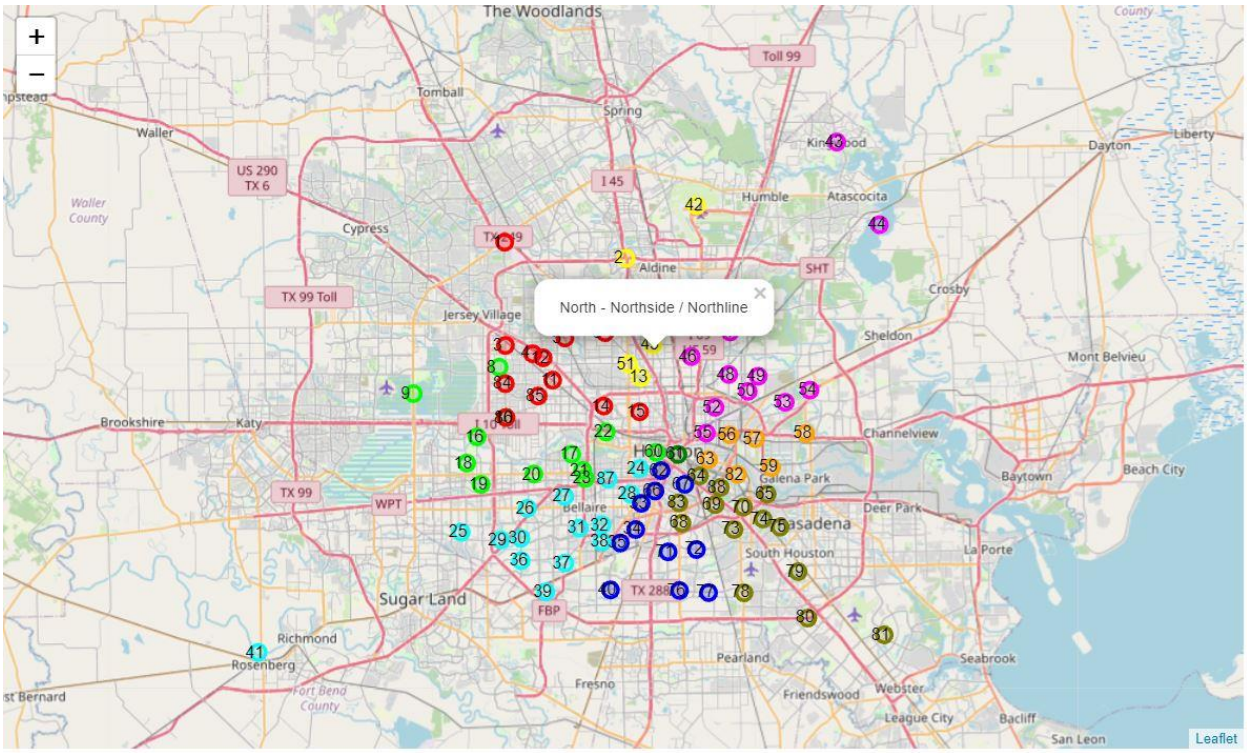
3.5 Interactive Data Visualization

I used the Folium interactive map to map out all 88 neighborhoods in Houston. The circles were filled up with different color based on their locations in the city, and labelled with their index numbers in data frame. The user can interact with the map to zoom-in, zoom-out, or click the circle to see the name and location for certain neighborhood.

Here is the code to implement it:

```
# get center coordinates of Houston and map it
lat_hou, lon_hou = google_get_coordinates(api_key, "Houston, TX")
map_houston = folium.Map(location=[lat_hou, lon_hou], zoom_start=10)


# add markers to map
for idx, lat, lng, location, neighborhood in zip(range(df.shape[0]),
                                                 df['Latitude'],
                                                 df['Longitude'],
                                                 df['Location'],
                                                 df['Neighborhood']):

    label_text = location + ' - ' + neighborhood
    label = folium.Popup(label_text)
    folium.Circle([lat, lng],
                  fill=True,
                  radius=800,
                  popup=label,
                  color=loc_color[location]).add_to(map_houston)

#     icon_html = '<div style="font-size: 10pt; color:black"><b>%s</b></div>'%(idx+1)
    icon_html = '<div style="font-size: 10pt; color:black"><t>%s</t></div>'%(idx+1)

#     print(icon_html, '\n')
    folium.map.Marker([lat, lng],
                      icon=DivIcon(
                          icon_size=(50,12),
                          icon_anchor=(10,10),
                          html=icon_html,
                          )
                      ).add_to(map_houston)
map_houston
```

Here is a screen shot of the interactive map. Pretty cool!



3.6 Getting Venues Using Foursquare

After getting all the coordinates, I need more information to analyze the neighborhoods. Using the tool, I learned from this course, I decided to use Foursquare to get all the nearby venues for

each neighborhood. I set the radius to 1 mile and up to 100 venues to count into my analysis for this study.

Below is the function I designed to get the information of venues for each neighborhood:

```
## test/default data: "westchase", 29.727756, -95.571662
def get_nearby_venues(name="westchase", lat=29.727756, lng=-95.571662, radius=1600, LIMIT=100):
    venues_list=[]
    # create the API request URL
    url_format = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={},{}&radius={}&limit=
    url = url_format.format(CLIENT_ID,CLIENT_SECRET, VERSION, lat, lng, radius, LIMIT)

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

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

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

    return nearby_venues
```

After looping through all neighborhoods, I get a data frame with 4168 rows and 7 columns. Here is the part of the data frame:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
4163	Lawndale / Wayside	29.7247703	-95.3112419	Burger King	29.736864200	-95.307390100	Fast Food Restaurant
4164	Lawndale / Wayside	29.7247703	-95.3112419	Taqueria Los Charros	29.737617000	-95.307508000	Mexican Restaurant
4165	Lawndale / Wayside	29.7247703	-95.3112419	Sprint Store	29.711409000	-95.313556000	Mobile Phone Shop
4166	Lawndale / Wayside	29.7247703	-95.3112419	WokkerTXRanger	29.711160597	-95.309757804	Food Truck
4167	Lawndale / Wayside	29.7247703	-95.3112419	Turimex International	29.735379346	-95.300747880	Bus Station

In this data frame, each venue too one row and each neighborhood took a number of rows.

3.7 Saving and Reading Intermediate Dataset Again

Then save this data frame into a CSV file “Houston_Venues.csv” in order to avoid re-doing all these data preprocessing again.

4. Data Analysis

4.1. Exploratory Data Analysis

To explore the insight of the dataset, there are some simple questions to answer:
Q1. How many venues in each neighborhood?
Use the code below to check it.

```
count_venues = houston_venues.groupby('Neighborhood').count()
count_venues.shape
```

```
(88, 6)
```

```
count_venues
```

Here is the top part of the result:

Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
Acres Home	4	4	4	4	4	4
Addicks / Park Ten	5	5	5	5	5	5
Afton Oaks / River Oaks	100	100	100	100	100	100
Alief	30	30	30	30	30	30
Astrodome Area	92	92	92	92	92	92
Braeburn	49	49	49	49	49	49
Braeswood	27	27	27	27	27	27
Brays Oaks (formerly Greater Fondren S.W.)	23	23	23	23	23	23
Briar Forest	88	88	88	88	88	88
Carverdale	20	20	20	20	20	20

Q2. How many unique venues are there in all neighborhood?

The code to check it:

```
hou_unique_venues = houston_venues['Venue Category'].unique()
hou_unique_venues = np.sort(hou_unique_venues)
# hou_unique_venues
print('There are {} unique venue categories.\n'.format(len(hou_unique_venues)))

print("The categories are:")
for idx, venue in enumerate(hou_unique_venues):
    print(idx+1, venue)

print('\nThere are {} unique venue categories.'.format(len(hou_unique_venues)))
```

The top part of the answer is:

```
There are 334 unique venue categories.

The categories are:
1 Accessories Store
2 Advertising Agency
3 Afghan Restaurant
4 African Restaurant
5 Airport
6 Airport Lounge
7 Airport Service
8 Airport Terminal
9 American Restaurant
10 Animal Shelter
11 Aquarium
12 Arcade
13 Argentinian Restaurant
14 Art Gallery
15 Art Museum
```

Q3. Are there any bars present in the venues?

```
"Bar" in hou_unique_venues
```

```
True
```

```
"Cocktail Bar" in hou_unique_venues
```

```
True
```

```
"Neighborhood" in hou_unique_venues
```

```
True
```

Q4. How many different bars or pubs in Houston?

In order to get all the bars or pubs, I implement a function to categorize the venue based on the keyword it contains:

```
## define a function to categorize venues into groups
def category_venue_by_keyword(category_name="no name", venue_list=[], word_list=[], exclude_list=[], print_out=True):
    out_list = []
    for venue in venue_list:
        flag = 0
        for word in exclude_list:
            if word in venue.lower():
                flag = 1
                break
        if flag:
            continue
        else:
            for word in word_list:
                if word in venue.lower():
                    out_list.append(venue)
                    break
    if print_out:
        print("Before removing {}, we have {} venues.".format(category_name, len(venue_list)))
        venue_list = [v for v in venue_list if v not in out_list]
        print("after removing {}, we have {} venues.".format(category_name, len(venue_list)))

        print("There are {} kinds of {}s in Houston.".format(len(out_list), category_name))
        print(houston_venues['Venue Category'].value_counts()[out_list])
    return venue_list, out_list
```

Using this function, I can answer this question now.

```
There are 16 kinds of bars in Houston.
Bar             66
Beer Bar         7
Cocktail Bar    13
Dive Bar        11
Gastropub        5
Gay Bar          3
Hookah Bar       1
Hotel Bar        3
Irish Pub        2
Juice Bar       12
Karaoke Bar      3
Pub             10
Sports Bar      23
Tiki Bar         1
Whisky Bar       2
Wine Bar        17
```

Q5. What other kinds of venues do we have?

I separated all venues into seven categories: bar, restaurant, food/drink, shopping, entertainment, outdoor and others.

Except from 16 kinds of bars:

There are 60 kinds of restaurants in Houston.

There are 36 kinds of food_drinks in Houston.

There are 75 kinds of shoppings in Houston.

There are 12 kinds of entertainments in Houston.
There are 8 kinds of outdoors in Houston.
And also 127 all other venues in Houston as well.
Q6. How many Neighborhoods for each venue?
Here is the method to know the answer and here is the top part of the answer.

```
group_venues_count = houston_venues.groupby('Venue Category').count()
group_venues_count.shape
```

```
(334, 6)
```

```
group_venues_count.head()
```

Venue Category	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude
Accessories Store	4	4	4	4	4	4
Advertising Agency	2	2	2	2	2	2
Afghan Restaurant	1	1	1	1	1	1
African Restaurant	3	3	3	3	3	3
Airport	2	2	2	2	2	2

4.2. Clustering the Neighborhoods

After I asked and answered some questions based on the data we have, we back to our major question. Which kind of neighborhood is the best choice to open a bar?
I decided to use K-mean clustering to further cluster the data and answer this question.

4.2.1. one hot coding

In order to do so, first, we need to use one hot code method to generate columns for each venue. Coincidentally, there is a venue called "Neighborhood". So we renamed the “Neighborhood” column as “Neighborhoods” for now.

```
## one hot encoding
to_onehot = pd.get_dummies(houston_venues[['Venue Category']], prefix="", prefix_sep="")
print(to_onehot.shape)

## add neighborhood column back to dataframe
df1 = houston_venues[["Neighborhood"]]
df1.columns = ["Neighborhoods"]
hou_venues_onehot = pd.concat([df1, to_onehot], axis=1).reindex(to_onehot.index)
print(hou_venues_onehot.shape)
print(hou_venues_onehot.columns[:15])

hou_venues_onehot.sort_values(["Neighborhoods"], inplace=True, ignore_index=True)
hou_venues_onehot.head()
```

Then we got the one-hot-code data frame with 4168 rows and 335 columns including 1 column for neighborhood names and 334 venue columns with all 0s and 1s.

	Neighborhoods	Accessories Store	Advertising Agency	Afghan Restaurant	African Restaurant	Airport	Airport Lounge	Airport Service	Airport Terminal	American Restaurant	Animal Shelter	Aquarium	Arcade	Argentinian Restaurant
0	Acres Home	0	0	0	0	0	0	0	0	0	0	0	0	0
1	Acres Home	0	0	0	0	0	0	0	0	0	0	0	0	0
2	Acres Home	0	0	0	0	0	0	0	0	0	0	0	0	0
3	Acres Home	0	0	0	0	0	0	0	0	0	0	0	0	0
4	Addicks / Park Ten	0	0	0	0	0	0	0	0	0	0	0	0	0

4.2.2. group venues into 7 categories

Next, for each neighborhood, we want to calculate the occurrence rate for each venue. In order to get an 88 rows and 335 columns dataset, in which each row is one neighborhood.

	Neighborhoods	Accessories Store	Advertising Agency	Afghan Restaurant	African Restaurant	Airport	Airport Lounge	Airport Service	Airport Terminal	American Restaurant	Animal Shelter	Aquarium	Arcade	Argentinian Restaurant
0	Acres Home	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.00000000	0.0	0.0	0.0	0.0
1	Addicks / Park Ten	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.00000000	0.0	0.0	0.0	0.0
2	Afton Oaks / River Oaks	0.0	0.0	0.0	0.01	0.0	0.0	0.0	0.0	0.05000000	0.0	0.0	0.0	0.0
3	Alief	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.00000000	0.0	0.0	0.0	0.0
4	Astrodome Area	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.02173913	0.0	0.0	0.0	0.0

Then, we want to group 334 venues into 7 categories based on the categories we have before, which will make our clustering easier to analyze.

```
## group data based on all_groups
## keep the Neighborhoods column
grouped_data = hou_venue_occur[["Neighborhoods"]]
for item in all_groups.keys():
    item_list = all_groups[item]
    item_data = hou_venue_occur.loc[:,item_list]
    grouped_data[item] = item_data.sum(axis=1)
# grouped_data.head()
grouped_data.shape
```

Then, we changed the column name “Neighborhood” column back to “Neighborhood”.
In this case, we end up with a dataset with 88 rows and 8 columns, and here is the top of it:

	Neighborhood	bar	restaurant	food_drink	shopping	entertainment	outdoor	others
0	Acres Home	0.000000000	0.25	0.000000000	0.250000000	0.000000000	0.000000000	0.500000000
1	Addicks / Park Ten	0.000000000	0.00	0.000000000	0.000000000	0.000000000	0.200000000	0.800000000
2	Afton Oaks / River Oaks	0.020000000	0.26	0.140000000	0.400000000	0.010000000	0.010000000	0.160000000
3	Alief	0.000000000	0.20	0.300000000	0.366666667	0.033333333	0.000000000	0.100000000
4	Astrodome Area	0.054347826	0.25	0.217391304	0.250000000	0.010869565	0.032608696	0.184782609

Finally, save the dataset for future use into CSV file “Hou_grouped_venues_occur.csv”.

4.2.3. Standardize data

The 7 categories would be the features used in the K-mean clustering. In order to make each feature to contribute equally in the modelling process, we need to standardize the data using sklearn library StandardScaler.

```
from sklearn.preprocessing import StandardScaler
raw_X = grouped_data.values[:,1:8]
raw_X = np.nan_to_num(raw_X)
print(raw_X[:5])
X = StandardScaler().fit_transform(raw_X)
X[:5]
```

Before standardizing: (first 5 rows)

```
[[0.0 0.25 0.0 0.25 0.0 0.0 0.5]
 [0.0 0.0 0.0 0.0 0.0 0.2 0.8]
 [0.02 0.26 0.14 0.39999999999999997 0.01 0.01 0.16]
 [0.0 0.19999999999999998 0.3 0.3666666666666667 0.03333333333333333 0.0
  0.1]
 [0.05434782608695652 0.25 0.21739130434782608 0.25 0.010869565217391304
  0.03260869565217391 0.18478260869565216]]
```

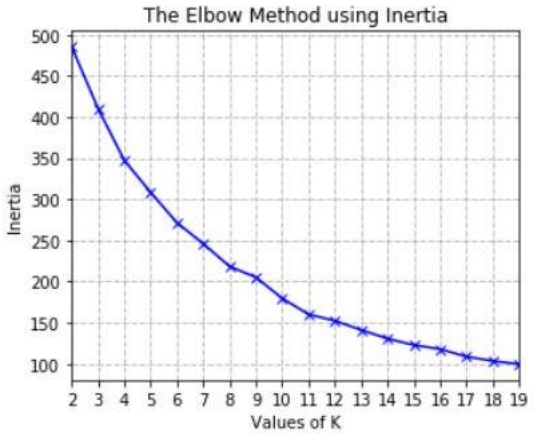
After standardizing: (first 5 rows)

```
array([[-0.68400715,  0.59405059, -1.86124911, -0.10197764, -0.65102942,
        -0.60849187,  1.07050802],
       [-0.68400715, -2.00598873, -1.86124911, -2.02271771, -0.65102942,
         4.05879481,  2.54591249],
       [-0.19904902,  0.69805217, -0.53540571,  1.0504664 , -0.04248043,
        -0.37512754, -0.60161704],
       [-0.68400715,  0.07404273,  0.97984389,  0.79436772,  1.3774672 ,
        -0.60849187, -0.89669794],
       [ 0.63381385,  0.59405059,  0.19751393, -0.10197764,  0.01043687,
         0.15247878, -0.47973581]])
```

4.3.4. Find the best K number

In order to get a good number of clusters to start with, I used Elbow Method to plot the inertia value of the K-mean clustering for 2 clusters to 20 clusters to see which number is the best choice.

Here is the plot of inertia verse values of K:



The plot is pretty smooth. I think k=4 can be a reasonable number of clusters.

4.3.5. Start modelling

I used the standardized data as input and use sklearn.cluster package KMeans library to model it.

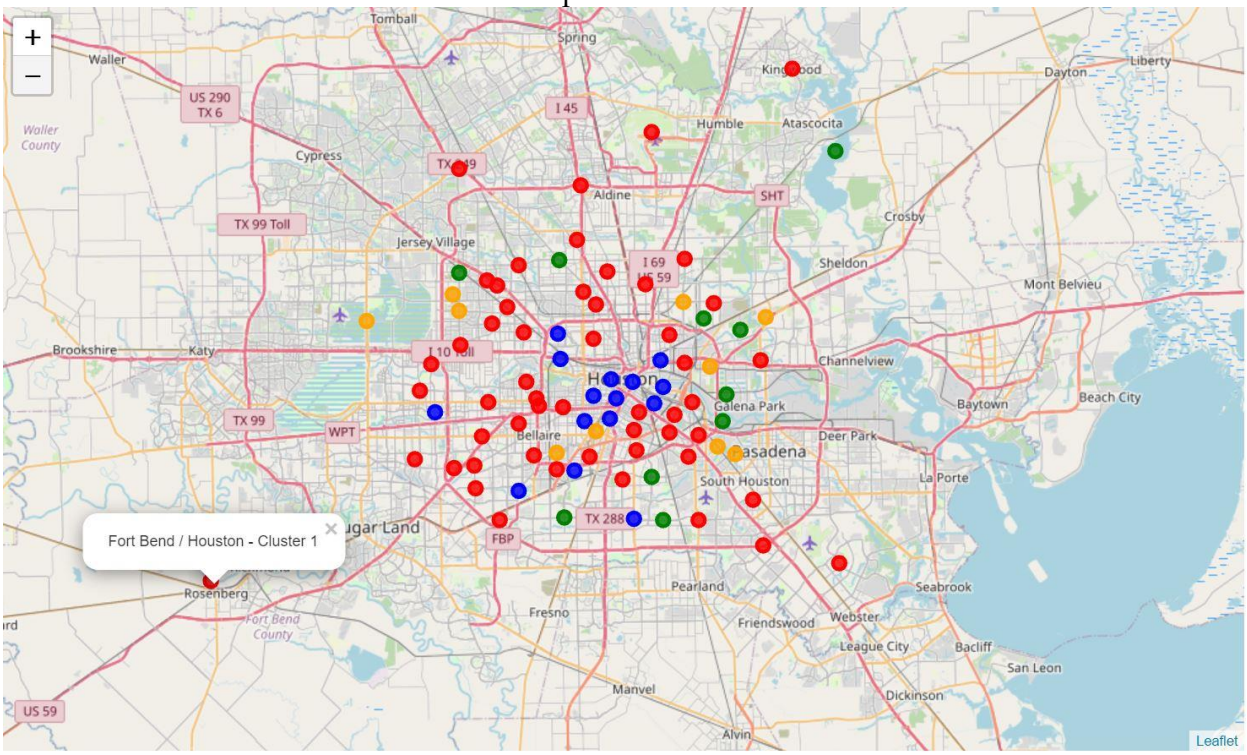
```
kmeans = KMeans(n_clusters=clusterNum, random_state=23).fit(X)
```

4.3.6. Visualize clusters using interactive visualization

In order to use Folium map tool to generate an interactive map, I combined the neighborhoods, locations, and coordinates information back with the occurrence rate and cluster labels. Here is the top part of the combined data:

	Neighborhood	Location	Latitude	Longitude	bar	restaurant	food_drink	shopping	entertainment	outdoor	others	Cluster Labels
1	Willowbrook	Northwest	29.9558598	-95.5459787	0.040000000	0.160000000	0.090000000	0.620000000	0.010000000	0.020000000	0.060000000	1
2	Greater Greenspoint	North	29.9406789	-95.4139056	0.032258065	0.322580645	0.241935484	0.225806452	0.016129032	0.016129032	0.145161290	1
3	Carverdale	Northwest	29.8587078	-95.5451620	0.000000000	0.050000000	0.300000000	0.100000000	0.000000000	0.000000000	0.550000000	3
4	Fairbanks / Northwest Crossing	Northwest	29.8509122	-95.5154068	0.035087719	0.210526316	0.210526316	0.438596491	0.000000000	0.052631579	0.052631579	1
5	Greater Inwood	Northwest	29.8657387	-95.4804344	0.000000000	0.166666667	0.233333333	0.300000000	0.000000000	0.000000000	0.300000000	1

Here is the screenshot of the interactive map:



5. Result and Discussion

5.1 Analysis of each Neighborhood Cluster

In order to analyze in details for each cluster, I implemented a function to separate the data for each cluster:

```
def get_cluster_df(cluster_num):
    cluster_temp = grouped_data.loc[grouped_data['Cluster Labels'] == cluster_num -1]
    cluster_df = pd.merge(df[['Neighborhood', 'Location']], cluster_temp, on='Neighborhood')
    return cluster_df
```

```
df_c1 = get_cluster_df(1)
df_c2 = get_cluster_df(2)
df_c3 = get_cluster_df(3)
df_c4 = get_cluster_df(4)
```

Here is an example of the top of the dataset for cluster-1 (label-0):

	Neighborhood	Location	bar	restaurant	food_drink	shopping	entertainment	outdoor	others	Cluster Labels
0	Lazybrook / Timbergrove	Northwest	0.097222222	0.263888889	0.180555556	0.222222222	0.000000000	0.041666667	0.194444444	0
1	Westchase	West	0.059523810	0.309523810	0.273809524	0.142857143	0.023809524	0.035714286	0.154761905	0
2	Washington Avenue Coalition / Memorial Park	West	0.066666667	0.116666667	0.100000000	0.150000000	0.033333333	0.083333333	0.450000000	0
3	Neartown / Montrose	Southwest	0.160000000	0.230000000	0.230000000	0.250000000	0.000000000	0.010000000	0.120000000	0
4	University Place	Southwest	0.090000000	0.230000000	0.190000000	0.310000000	0.010000000	0.010000000	0.160000000	0

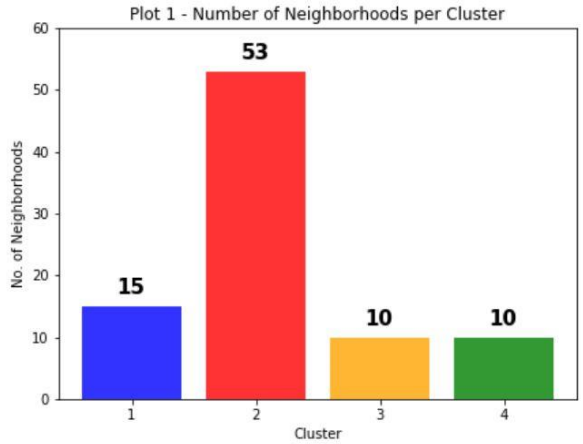
What is the locations for cluster-1 (label-0) neighborhoods?

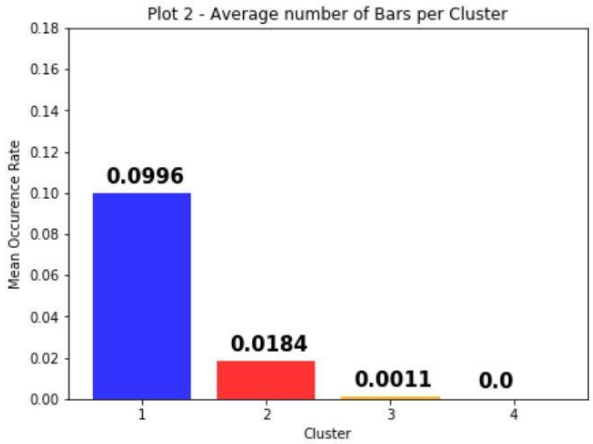
```
1 df_c1.groupby("Location").count()
```

Location	Neighborhood	bar	restaurant	food_drink	shopping	entertainment	outdoor	others	Cluster Labels
East	1	1	1	1	1	1	1	1	1
Northeast	1	1	1	1	1	1	1	1	1
Northwest	1	1	1	1	1	1	1	1	1
South	4	4	4	4	4	4	4	4	4
Southeast	1	1	1	1	1	1	1	1	1
Southwest	3	3	3	3	3	3	3	3	3
West	3	3	3	3	3	3	3	3	3
—	1	1	1	1	1	1	1	1	1

5.2. Bar Chats

We wanted to use bar chart to analyze the clustered data. In order to do so, we did two charts, one is the number of neighborhoods per cluster, the other is the average number of bars per cluster.
Here are the two plots we got:





As we can see in plot 1, cluster 2 is the one with the most number of the neighborhoods. In plot 2 we can see, cluster 1 is the one with the most number of bars.

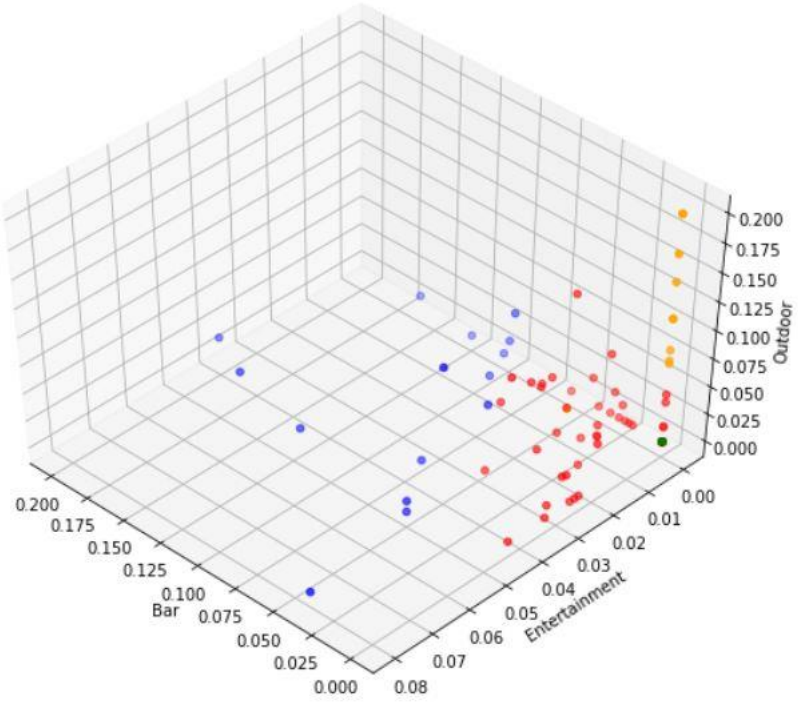
5.3. Find some correlations

In order to analyze the cluster data with some overview, I grouped data by their cluster labels and try to find some correlations between each features.

```
rank_data = grouped_data.groupby('Cluster Labels').mean().iloc[:, 0:7]
my_rank = rank_data/rank_data.sum()
my_rank["color"] = ["blue", "red", "yellow", "green"]
my_rank
```

Cluster Labels	bar	restaurant	food_drink	shopping	entertainment	outdoor	others	color
0	0.835757279	0.319180814	0.284749021	0.222287626	0.707742881	0.121221877	0.177137435	blue
1	0.154489424	0.360482102	0.335345633	0.356936229	0.233961188	0.093803019	0.128747214	red
2	0.009753296	0.194841619	0.335077134	0.193629666	0.058295931	0.784975104	0.236392642	yellow
3	0.000000000	0.125495465	0.044828211	0.227146479	0.000000000	0.000000000	0.457722708	green

It seems that there are some interesting correlations between bar, entertainment and outdoor according to the grouped data above. So I decided to generate a 3D plot to see these interesting correlations using Axes3D library in the mpl_toolkits.mplot3d package.
Here is the beautiful 3D plot:



6. Conclusion

Based on the analysis above, especially the clustering map and the 3D scatter plot, the conclusion is as follow:

- Interestingly, in all these venues, there is a positive correlation between the bars and entertainments, but negative correlations between bars and outdoor or other venues.

- The neighborhood cluster-1 (label-0, blue) includes the most amount of bars, as well as the most number of entertainments, but fewer outdoor and other venues; There are 15 neighborhoods in this cluster, and most of them are close to the center of the city. These neighborhoods are good, but to open a new bar, it might be quite competitive.
- The neighborhood cluster-2 (label-1, red) is the biggest group, which has 57 neighborhoods in it. There are plenty of restaurants, other food and drinks, shopping facilities, entertainments, and a few outdoor and other venues. Surprisingly, there is only a few numbers of bars in this cluster. The neighborhoods in this cluster could be a good choice to open a new bar.
- There are 10 neighborhoods in cluster-3 (label-2, yellow). In this cluster, there are only a few bars and entertainments, with a number of outdoor and other venues. The locations are quite far away from the city center.
- Lastly, in cluster-4 (label-3, green), it also has 10 neighborhoods. No bar, outdoor or entertainment can be found. If you don't want competition, then these neighborhoods are the best choice.

In conclusion, cluster2 is the best choice for more competitive new bars, while cluster4 can be the best for new bars don't want to compete with other bars.

7. Future Work

- Look into each cluster and analyzing them with more details;
- Try other clustering algorithms or consider different structure of the features used in cluster modelling;
- Use more than 100 venues and bigger radius in one neighborhood for analysis from Foursquare API.