Corner Finder

1. Introduction

According to the Census around 35.5 million Americans move each year to new cities. The most stressful part of moving is finding a new neighborhood you can call home. The goal of this project is to find your corner every where you move or travel. Finding your corner is targeted to people that are relocating to a new place and want to find the environment they already love in another city. This is done by comparing two cities and creating a cluster among neighborhoods. In this scenario neighborhoods in Miami, FL and SanFrancisco, CA are compared to find the most similar neighborhoods.

2. Data Acquisition and Cleaning

This project has two data phases. The first one is comprised of acquiring and cleaning data to create a data frame with Miami's and San Francisco's neighborhoods and coordinates. The second phase is connecting to the Foursquare API to obtained the venues data in the neighborhoods.

2.1 Neighborhood Data acquisition and cleaning

The neighborhood data for Miami and SanFrancisco was scraped from multiple websites utilizing BeautifulSoup and request packages. The goal on this step was to create a data frame with neighborhoods with its latitude and longitude. The Miami neighborhoods data was scraped from wikipedia, obtaining the neighborhood names populations and coordinates for 25 neighborhoods.

The SanFrancisco neighborhoods data was first scraped from the *San Francisco Burden of Disease & Injury Study* website, that was last updated on 2004. This website had the neighborhood information and corresponding zip codes. However, the website did not have the coordinates of each neighborhood. To obtained this coordinates the uszipcode packaged was utilized. The coordinates provided by the uszipcode packaged was incorrect for the 'Outer Richmond' neighborhood, the correct coordinates were obtained from wikipedia and changed in the data frame. After the creation of the Miami and San Francisco data frame in the previous step, both of the data frames were join creating one data frame.

2.2 Utilizing the Foursquare API

The Foursquare API offers a global database of venues data with more than 35 different end points. To connect to this API a ClientId and key are required, these can be obtained by registering in the Foursquare website. In this project the search endpoint was used by creating a get request with the following parameters:latitude, longitude, limit, radius and category. The latitude and longitude parameters goes in the location field as integers, other options to locate venues in the search are: street address, cross street, city, state, postal code and country. The limit parameter represents the number of results return, up to 50. The radius parameter represent the radius meter from the specified location in meters, the maximum is 100,000 meters. The category parameter is a category id to limit results; there are top level categories with sub categories.

Each request had static and fluctuating parameters. The static parameters were radius and limit. The limit was set to the maximum, returning 50 venues from the location provided and the radius parameter was set to 4023.36 meters (2.5 miles). The fluctuating parameters were latitude, longitude and the category. The latitude and longitude were from the Miami and San Francisco data frame. The category parameter was set to arts and entertainment, college and university,

events, food, nightlife, outdoor recreation, shop and services, residence, arts and entertainment. Those are all the top level parameters offered by the Foursquare API.

The request described above returns a JSON object with multiple venues, with the following information for each: id, name, location and categories. The id is a unique id for that venue. The name is the best known name for the venue. The location is an object that may contained the street address, cross street, city, state, postal code, latitude, longitude, country and distance. The categories contained the primary category of the venue with possible subcategories, in this case there is only one category in the request, returning one category at a time.The amount of venues in each location and category are recorded.. Below is an example of a request and a return:

Request made in Python:

```
#arts and entertainment, college and university, events,food,nightlife,outdoor recreation, shop and services, residence
categories = ['4d4b7104d754a06370d81259','4d4b7105d754a06372d81259','4d4b7102d754a06370d81259','4d4b7105d754a06374d8125
#
#function to get
def getNearbyVenues(names, latitudes, longitudes,categories, radius=6023.36):

    venues_list=[]

    for category in categories:
        for name, lat, lng in zip(names, latitudes, longitudes):

            # create the API request URL
            url = 'https://api.foursquare.com/v2/venues/search?&client_id={}&client_secret={}&v={}&ll={},{}&radius={}&l
                CLIENT_ID,
                CLIENT_SECRET,
                VERSION,
                lat,
                lng,
                radius,
                LIMIT,
                'browse',
                category)

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


            # return only relevant information for each nearby venue
            venues_list.append([{
                name,
                lat,
                lng,
                len(results['venues']),
                category
            }])

    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 Count',
                  'Venue Category']


    return(nearby_venues)
```

Return:


```
{
  "meta": {
    "code": 200,
    "requestId": "5ac51c7e6a607143d811cecb"
  },
  "response": {
    "venues": [
      {
        "id": '5642acf9493e51025cf4a7a5',
        "name": "Mr. Purple",
        "location": {
          "address": "180 Orchard St",
          "crossStreet": "btwn Houston & Stanton St",
          "lat": 40.721737442777209,
          "lng": -73.988036872829966,
          "labeledLatLngs": [
            {
              "label": "display",
              "lat": 40.721737442777209,
              "lng": -73.988036872829966
            }
          ],
          "distance": 8,
          "postalCode": "10002",
          "cc": "US",
          "city": "New York",
          "state": "NY",
          "country": "United States",
          "formattedAddress": [
            "180 Orchard St (btwn Houston & Stanton St)",
            "New York, NY 10002",
            "United States"
          ]
        },
        "categories": [
          {
            "id": "4bf58dd8d48988d1d5941735",
            "name": "Hotel Bar",
            "pluralName": "Hotel Bars",
            "shortName": "Hotel Bar",
            "icon": {
              "prefix": "https://ss3.4sqi.net/img/categories_v2/travel/hote
              "suffix": ".png"
            },
            "primary": true
          }
        ],
        "venuePage": {
          "id": "150747252"
        }
      }
    ]
  }
}
```

3. Methodology

The goal of this project is to compare neighborhoods in Miami and San Francisco and grouped them by similarity. The similarity are evaluated by the amount of venues in each category from the FourSquare API. The number of venues are then used to create a k-mean cluster. The k-mean cluster calculated with Python and the Scikit-learn library.

The scikit-learn library is one of the most used libraries in data science. This library contains Machine Leaning algorithms. The k-means algorithm in the library uses either Lloyd's or Elkan's algorithm. Lloyd's algorithm is named after Stuart P. Lloyd, the algorithm placed even distance in subset of Euclidean spaced and partition of theses spaces to uniformly convex cells (Lloyd, 1982) . The Elkan's algorithm take further the k-means by using the triangle inequality to avoid calculating all centroid distances.

3.1 K-mean cluster

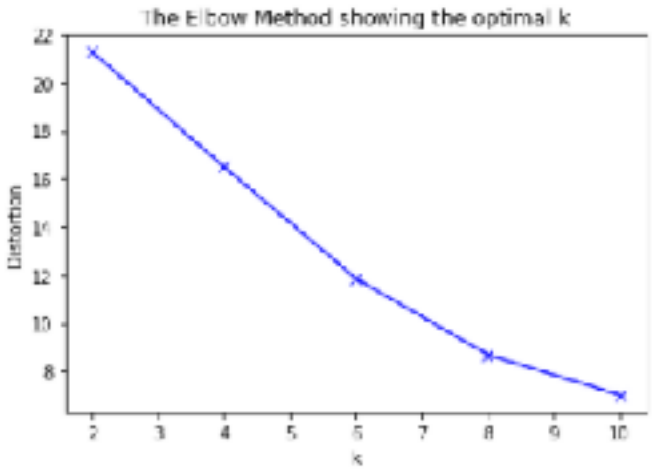
The k-mean cluster algorithm was chosen because of it ease of understanding, implementation and runtime. It also aligns with the goal of the project effectively grouping neighborhoods based on its similarities. The k-means cluster algorithm groups similar data into *k* clusters. The goal of this algorithm is to put individual in the cluster closest to the cluster's mean value (Sterling etl, 2018). This is done by selecting k initial centroids at random and iterating trough: assigning the individual to the closest cluster minimizing the inter cluster Euclidian distance and maximizing the outer cluster distance until the cluster converges.

The amount of *k* clusters was selected utilizing the elbow method. The elbow method is a heuristic method, implemented in mathematics based on the law of diminishing returns. In other words the clusters are chosen were adding an additional group (*k)* do not increase the better modeling of the data. The elbow methods works by plotting the number of clusters vs variance explained, the k means with decreasing variance is selected.

4. Results

The amount clustered neighborhoods as mentioned on the methodology above was selected based on an the elbow method. The elbow methods works by plotting the number of clusters vs variance explained, this results on an elbow shaped figure depicting the right *k.* The Elbow method created with the data is below:

Plot 1 : Distortion vs k



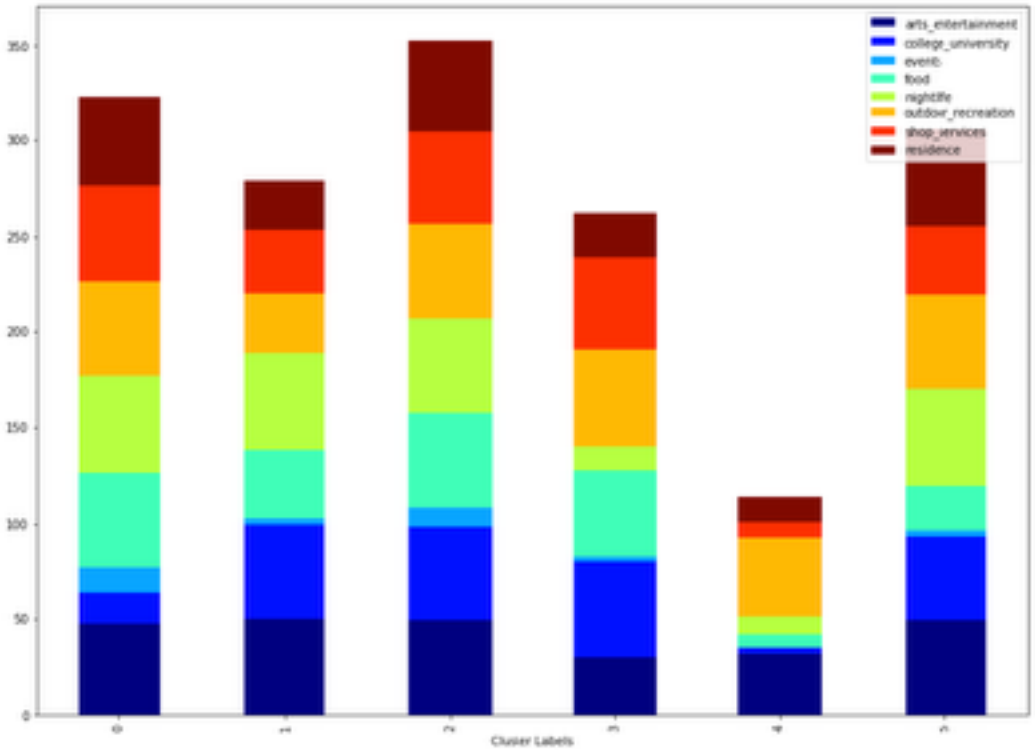
In the table above, the “elbow” or inflection point is shown on 6 and 8. In this case 6 is chosen as an attempt of having the least clusters possible. The neighborhood data is then clustered in 6 different groups each with different number of neighborhoods. The amount of neighborhoods per cluster is below:

Table 1 : Neighborhood Count by Cluster

Cluster Labels	Neighborhood
0	4
1	6
2	24
3	4
4	1
5	5

Each cluster has different amount of venues, the mean of the venues and the type of venues is showed in the box plot below:

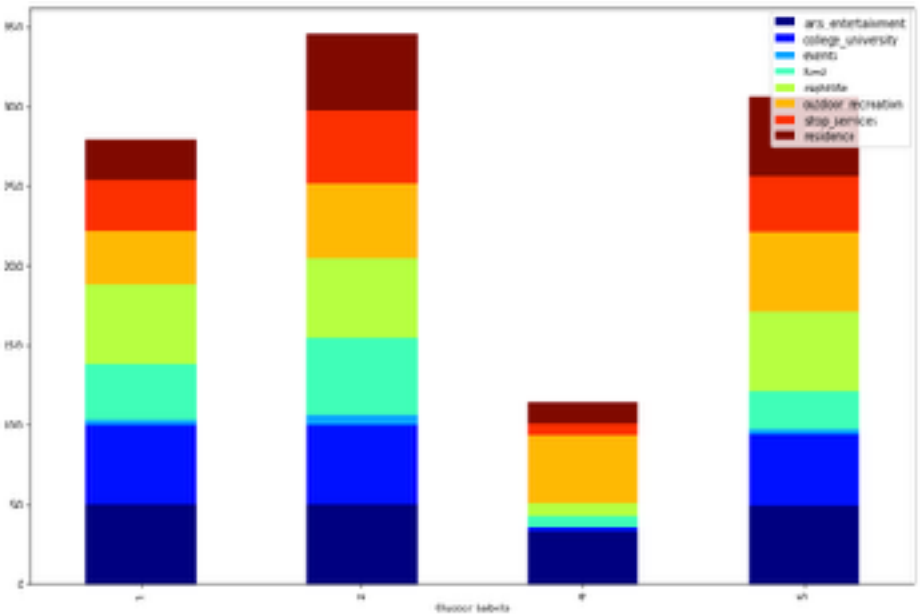
Plot 2: Neighborhoods Cluster for Miami and San Francisco vs means of each venue



4.1 Miami

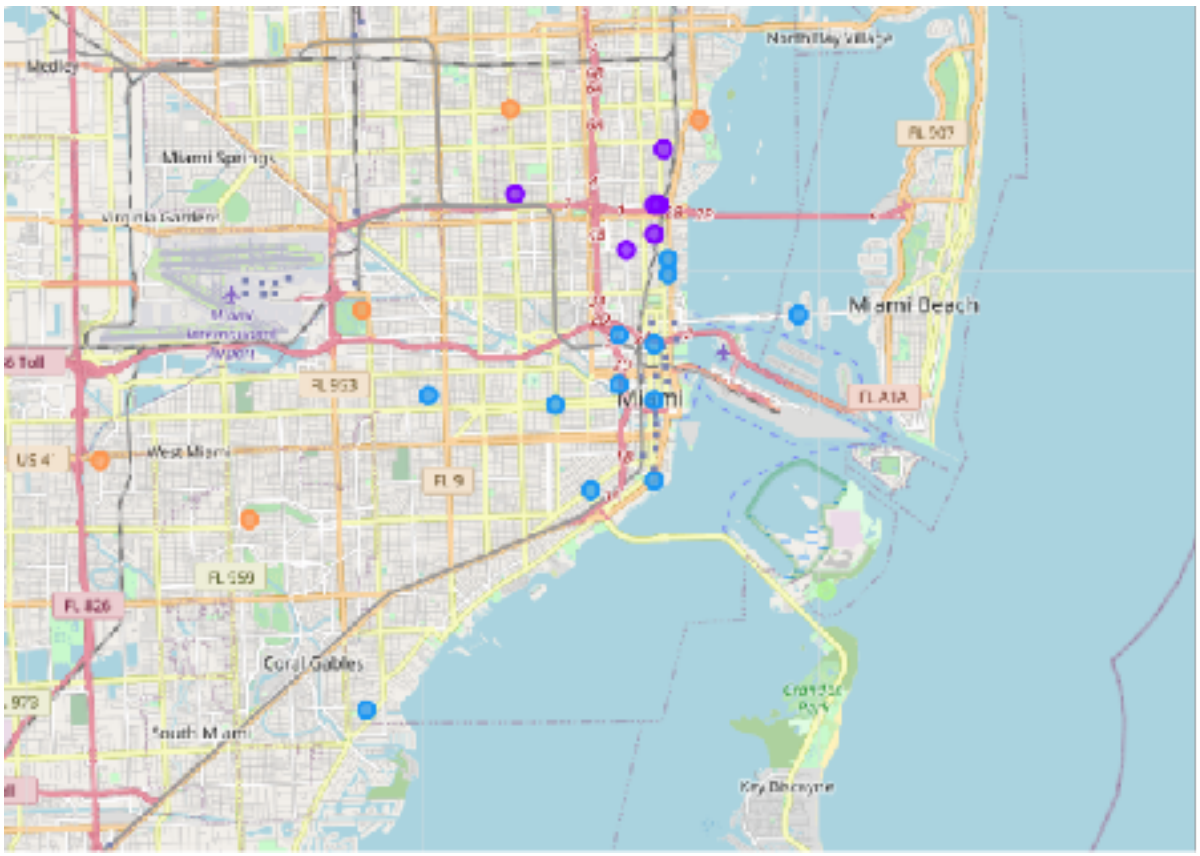
The Neighborhood Cluster only for Miami with each venue means is below:

Plot 3: Neighborhoods Cluster for Miami vs means of each venue



Miami, neighborhood clusters are below:

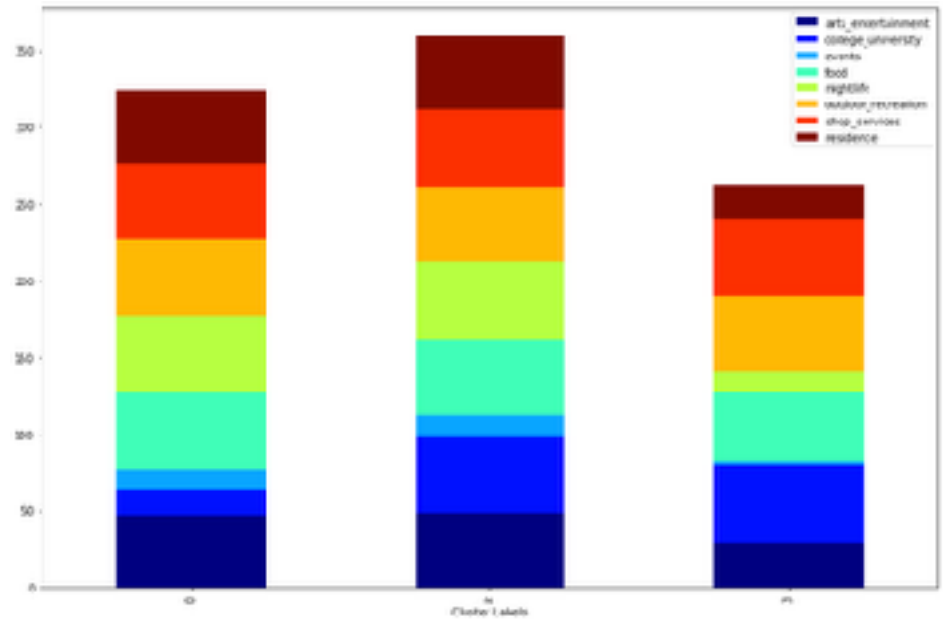
Map 1 : Miami neighborhoods cluster



4.2 San Francisco

The Neighborhood Cluster only for San Francisco with each venue means is below:

Plot 4: Neighborhoods Cluster for San Francisco vs means of each venue



San Francisco, neighborhood clusters are below:

Map 2 : San Francisco neighborhoods cluster

5. Discusion

The clustering exercised performed with Miami and San Francisco neighborhoods resulted in 6 grouped neighborhoods. Only one cluster - cluster 2, based on count and type of venues was found in Miami and San Francisco. Is important to mention that cluster 2, has the highest number of neighborhoods and venues among clusters. This can lead to the idea of a neighborhood archetype on every city , that is central to a variety of venues. The other clusters were in Miami or San Francisco. There are three different clusters besides cluster 2 in Miami. These clusters are cluster 1,4 and 5. Cluster 1 has an average of 275 venues in total and contains 6 different neighborhoods. Cluster 4 has an average of 125 venues in total and contains one neighborhood. Cluster 5 has an average of 325 venues in total and contains five different neighborhoods. In the other hand, there are two different clusters besides cluster 2 in San Francisco. These clusters are cluster 0 and 3. Cluster 0 has an average of 325 venues in total and contains four different neighborhoods. Cluster 3 has an average of 260 venues in total and contains four different neighborhoods.

This project shows that San Francisco and Miami are really different; but it contains a similar type of neighborhood that is pronounce in different locations of the city. These type of neighborhoods (cluster 2) is recurrent and in most cases next to each other. However is visible in San Francisco and Miami map, that in some instances , the cluster 2 type of neighborhood can be found outside the center of the city. It will be interesting to understand what are the factors that drives high diverse and count of venues location outside of the city center. Also, further investigating if the cluster 2 type of neighborhoods is in fact an archetype that city centers evolve to.