Capstone Project

The Battle of Neighborhoods

Best Neighborhood for retirement in South Florida

Machine learning allows for the creation of computational models capable of identifying patterns in multi-dimensional datasets. This project aims to utilize all Data Science Concepts learned in the IBM Data Science Professional Course i.e. define a Business Problem, the data that will be utilized and using that data, analyze the data using Machine Learning tools. In this project, we will go through all the steps to provide a solution that can be leveraged by the business stakeholders to make their decisions.

1. Introduction/Business Problem:

1.1. **Introduction:** South Florida, with its year-round warm weather, beaches, golf courses and flat geography, is an attractive option for retirement. First, the weather, older people hate old and they want to spend their retirement days in a warm and sunny place. And the Sunshine State is the perfect place for this. Florida is also cheaper and has no income tax. This is a major factor for senior citizens living on restricted income. There are more seniors moving to Florida than in some other parts of the USA. Florida is a very popular travel destination. And many people who come to live in South Florida are the ones who have been visiting it for many years. And they already know all the pros and cons of living in Florida.

1.2. **Problem:** For someone considering retiring to South Florida, there are dozens of cities, hundreds of neighborhoods available, and it can be a daunting task deciding where to move to.

1.3. **The objective** of this project is to is help someone who is unfamiliar with South Florida decide where to move to. Crime rate is the first factor to look at, since crime is prevalent in some communities. The programs filters all the cities of Florida, to just the cities in South Florida counties, and then it uses crime statists from FBI to mark the cities as green, yellow, orange and red, green being the safest cities. Next, since moving to a beach city is more desirable for most retirees, the program selects the safe-beach cities.

Finally, the project uses Foursquare location data and regional clustering of venue information to determine what might be the ‘best’ neighborhood out of the safe cities for retirement.

Through this project, we will find the most suitable location in South Florida for retirement.

2. Target Audience

This project is aimed towards retired or retiring people, or old people considering moving to South Florida. The analysis will provide vital information that can be used by the target audience.

3. Data - Sources, Acquisition and Cleaning:

The data that will be required will from multiple sources which will provide the list of cities in Florida, neighborhoods in these cities (via Wikipedia), the Geographical location of the neighborhoods (via GeoPy Geocoder package) and Venue data (via Foursquare). The Venue data will help find which neighborhood is best suitable for retirement.

3.1. **Data Sources**

- List of all the Florida Cities by counties is available at: https://dos.myflorida.com/library-archives/research/florida-information/government/local-resources/citycounty-list/counties/

  This Division manages the State Library and Archives, supports public libraries, directs records management services, and is the designated information resource provider for the state of Florida.

- Crime Data for all Florida cities is available at FBI Website. The FBI collects these data through the Uniform Crime Reporting (UCR) Program. The table provides the volume of violent crime as reported by city and town law enforcement agencies. The link for Florida is: https://ucr.fbi.gov/crime-in-the-u.s/2015/crime-in-the-u.s.-2015/tables/table-8/table-8-state-pieces/table_8_offenses_known_to_law_enforcement_florida_by_city_2015.xls

- List of best beach cities, as of June 2020, is available from https://wallethub.com/edu/best-beach-towns-to-live-in/36567/

- To get the coordinates given Neighborhood we use Geopy geocoder. Geopy makes it easy for Python developers to locate the coordinates of addresses, cities, countries, and landmarks across the globe using third-party geocoders and other data sources. The Geopy geocoder documentation can be found in the following link: https://geopy.readthedocs.io/en/stable/

- Venue Data using Foursquare

  The places by Foursquare API is a database of more than 105 million places worldwide and is going to be consulted for this project. To explore the cities, we will use the Venue Recommendation which returns a list of recommended venues near a certain location.

3.2. **Data Acquisition and Cleaning**

- List of all the Florida Cities by counties was read from https://dos.myflorida.com/library-archives/research/florida-information/government/local-resources/citycounty-list/counties/

  Data is in format that is suitable for analysis, its loaded to Pandas dataframe directly

```
11]: url='https://dos.myflorida.com/library-archives/research/florida-information/government
     dfflcities=pd.read_html(url, header=0)[0]
     dfflcities.columns = ['county','city','citytype']
     dfflcities.head()
```

ut[11]:

	county	city	citytype
0	Alachua	Alachua	city
1	Alachua	Archer	city
2	Alachua	Cross Creek	populated place
3	Alachua	Earleton	populated place
4	Alachua	Gainesville	city

Since we are interested in South Florida, filter cities for six South Florida Counties: Broward, Collier, Miami-Dade, Lee, Monroe and Palm Beach.

```
df_sfcities = dfflcities[((dfflcities['county']=='Broward')
df_sfcities.reset_index(drop=True, inplace=True)
df_sfcities['address']=df_sfcities['city']+', Florida'
df_sfcities.head(10)
```

]:

	county	city	citytype	address
0	Broward	Coconut Creek	city	Coconut Creek, Florida
1	Broward	Cooper City	city	Cooper City, Florida
2	Broward	Coral Springs	city	Coral Springs, Florida

- Crime Data for all Florida cities was read from https://ucr.fbi.gov/crime-in-the-u.s/2015/crime-in-the-u.s.-2015/tables/table-8/table-8-state-pieces/table_8_offenses_known_to_law_enforcement_florida_by_city_2015.xls

  Data is in format that is suitable for analysis, its loaded to Pandas Dataframe directly.

  The Data tables provides totals violent crimes for a city, and the total population, From that, crime-rate per 1000 was calculated.

  Since only the city name and crime-rate(per K) is of interest, its loaded to a final crime Data frame used for analysis.

:

```
flcrimeurl='https://ucr.fbi.gov/crime-in-the-u.s/2015/crime-in-the-u.s.-2015/tables/table-8/tabl
header = {
  "User-Agent": "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/5
  "X-Requested-With": "XMLHttpRequest"
}
r = requests.get(flcrimeurl, headers=header)
flcrime = pd.read_html(r.text)
dfflcrime = flcrime[0].dropna(axis=0, thresh=4)
dfflcrime['crimeperk']=round(1000*dfflcrime['Violentcrime']/dfflcrime['Population'],0)
dfflcrime.head()
```

8]:

	City	Population	Violentcrime	Murder andnonnegligentmanslaughter	Rape(reviseddefinition)1	Rape(legacydefiniti
0	Alachua	9687	33	0	4	
1	Altamonte Springs	42409	154	0	25	
2	Apalachicola	2287	1	0	0	
3	Apopka	48478	179	2	12	
4	Arcadia	7741	63	0	5	

```
# Calculate Crime per K and Load only City and Crime rate to a dataframe, will be used for f
df_flcrime = dfflcrime.filter(['City','crimeperk'], axis=1)
df_flcrime.columns = ['city','crimerate']
df_flcrime.head()
```

]:

	city	crimerate
0	Alachua	3.0
1	Altamonte Springs	4.0
2	Apalachicola	0.0
3	Apopka	4.0
4	Arcadia	8.0

- Geolocation data for all cities was retrieved using using Geopy geocoder. The Geopy geocoder documentation can be found in the following link: https://geopy.readthedocs.io/en/stable/

```
from geopy.geocoders import Nominatim
from geopy.extra.rate_limiter import RateLimiter

locator = Nominatim(user_agent="myGeocoder")
geocode = RateLimiter(locator.geocode, min_delay_seconds=1)
df_sfcities['location'] = df_sfcities['address'].apply(geocode)
df_sfcities['point'] = df_sfcities['location'].apply(lambda loc: tuple
df_sfcities[['latitude', 'longitude', 'altitude']] = pd.DataFrame(df_s
df_sfcities.head()
```

]:

	county	city	citytype	address	
0	Broward	Coconut Creek	city	Coconut Creek, Florida	(Coconut Creek, Broward
1	Broward	Cooper City	city	Cooper City, Florida	(Cooper City, Broward Cc
2	Broward	Coral Springs	city	Coral Springs, Florida	(Coral Springs, Broward
3	Broward	Dania Beach	city	Dania Beach, Florida	(Dania Beach, Broward C

- List of best beach cities was read from https://wallethub.com/edu/best-beach-towns-to-live-in/36567/

  Data is in format that is suitable for analysis, its loaded to Pandas dataframe directly. Since we are interested only in Florida cities,

```
urlbeaches = 'https://wallethub.com/edu/best-beach-towns-to-live-in/36567'
header = {
  "User-Agent": "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like G
  "X-Requested-With": "XMLHttpRequest"
}
r = requests.get(urlbeaches, headers=header)
lsbeached = pd.read_html(r.text)
dfbeaches = lsbeached[0].dropna(axis=0, thresh=4)
dfbeaches[['city','state']] = dfbeaches['City'].str.split(', ',expand=True)
dfbeaches.head()
```

3]:

	' Rank	City	Total Score	'Affordability' Rank	'Weather' Rank
0	1	Naples, FL	62.50	30	100
1	2	Lahaina, HI	61.25	68	31
2	3	Boca Raton, FL	60.96	17	5
3	4	Newport Beach, CA	60.01	69	56
4	5	Santa Monica, CA	59.87	122	26

```
df_flbeaches=dfbeaches[dfbeaches['state']=='FL'].filter(['city','state']
df_flbeaches.head()
```

]:

	city	state
0	Naples	FL
2	Boca Raton	FL
5	Sarasota	FL
9	Vero Beach	FL
11	Destin	FL

- Top Neighborhoods and geolocation data for each neighborhood:
  Top neighborhood data was retrieved from Wikipedia and loaded to lists., and to a final neighborhood Dataframe, Neighborhood geo-location was retrieved using geopy.

```
boca=['Yamato','Century Village','Villages of Oriole','West Deerfield Beach','Delray Beach']
naples=['The Old Naples','Port Royal','Sun Terrace','Moorings','Royal Harbor']
marco=['Marco Island','Pelican Bay','Vanderbilt Beach']
coral=['Riviera','Gables By The Sea','Gables Estates','Coral Groves','Cocoplum','Baker Homestead','
dfneigh = pd.DataFrame(columns=['city','neighbourhood'])
for i in range(len(boca)):
    dfneigh = dfneigh.append({'city': 'Boca Raton','neighbourhood':boca[i]}, ignore_index=True)
for i in range(len(naples)):
    dfneigh = dfneigh.append({'city': 'Naples','neighbourhood':naples[i]}, ignore_index=True)
for i in range(len(marco)):
    dfneigh = dfneigh.append({'city': 'Marco Island','neighbourhood':marco[i]}, ignore_index=True)
for i in range(len(coral)):
    dfneigh = dfneigh.append({'city': 'Coral Gables','neighbourhood':coral[i]}, ignore_index=True)

dfneigh.head()
```

```
df_sfneigh = pd.merge(df_sfcities, dfneigh, on = 'city')
df_sfneigh['neighbourhood_address']=df_sfneigh['neighbourhood']+", Florida" #+df_s
df_sfneigh['location'] = df_sfneigh['neighbourhood_address'].apply(geocode)
df_sfneigh['point'] = df_sfneigh['location'].apply(lambda loc: tuple(loc.point) if
df_sfneigh.head()
```

9]:

	county	city	citytype	address	location
0	Collier	Marco Island	city	Marco Island, Florida	(Marco Island, Collier County, Florida, United...
1	Collier	Marco Island	city	Marco Island, Florida	(Pelican Bay, Collier County, Florida, United ...
2	Collier	Marco Island	city	Marco Island, Florida	(Vanderbilt Beach, Pelican Marsh, Collier Coun...
3	Collier	Naples	city	Naples, Florida	(Naples, Collier County, Florida, United State...
4	Collier	Naples	city	Naples, Florida	(Port Royal, Naples, Collier County, Florida, ...

- Venue Data using Foursquare

  The places by Foursquare API is a database of more than 105 million places worldwide and is going to be consulted for this project. To explore the cities, we will use the Venue Recommendation which returns a list of recommended venues near a certain location. In order to make the query in the Foursquare API we need the coordinates given in Latitude and Longitude for a given Neighborhood. So. the first thing me we need to do is to get coordinates for each Neighborhood in each of the cities.

Function to get the top 100 venues that are in a neighbourhood within a radius of 500 meters.

```
def getNearbyVenues(names, latitudes, longitudes, radius=500):
    venues_list=[]
    for name, lat, lng in zip(names, latitudes, longitudes):
        # create the API request URL
        url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={},{}&radius={}&limit={}'.format(
            CLIENT_ID,
            CLIENT_SECRET,
            VERSION,
            lat,
            lng,
            radius,
            LIMIT)

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

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

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

    return(nearby_venues)
```

4. Methodology

4.1. Exploratory Data Analysis

One the preliminary data – all the citied names, is retrieved, the question below and the solution depict the process of analyzing data and reaching the conclusions.

Question: Which are the safest South Florida Cities?
Answer: We merge the South Florida cities data, with cities crime rate data. The crime rate is classified into four categories – Green, Yellow, Orange, Red, Green being the cities with lowest crime rate of <= 2 crimes per K.
Filtering cities data based on crime rate, we get a list of cities with 'Green' status.
We can visualize these cities on Map using Folium.

:

```
sfcities_df = pd.merge(df_sfcities, df_flcrime, on = 'city')
sfcities_df.head()
```

.4]:

	county	city	citytype	address	
0	Broward	Coconut Creek	city	Coconut Creek, Florida	(Coconut Creek, Broward County, Flori
1	Broward	Cooper City	city	Cooper City, Florida	(Cooper City, Broward County, Florida,
2	Broward	Coral Springs	city	Coral Springs, Florida	(Coral Springs, Broward County, Flori
3	Broward	Deerfield Beach	city	Deerfield Beach, Florida	(Deerfield Beach, Broward County, Flo
4	Broward	Fort Lauderdale	city	Fort Lauderdale, Florida	(Fort Lauderdale, Broward County, Flo

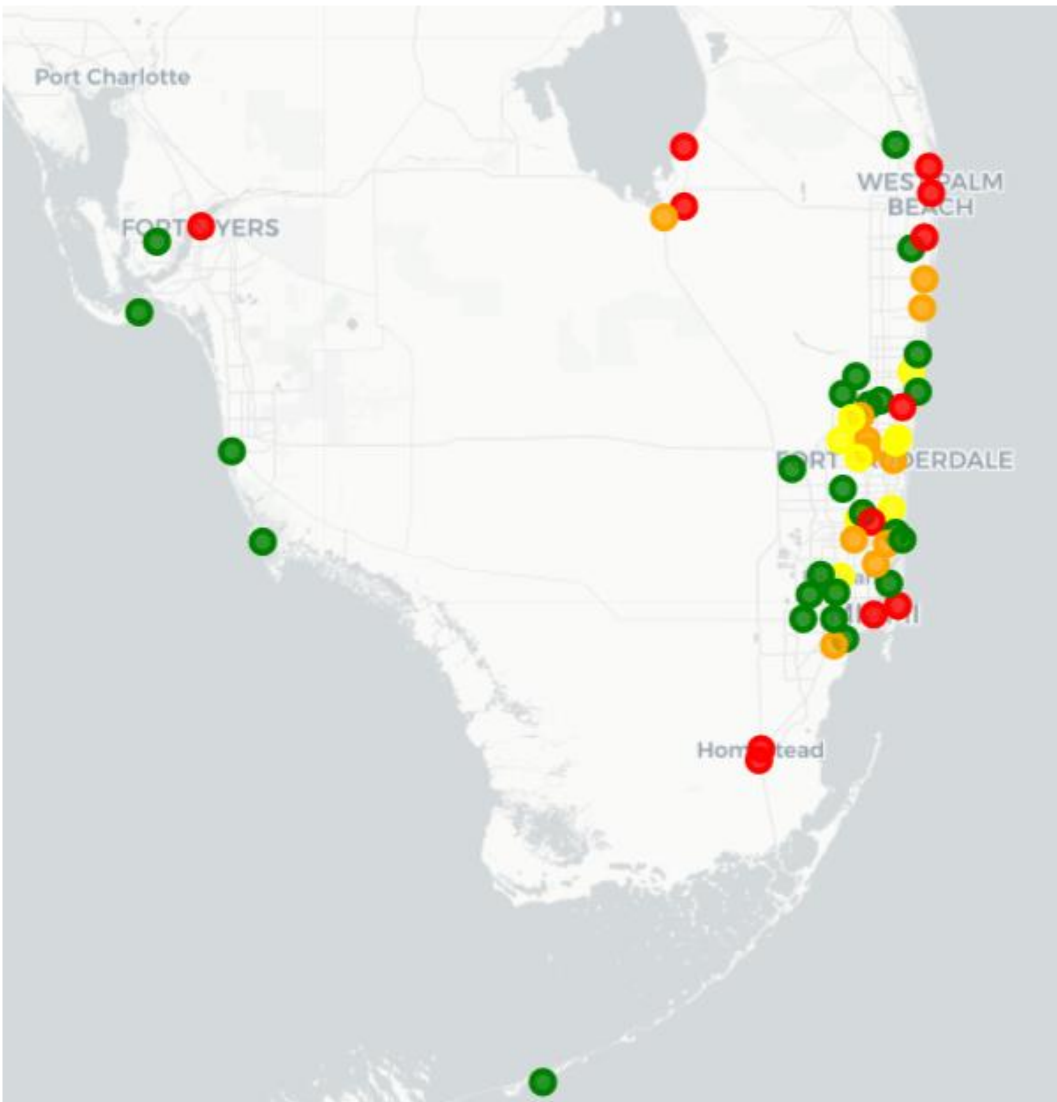
```
def label_crime (row):
    if row['crimerate'] <= 2 :
        return 'green'
    if row['crimerate'] <= 5 :
        return 'yellow'
    if row['crimerate'] <= 8 :
        return 'orange'
    return 'red'

sfcities_df['crimelevel'] = sfcities_df.apply (lambda row: label_crime(r
sfcities_df.head()
```

5]:

	county	city	citytype	address	crimerate	crimelevel
0	Broward	Coconut Creek	city	Coconut Creek, Florida	1.0	green
1	Broward	Cooper City	city	Cooper City, Florida	1.0	green
2	Broward	Coral Springs	city	Coral Springs, Florida	2.0	green
3	Broward	Deerfield Beach	city	Deerfield Beach, Florida	4.0	yellow
4	Broward	Fort Lauderdale	city	Fort Lauderdale, Florida	7.0	orange

]:
```
import folium
mapbroward = folium.Map(
    location=[26,-81],
    tiles='cartodbpositron',
    zoom_start=8,
)
sfcities_df.apply(lambda row:folium.CircleMarker(
    location=[row["latitude"],
              row["longitude"]],radius=5,color=row["crimelevel"],
    fill=True,fill_color=row["crimelevel"],
    fill_opacity=0.8).add_to(mapbroward), axis=1)
mapbroward
```



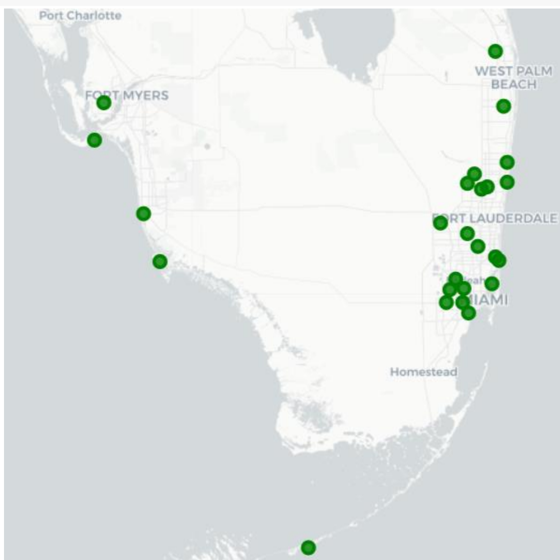
And Finally – the filtered Safe Cities:

```
df_safesf = sfcities_df[sfcities_df['crimelevel']=='green']
df_safesf.reset_index(drop=True, inplace=True)
df_safesf.head()
```

1]:

	county	city	citytype	address	
0	Broward	Coconut Creek	city	Coconut Creek, Florida	(Coconut Creek, Brc
1	Broward	Cooper City	city	Cooper City, Florida	(Cooper City, Browa
2	Broward	Coral Springs	city	Coral Springs, Florida	(Coral Springs, Brc
3	Broward	Lighthouse Point	city	Lighthouse Point, Florida	(Lighthouse Point, I
4	Broward	Margate	city	Margate, Florida	(Margate, Broward

```
mapsafe = folium.Map(
    location=[26,-81],
    tiles='cartodbpositron',
    zoom_start=8,
)
df_safesf.apply(lambda row:folium.CircleMarker(
    location=[row["latitude"],
              row["longitude"]],radius=5,color='green',
    fill=True,fill_color='green',
    fill_opacity=0.8).add_to(mapsafe), axis=1)
mapsafe
```



Question: From the list of safe cities of South Florida, which are the top Beach Cities?
Answer: We merge the South Florida Safe cities Dataframe, with List of Beach cities to get our top South Florida beach Cities. Again, we can visualize these cities on Map using Folium.

```
df_safebeach= pd.merge(df_safesf,df_flbeaches, on = 'city')
df_safebeach
```

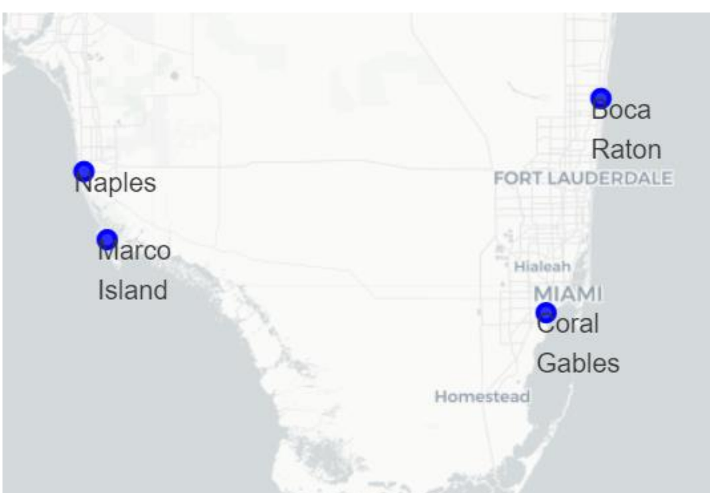
4]:

	county	city	citytype	address	
0	Collier	Marco Island	city	Marco Island, Florida	(Marco Island, Collier (
1	Collier	Naples	city	Naples, Florida	(Naples, Collier Count
2	Miami-Dade	Coral Gables	city	Coral Gables, Florida	(Coral Gables, Miami-Da
3	Palm Beach	Boca Raton	city	Boca Raton, Florida	(Boca Raton, Palm Beach

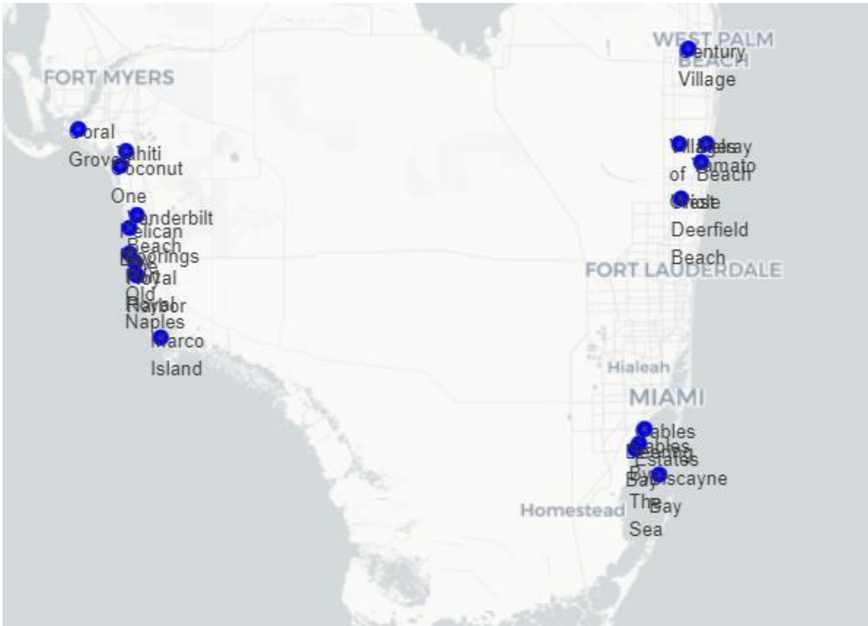
```
mapbeach = folium.Map(
    location=[26,-81],
    tiles='cartodbpositron',
    zoom_start=8,
)
df_safebeach.apply(lambda row:folium.CircleMarker(
    location=[row["latitude"],
              row["longitude"]],radius=5,color='blue',
    fill=True,fill_color='blue',
    fill_opacity=0.8).add_to(mapbeach), axis=1)

df_safebeach.apply(lambda row:folium.Marker(
    location=[row["latitude"],
              row["longitude"]],
        icon=DivIcon( html='<div style="font-size: 12pt">'+row["city"]
        ).add_to(mapbeach), axis=1)

mapbeach
```



Question: What are the neighborhoods in these top safe cities?
Answer: Top neighborhood data was retrieved from Wikipedia and loaded to lists and merged to create a neighborhood Dataframe, with geolocations for each neighborhood retrieved from geopy. These neighborhoods can be visualized on map.



Question: What are venue categories in these neighborhoods?

Answer: There are 224 unique venue categories in our final South Florida neighborhoods**.**

```
unique_vc = len(sofl_venues['Venue Category'].unique())
print(f'There are {unique_vc} unique venue categories in our final South Florida neighborhoods')
sofl_venues.groupby('Venue Category')['Venue Category'].count().sort_values(ascending=False)
```

```
There are 224 unique venue categories in our final South Florida neighborhoods
```

```
50]: Venue Category
     Hotel                 52
     Pizza Place           51
     Seafood Restaurant    40
     Italian Restaurant    38
     American Restaurant   33
```

Question: Are there any null values in the dataset?

Answer: No**.**

```
sofl_venues.isnull().values.any()
```

```
4]: False
```

4.2 Data Pre-Processing and cleaning

The preliminary dataset was cleaned according to the answers listed in the Exploratory Data Analysis section above. The derived neighborhood Dataframe was further cleaned to merge some of the related venue categories.

e.g. All the different Restaurant categories (American Restaurant, Italian Restaurant...) were assigned the category 'Restaurant'. All the pet related categories – dog park, veterinarian, pet store etc, were assigned category 'pet services'

```
sofl_venues.loc[sofl_venues['Venue Category'].str.contains('Restaurant|Steakhouse|Buffet', na = False) ,
sofl_venues.loc[sofl_venues['Venue Category'].str.contains('Donut|Sandwich|Coffee|Breakfast|Bistro|Cafe|C
sofl_venues.loc[sofl_venues['Venue Category'].str.contains('Pizza|Hot Dog|Diner|Deli|Burrito|Fried Chicke
sofl_venues.loc[sofl_venues['Venue Category'].str.contains('Dessert|Ice Cream|Icecream|Smoothie|Tea|Yogur
sofl_venues.loc[sofl_venues['Venue Category'].str.contains('Museum', na = False) , 'Venue Category'] = 'M
sofl_venues.loc[sofl_venues['Venue Category'].str.contains('Outdoors|Historic|Garden', na = False) , 'Ven
sofl_venues.loc[sofl_venues['Venue Category'].str.contains('Pet|Veterinarian|Dog', na = False) , 'Venue C
```

```
sofl_venues['Venue Category'].unique()
```

```
9]: array(['Park', 'Resort', 'Museum', 'Tiki Bar', 'Beach', 'Restaurant',
           'Cafe', 'Spa', 'Pool', 'Dining', 'Golf Course', 'Hotel',
           'Dessert Shop', 'Grocery Store', 'Multiplex', 'Supplement Shop',
           'Discount Store', 'Convenience Store', 'Sports Club',
           'Shopping Mall', 'Pharmacy', 'Pub', 'Harbor / Marina',
           'Salon / Barbershop', 'Gym', 'Smoke Shop', 'Art Gallery',
           'Clothing Store', 'River', "Women's Store", 'Fishing Spot',
```

Next, we create a list of categories that of interest to a person retiring, This list was used to create a final dataset of neighborhoods and venues

```
: cats_of_interest_retire = ['Park',  'Museum', 'Restaurant','Bookstore', 'Cafe', 'Dining', 'Golf Course',
                    'Dessert Shop','Outdoors','Theater','Beach','Pet Services']


  fl_retire_venues = sofl_venues[sofl_venues['Venue Category'].isin(cats_of_interest_retire)]
  print(fl_retire_venues['Venue Category'].unique())
  fl_retire_venues.head(5)
```

```
['Park' 'Museum' 'Beach' 'Restaurant' 'Cafe' 'Dining' 'Golf Course'
 'Dessert Shop' 'Outdoors' 'Pet Services' 'Theater' 'Bookstore']
```

56]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Marco Island, Florida	25.936336	-81.715683	Mackle Park	25.930477	-81.713267	Park
2	Marco Island, Florida	25.936336	-81.715683	Marco Island Museum	25.933500	-81.715930	Museum
4	Marco Island, Florida	25.936336	-81.715683	Marco Beach	25.925858	-81.729895	Beach
5	Marco Island, Florida	25.936336	-81.715683	Quinn's on the Beach	25.927387	-81.729544	Restaurant
6	Marco Island, Florida	25.936336	-81.715683	Doreen's Cup of Joe	25.943632	-81.732358	Cafe

```
print(fl_retire_venues.groupby('Venue Category')['Venue Category'].count().sort_values(ascending=False))
```

```
Venue Category
Restaurant      267
Cafe            104
Dining           74
Park             40
Beach            36
Golf Course      26
```

4.3 One-Hot-Encoding Venue Categories

In order to use Foursquare's category values to find similar neighborhoods based on desired venues, a one-hot-encoding representation of each entry was created using Pandas' 'get_dummies' function. The result was a dataframe of retirement-related venues where entry venue category is represented by a value of 1 in the column of matching venue category, as shown below:

```
fl_retire_venues_onehot = pd.get_dummies(fl_retire_venues[['Venue Category']], prefix="", prefix_sep="")
fl_retire_venues_onehot['Neighborhood'] = fl_retire_venues['Neighborhood']

fixed_columns = [fl_retire_venues_onehot.columns[-1]] + list(fl_retire_venues_onehot.columns[:-1])
fl_retire_venues_onehot = fl_retire_venues_onehot[fixed_columns]
fl_retire_venues_onehot.head(10)
```

70]:

	Neighborhood	Beach	Bookstore	Cafe	Dessert Shop	Dining	Golf Course	Museum	Outdoors	Park	Pet Services	Restaurant	Theater
0	Marco Island, Florida	0	0	0	0	0	0	0	0	1	0	0	0
2	Marco Island, Florida	0	0	0	0	0	0	1	0	0	0	0	0
4	Marco Island, Florida	1	0	0	0	0	0	0	0	0	0	0	0
5	Marco Island, Florida	0	0	0	0	0	0	0	0	0	0	1	0
6	Marco Island, Florida	0	0	1	0	0	0	0	0	0	0	0	0
7	Marco Island, Florida	0	0	0	0	0	0	0	0	0	0	1	0
8	Marco Island, Florida	1	0	0	0	0	0	0	0	0	0	0	0
9	Marco Island, Florida	0	0	1	0	0	0	0	0	0	0	0	0
11	Marco Island, Florida	0	0	1	0	0	0	0	0	0	0	0	0
13	Marco Island, Florida	0	0	0	0	0	0	0	0	0	0	1	0

4.4 Aggregating Venues for each Category

From the One-Hot-Encoding Venue Categories Dataframe, we determine the total amount of venues of each category in each neighborhood

```
fl_interestv_counts = fl_retire_venues_onehot.groupby('Neighborhood').sum()
fl_interestv_counts.head(10)
```

]:

Neighborhood	Beach	Bookstore	Cafe	Dessert Shop	Dining	Golf Course	Museum	Outdoors	Park	Pet Services	Restaurant	Theater
Baker Homestead, Florida	1	0	2	0	2	0	0	0	1	0	11	0
Century Village, Florida	0	0	5	1	8	0	0	0	3	1	17	0
Coconut One, Florida	0	0	1	0	0	2	0	0	0	0	2	0
Cocoplum, Florida	0	0	0	0	0	0	0	0	1	0	0	0
Coral Groves, Florida	0	1	2	0	0	2	0	0	1	0	9	0
Deering Bay, Florida	0	0	0	0	0	1	0	0	4	0	0	0

4.5 Feature Generation

The encoded dataset of retirement-desired venues in South Florida neighborhoods was then used to quantify a profile for each neighborhood. For each venue category, the percent distribution of venues across each neighborhood was calculated. This information would then be used to fit a K-Means clustering algorithm to the data in an effort to determine neighborhoods of similar profile.

First, the total number of venues for each category was determined:

```
flvenue_totals = {}
for category in cats_of_interest_retire:
    flvenue_totals[category] = fl_interestv_counts[category].sum()
flvenue_totals
```

```
]: {'Park': 40,
 'Museum': 7,
 'Restaurant': 267,
 'Bookstore': 2,
 'Cafe': 104,
 'Dining': 74,
 'Golf Course': 26,
 'Dessert Shop': 24,
 'Outdoors': 7,
 'Theater': 3,
 'Beach': 36,
 'Pet Services': 11}
```

Next. For each venue category, determine the percentage of entities in each neighborhood

```
: flvenue_mean = pd.DataFrame()
for category, total in flvenue_totals.items():
    flvenue_mean[category] = fl_interestv_counts[category].apply(lambda x: x / total)
flvenue_mean = flvenue_mean.reindex(sorted(flvenue_mean.columns), axis=1).reset_index()
flvenue_mean.head(5)
```

85]:

	Neighborhood	Beach	Bookstore	Cafe	Dessert Shop	Dining	Golf Course	Museum	Outdoors	Park
0	Baker Homestead, Florida	0.027778	0.0	0.019231	0.000000	0.027027	0.000000	0.0	0.0	0.025
1	Century Village, Florida	0.000000	0.0	0.048077	0.041667	0.108108	0.000000	0.0	0.0	0.075
2	Coconut One, Florida	0.000000	0.0	0.009615	0.000000	0.000000	0.076923	0.0	0.0	0.000
3	Cocoplum, Florida	0.000000	0.0	0.000000	0.000000	0.000000	0.000000	0.0	0.0	0.025
4	Coral Groves, Florida	0.000000	0.5	0.019231	0.000000	0.000000	0.076923	0.0	0.0	0.025

Finally, a Dataframe showing the top five venue categories for each neighborhood was created:

```
def return_top_venue_categories(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]
```

```
num_top_venues = 5
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('{}{} Top Venue Category'.format(ind+1, indicators[ind]))
    except:
        columns.append('{}th Top Venue Category'.format(ind+1))

neighborhoods_top_venue_categories = pd.DataFrame(columns=columns)
neighborhoods_top_venue_categories['Neighborhood'] = flvenue_mean['Neighborhood']
for ind in np.arange(flvenue_mean.shape[0]):
    neighborhoods_top_venue_categories.iloc[ind, 1:] = return_top_venue_categories(fl
neighborhoods_top_venue_categories.head(10)
```

	Neighborhood	1st Top Venue Category	2nd Top Venue Category	3rd Top Venue Cat
0	Baker Homestead, Florida	Restaurant	Beach	
1	Century Village, Florida	Dining	Pet Services	
2	Coconut One, Florida	Golf Course	Cafe	Rest
3	Cocoplum, Florida	Park	Theater	Rest
4	Coral Groves, Florida	Bookstore	Golf Course	Rest
5	Deering Bay, Florida	Park	Golf Course	T
6	Delray Beach, Florida	Theater	Outdoors	Rest
7	Gables By The Sea, Florida	Park	Theater	Rest
8	Gables Estates, Florida	Outdoors	Park	
9	Golden Triangle, Florida	Museum	Beach	

4.6 Cluster Modeling

Scikit-learn's K-Means clustering was used to determine similar neighborhoods based on venue percentage. The image below shows the data being scaled and the K-Means model being created:

```
from sklearn.cluster import KMeans
kclusters = 10
venue_grouped_clustering = flvenue_mean.drop('Neighborhood', 1)
kmeans = KMeans(n_clusters=kclusters, random_state=0).fit(venue_grouped_clu
neighborhoods_top_venue_categories.insert(1,'Cluster Labels', kmeans.labels
neighborhoods_top_venue_categories.head(10)
```

]:

	Neighborhood	Cluster Labels	1st Top Venue Category	2nd Top Venue Catego
0	Baker Homestead, Florida	2	Restaurant	Bea
1	Century Village, Florida	1	Dining	Pet Servic
2	Coconut One, Florida	2	Golf Course	Ca
3	Cocoplum, Florida	2	Park	Thea
4	Coral Groves, Florida	0	Bookstore	Golf Cour
5	Deering Bay, Florida	2	Park	Golf Cour
6	Delray Beach, Florida	6	Theater	Outdoc
7	Gables By The Sea, Florida	2	Park	Thea
8	Gables Estates, Florida	8	Outdoors	Pa
9	Golden Triangle, Florida	4	Museum	Bea

5. Results

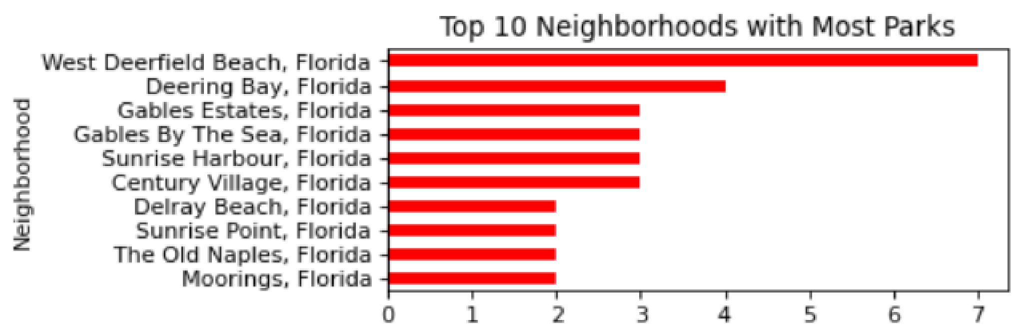
5.1. Top neighborhoods for each venue

Using the Dataframe of venue counts shown above, horizontal bar plots were created for select venue categories to help visualize the top neighborhoods with the most of each particular venue. Matplotlib library is used for plotting.

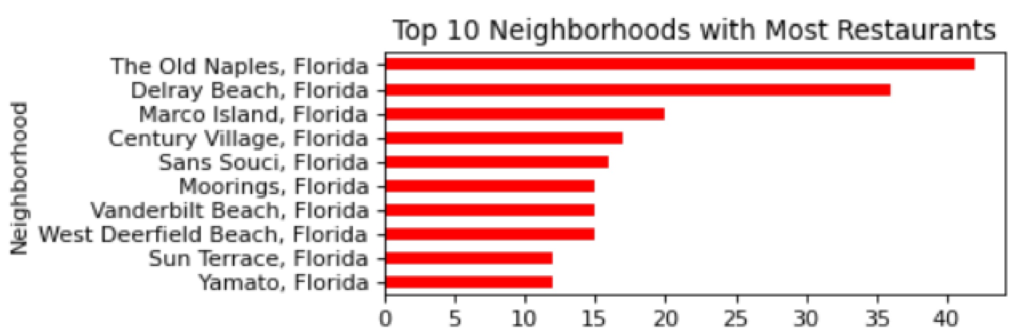
```
%%javascript
IPython.OutputArea.auto_scroll_threshold = 9999;
//Now the output window will be large and we can see all the outputs without scrolling
```

```
import matplotlib.pyplot as plt
n = 10 #Show top 10 neighbourhoods
clrs = ['red']
for category in cats_of_interest_retire:
    plt.figure(num=None, figsize=(12, 7), dpi=80, facecolor='w', edgecolor='k')
    plt.title(f'Top {n} Neighborhoods with Most {category}s')
    top_category_neighs = fl_interestv_counts[category].sort_values(ascending=False)[0:n]
    top_category_neighs = top_category_neighs.sort_values(ascending=True)
    top_category_neighs.plot.barh(y=category, rot=0, color=clrs,figsize=(5,2))
```

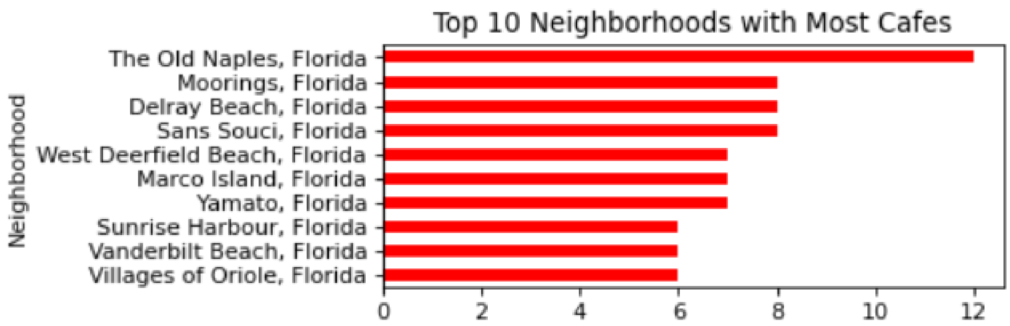
Top 10 Neighborhoods with Most Parks



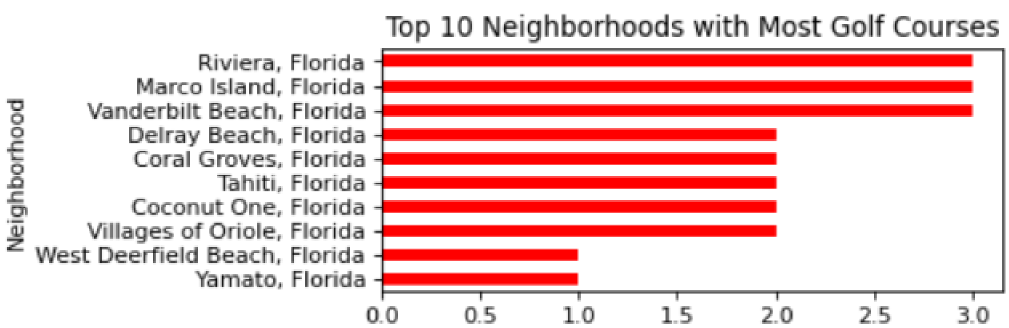
Top 10 Neighborhoods with Most Restaurants



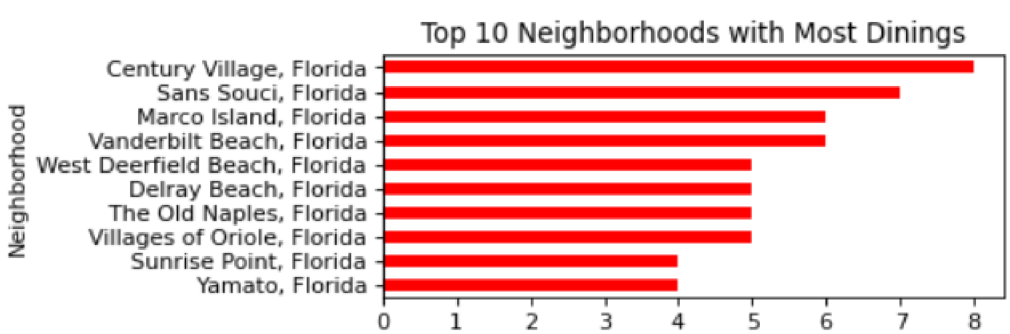
Top 10 Neighborhoods with Most Cafes



Top 10 Neighborhoods with Most Golf Courses



Top 10 Neighborhoods with Most Dining



5.2. Neighborhoods with Similar profiles

A dataframe merged from neighborhood location data, top venue category by neighborhood, and cluster labels was created– It allows us to see neighborhoods with similar profiles.

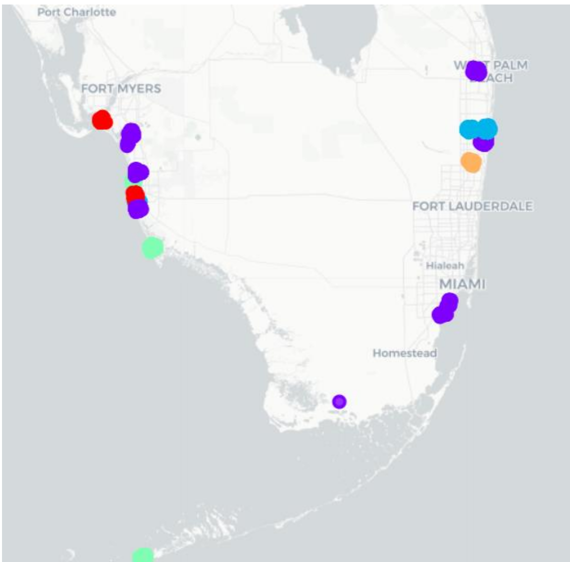
```
fl_neighborhood_retire_profile = sofl_venues.join(neighborhoods_top
fl_neighborhood_retire_profile.head()
```

]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue
0	Marco Island, Florida	25.936336	-81.715683	Mackle Park
1	Marco Island, Florida	25.936336	-81.715683	JW Marriott Marco Island Beach Resort
2	Marco Island, Florida	25.936336	-81.715683	Marco Island Museum
3	Marco Island, Florida	25.936336	-81.715683	Tiki Bar
4	Marco Island, Florida	25.936336	-81.715683	Marco Beach

5.3. Visualization of Clusters - Neighborhoods with Similar profiles

We again use folium to visualize neighborhoods of profile by coloring each neighborhood point based on cluster label:



5.4. Cluster Evaluation

We iterate through and prints the results of each cluster:

Each cluster shows a list of neighborhoods with their respective top venue categories. We can compare the resulting clusters to the bar plots in the Data Visualization section and get a sense that the clusters are properly grouping neighborhoods based on venue counts.

It is interesting to see that some clusters are very small, e.g. cluster 4 has only one neighborhood, a beach area with lots of restaurants and café, but no parks, libraries etv

```
fl_neighborhood_retire_profile_cl = fl_neighborhood_retire_profile.loc[fl_neighborhood_
'] == 4].filter(['Neighborhood','Venue Category'], axis=1)
dftemp = fl_neighborhood_retire_profile_cl.groupby('Neighborhood')['Venue Category'].ap
dftemp
```

	Neighborhood	Venue Category
0	West Deerfield Beach, Florida	Park,Grocery Store,Restaurant,Cafe,Restaurant,...

Bigger cluster 3 appear to be grouping neighborhoods with assortments of venues such as Parks, Scenic areas, Museums etc.

```
fl_neighborhood_retire_profile_cl = fl_neighborhood_retire_p
'] == 3].filter(['Neighborhood','Venue Category'], axis=1)
dftemp = fl_neighborhood_retire_profile_cl.groupby('Neighbor
dftemp
```

	Neighborhood	Venue Category
0	Golden Triangle, Florida	Scenic Lookout,Resort,Dining,Resort,Hotel,Muse...
1	Marco Island, Florida	Park,Resort,Museum,Tiki Bar,Beach,Restaurant,C...
2	Pelican Bay, Florida	Outdoors,Beach,Restaurant,Restaurant,Concert H...
3	Sun Terrace, Florida	Creperie,Burger Joint,Dining,Resort,Cafe,Cloth...
4	Sunrise Point, Florida	Dining,Restaurant,Dining,Park,Park,Boutique,Ba...

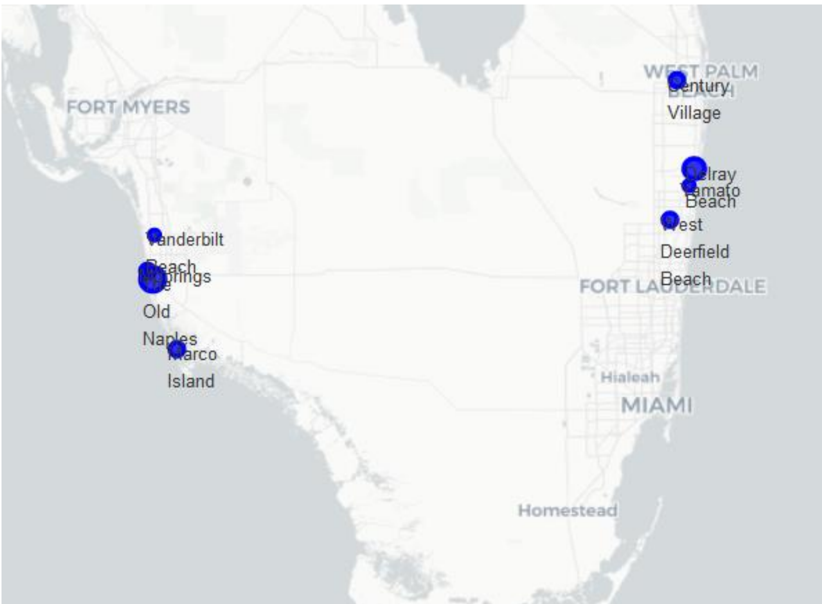
5.5. Top neighborhoods

And here are our top neighborhoods

```
fl_conclude_ratings=fl_interestv_counts.copy()
fl_conclude_ratings['Venue Total']= fl_conclude_ratings.i
fl_conclude_ratings=fl_conclude_ratings.filter(['Neighborh
ascending=False)
fl_conclude_ratings
dffinal = pd.DataFrame(columns=['venuetotal','neighbourhoo
for index, row in  fl_conclude_ratings.iterrows():
    dffinal = dffinal.append({'venuetotal': row['Venue Tot
dffinal
```

	venuetotal	neighbourhood_address
0	71	The Old Naples, Florida
1	61	Delray Beach, Florida
2	44	Marco Island, Florida
3	42	Moorings, Florida
4	40	West Deerfield Beach, Florida
5	35	Century Village, Florida
6	33	Vanderbilt Beach, Florida
7	33	Sans Souci, Florida
8	30	Sun Terrace, Florida
9	27	Yamato, Florida

On Map:



6. Discussion

Based on the results of our analysis, Neighborhood 'The Old Naples' in Naples, 'Delray Beach' in Boca Raton and Marco Island are the top neighborhood for retiring. Based on bestplaces.net, neighborhoods in Naples, Boca Raton and Marco Island do top the most desirable places in South Florida. All these neighborhoods are residential enclaves with beaches and a variety of activities, serves more year-round and winter residents than vacationing tourists. There are many golf courses and golf-course developments with upscale housing and shopping areas. The area is generally attractive and has a relaxed, modern feel, and the commercial activity is mainly related to supporting the area’s residents.

7. Limitations

All of the above analysis is based on Four-Square Places API. The Places API offers real-time access to Foursquare’s global database of rich venue data, but since we used a free Sandbox Tier Account of Foursquare that has limitations on number of API calls and results returned. To get better results, future research work and more comprehensive analysis could consider using a paid account to bypass these limitations as well as incorporating data from other external databases.

8. Conclusion

Using this project, we tried to solve problem, provides a solution - List of top stakeholder's neighborhoods in South Florida and groups of 'similar' neighborhoods - based on Machine learning and clustering algorithms.

This project can be expanded on in several different ways. The desired venue categories can be based on user preferences (Some Retired people may like Bars and nightlife instead of quite neighborhoods). Foursquare’s API could be further interrogated to retrieve and consider more venues. The clustering model could become the basis for a recommendation system aimed to provide neighborhoods of similar profile to users.

Machine learning and clustering algorithms can be applied to multi-dimensional datasets to find similarities and patterns in the data. Clusters of neighborhoods of similar profile can be generated using high-quality venue location data. There is a preface on high-quality because analysis models are only as good as the input into them (garbage in, garbage out). Luckily, foursquare offers a robust ‘Places API’ service that, although (as we have seen) not perfect (nothing is), can be leverages in similar studies and model-making.

Project GitHub: https://github.com/v2rinku/capstonetst

9. References

[0] — List of Florida cities by county

[1] — Crime date for Florida

[2] — ‘Places API’ Documentation — Foursquare

[3] — geopy for finding geolocations for cities/neighborhoods