Segmenting and Clustering Neighborhoods in Vancouver City

DIEGO NUNES BOTELHO

Data Acquisition and Preparation

In this section, the process of acquiring, cleaning, and preparing the dataset used in this project for the next stages will be specified. To be able to do this project, two types of data are needed:

- **Neighborhood Data:** datasets with the list names of the neighborhoods of Vancouver and their latitude and longitude coordinates. The neighborhoods names were obtained in the website of Vancouver (https://vancouver.ca/news-calendar/areas-of-the-city.aspx) and the latitude and longitude data were obtained using a recursive function that would return the geocode of the address passed into it.
- **Venues Data:** data that describes the top 100 venues (restaurants, cafes, parks, museums, etc) in each neighborhood of Vancouver. The data should list the venues of each neighborhood with their categories. For example:

Table 1 – Example of the Venues Data

Venue	Category
Spartacus Books	Bookstore
Trout Lake Fitness Centre	Gym / Fitness Center

This data will be retrieved from Foursquare which is one of the world largest sources of location and venues data. Foursquare API will be utilized to get and download the data.

1. Neighborhood Data – Vancouver

A dataset was created from the combination of two sources. The following figures show the process for generating the data table.

[2]:
```
# Vancouver neighborhoods
van_neighborhoods = ['Arbutus Ridge', 'Cedar Cottage', 'Champlain Heights', 'Chinatown', 'Coal Harbour', 'Collingwood',
                     'Commercial Drive', 'Creekside', 'Downtown', 'Downtown Eastside', 'Dunbar-Southlands', 'Fairview',
                     'False Creek North', 'False Creek South', 'Gastown', 'Grandview-Woodland', 'Granville Island',
                     'Hastings-Sunrise', 'Hastings Crossing', 'Hastings East', 'Kensington-Cedar Cottage', 'Kerrisdale',
                     'Killarney', 'Kitsilano', 'Knight', 'Langara', 'Little Mountain', 'Main', 'Marpole', 'Mole Hill',
                     'Mount Pleasant', 'Musqueam', 'Oakridge', 'Quilchena', 'Renfrew-Collingwood', 'Riley Park',
                     'Shaughnessy', 'South Cambie', 'South Granville', 'South Hill', 'South Vancouver', 'Southlands',
                     'Southwest Marine', 'Sunrise', 'Sunset', 'Victoria-Fraserview', 'West Broadway', 'West End', 'West Point Grey', 'Yaletown']
```

[3]:
```
df_van = pd.DataFrame(van_neighborhoods)
df_van.columns = ['Neighborhood']
df_van.head()
```

[3]:

	Neighborhood
0	Arbutus Ridge
1	Cedar Cottage
2	Champlain Heights
3	Chinatown
4	Coal Harbour

Figure 2 – Neighborhoods of Vancouver

```
[4]: #create a function to handle TimeOuts from Geocoder
from geopy.exc import GeocoderTimedOut
locator = Nominatim(user_agent = "bostonagent")

def do_geocode(address):
    try:
        return locator.geocode(address)
    except GeocoderTimedOut:
        return do_geocode(address)
```

```
[5]: neighborhoods = df_van.values.tolist()

latitude = []
longitude = []
for neighborhood in neighborhoods:
    print('-', end='')
    coord = do_geocode('{}, Vancouver'.format(neighborhood))

    #check to make sure all latitude and longitude values are present in the Nominatim API
    #handles the case where Nominatim returns a 'None' object because the neighborhood does not exist in their API

    if (coord == None):
        latitude.append('0')
        longitude.append('0')
    else:
        latitude.append(coord.latitude)
        longitude.append(coord.longitude)

#add coordinates columns to dataframe
df_van['Latitude'] = latitude
df_van['Longitude'] = longitude

df_van.head()
```

Figure 3 – Function to obtain Latitude and Longitude

The Figure 4 shows the resulting dataframe which contains data on 50 neighborhoods.

[5]:

	Neighborhood	Latitude	Longitude
0	Arbutus Ridge	49.240968	-123.167001
1	Cedar Cottage	49.251622	-123.064548
2	Champlain Heights	49.215266	-123.030915
3	Chinatown	49.279981	-123.104089
4	Coal Harbour	49.290375	-123.129281

Figure 4 –Vancouver neighborhood-data dataframe

Having data of the coordinates of Vancouver, it is possible to draw a map using Folium Python package of Vancouver and its neighborhoods. Figure 5 shows this map; each blue circles represents the location of one neighborhood.

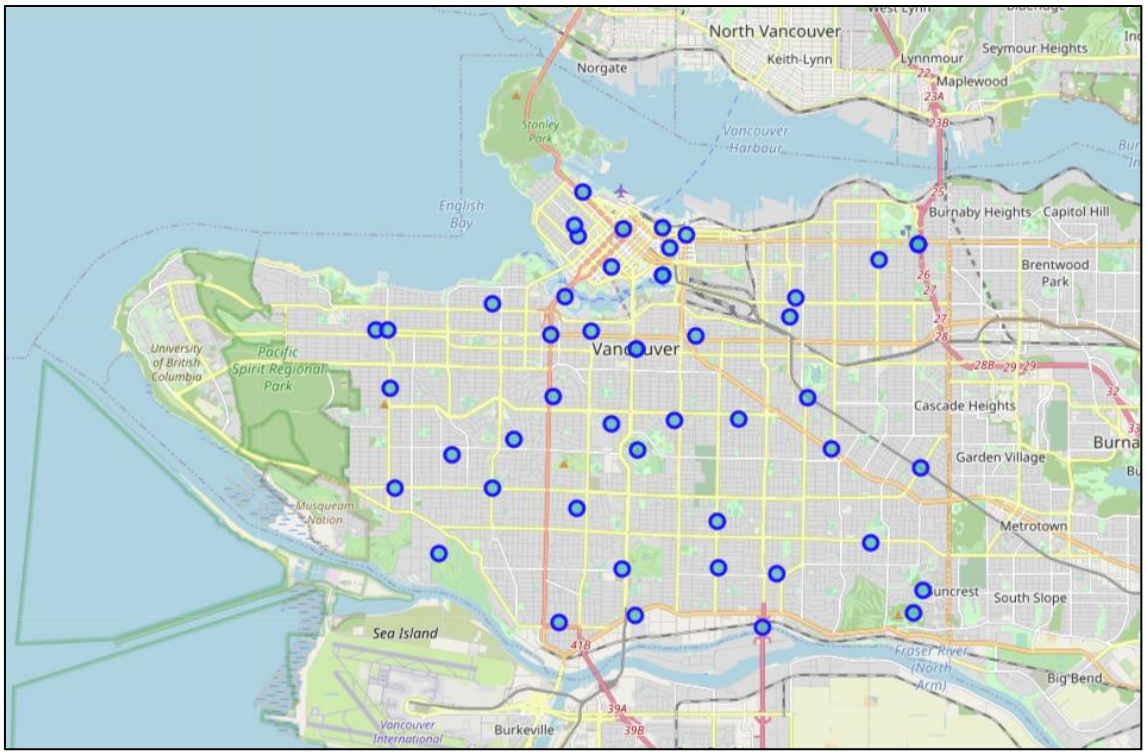

Figure 5 – A map of Vancouver and its neighborhoods

1. Venues Data – Vancouver

For the city, data that describes the venues of its neighborhoods and the categories of these venues is needed. Venues data will be retrieved from Foursquare which is a popular source of location and venue data. Foursquare API service will be utilized to access and download venues data. To retrieve data from Foursquare using their API, a URL should be prepared and used to request data related a specific location. An example URL is the following:

```
https://api.foursquare.com/v2/venues/search?
&client_id=1234&client_secret=1234&v=20180605&
ll=40.89470517661,-73.84720052054902&radius=500&limit=100
```

Figure 7 – URL example

Where search indicates the API endpoint used, client_id and client_secret are credentials used to access the API service and are obtained when registering a Foursquare developer account, v indicates the API version to use, ll indicates the latitude and longitude of the desired location, radius is the maximum distance in meters between the specified location and the retrieved venues, and limit is used to limit the number of returned results if necessary.

Figure 8 shows the code used to create a function that takes as input the names, latitudes, and longitudes of the neighborhoods, and returns a dataframe with information about each neighborhood and its venues. It creates an API URL for each neighborhood and retrieves data about the venues of that neighborhoods from Foursquare.

```
[9]: # Explore Neighborhoods in Vancouver

LIMIT = 100
def getNearbyVenues(names, latitudes, longitudes, radius=500):

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

        # 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)
```

Figure 8 – Code used to build a venues dataframe for a city neighborhoods.

For the study restricting outdoors and recreation activities and restaurants, were necessary do add category ID (Outdoors & Recreation ID- 4d4b7105d754a06377d81259; and restaurants ID - 4d4b7105d754a06374d81259).