Toronto Gamestop

Liujian Wang

Part 1. Introduction/Business Problem

Gamestop is a dominating video game retailer in north America. Gamestop stores are chain stores that sell video games as well as gaming related product, usually scattered among major cities. If someone is going to step into this business and open a Gamestop store, the best strategy is to find the neighborhood in Toronto that has the least competitors, i.e., other Gamestop stores, and the neighborhood that has highest income and population.

Part 2. Data acquisition and processing

The Foursquare location data will be used to get the number of Gamestop in each neighborhood, and from: https://en.wikipedia.org/wiki/Demographics_of_Toronto_neighbourhoods.
we can get the population and average income to help making decisions.

https://developer.foursquare.com/docs/resources/categories Using Foursquare API, we can find the potential competitors:

- Electronics store: 4bf58dd8d48988d10b951735. this is the category of electronic stores like BestBuy, usually they would sell games besides electronic products.
- Video game store: 4bf58dd8d48988d122951735. this is the category of videogame stores like BameStops, these are the major competitors of the new store.

1. BeautifulSoup4
Beautifulsoup4 API is used to create a data-frame containing name of the neighborhoods of Toronto. From it I can get the population and average income to help making decisions.

	Name	Population	Average Income
0	Agincourt	44,577	25,750
1	Alderwood	11,656	35,239
2	Alexandra Park	4,355	19,687
3	Allenby	2,513	245,592
4	Amesbury	17,318	27,546
5	Armour Heights	4,384	116,651
6	Banbury	6,641	92,319

Fig.1 Raw data returned by BeautifulSoup API

2. Google geocode

Generally, the Foursquare API is a good tool of translating from a given latitude and longitude to venues that are nearby. So it is important to first get location info of each neighborhood. Here I am using Google geocode API key to find the coordinates.

	Latitude	Longitude
0	43.785353	-79.278549
1	43.601717	-79.545232
2	43.650758	-79.404298
3	43.711351	-79.553424
4	43.706162	-79.483492

Fig.2 Coordination returned by geocode

A function (see appendix 1) uses Google geocode API to return a dataframe, but some of the neighbourhoods have no coordination, so I need to remove these neighborhoods from the dataframe with another function (appendix 2).

3. Foursquare
In order to find out how many competitors are there in each neighborhood, I used Foursquare API to request store information. A function (appendix 3) was used to recursively request venue infomation from Foursquare.

After I have got the data and put them into dataframes, it is important to connect the datafram with other data from geocode. To do that, I reset the index, now instead of numbers, the new datafram used neighbourhood as index. Then I changed the column names so they can be compatible between two dataframes. The datatype of each column also needed be be casted to int64 so that python can process it.

	Population	Average Income	Latitude	Longitude	Competitors
Neighbourhood					
Agincourt	44,577	25,750	43.785353	-79.278549	7
Alderwood	11,656	35,239	43.601717	-79.545232	2
Alexandra Park	4,355	19,687	43.650758	-79.404298	45
Allenby	2,513	245,592	43.711351	-79.553424	3
Amesbury	17,318	27,546	43.706162	-79.483492	3

Fig.3 Coordination and number of competitors

At last, there are 165 venues left, which means there at least 165 competitors allocated in this city. We then conducted statistics of numbers of competitors in each neighborhood.

Part 3 Data visualization

In order to get an overall picture of competitors in the Toronto City, folium was used to pin all the venues on the map of Toronto just to get some sense of how the stores are distributed among neighborhoods. Here I am using folium to visualize the data.

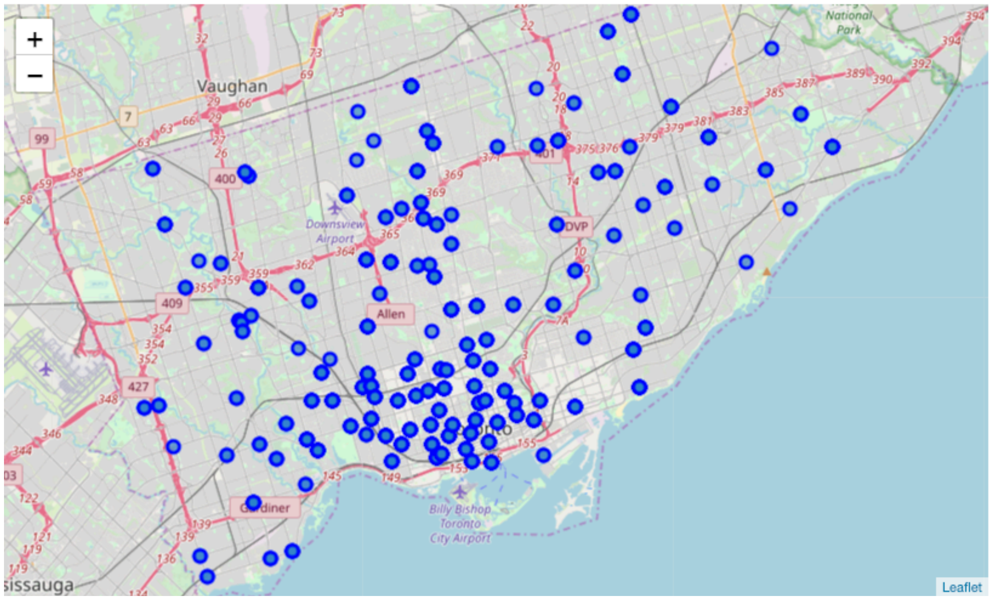

Fig.4 Distribution of competitors among the city

As we can see, in the downtown area stores are more densely allocated, this is reasonable since there is usually more people and better business environment in downtown area. But it also means competition is more intense here. In addition, the cost of running a store in such places is usually higher than others, since the rents are very expensive. So opening a videogame store here should be under full consideration.

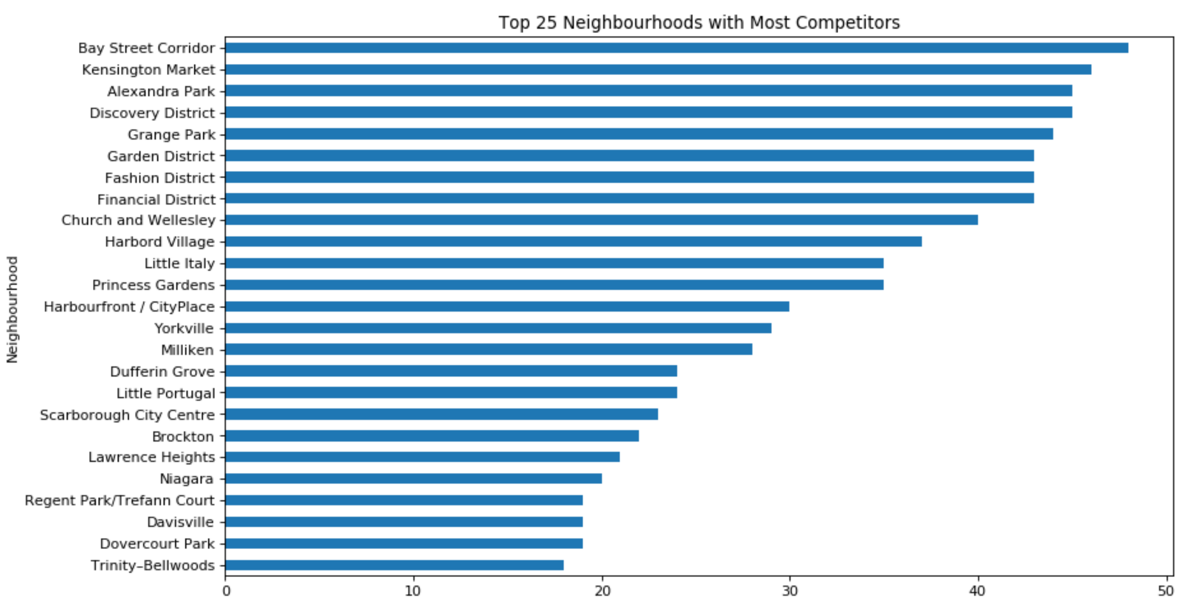

Fig.5 Top 25 neighborhoods with most competitors

The picture is supported by statistic data, Fig.5 lists the top 25 neighborhoods that have most competitors. As one can see from the top 25 neighborhoods with most video game stores, there are most stores in Bay Street corridor, Kensington Market, Alexandra Park and Discovery District, but the average income of these neighborhoods are not outstanding, therefore it may not be a good idea to open another store in these neighborhoods.

One may also look at the population and average income data in each neighborhood, since both can be important indicators of the consumption power of local residents. From Fig.6 the bar chart of most population, we can see that Old East York is most densely populated. Here if we assume that the composition of video game players is evenly distributed among all Toronto city, then higher population could be an indicator of more potential customers, although we still need to check their consumption strength with other indicators such as income level, average age and land rents.

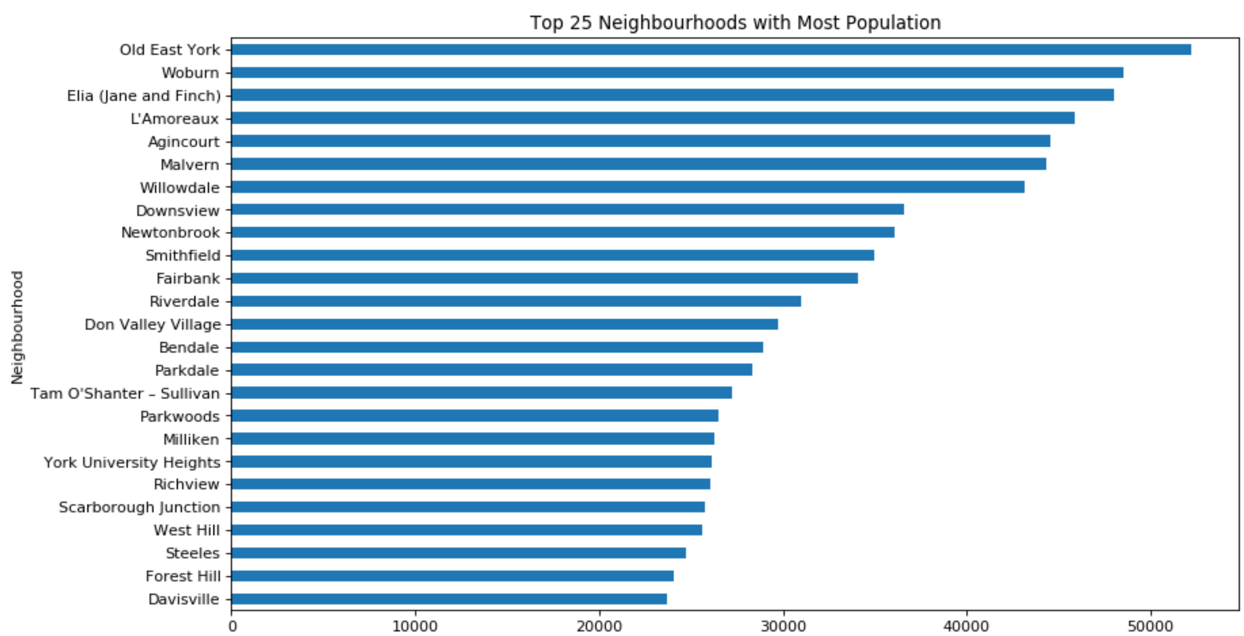

Fig.6 Top 25 neighborhoods with most population

From the bar chart of most average income, it seems that people in Bridle Path, Allenby, and Hoggs Hollow are wat more better off than the rest Toronto city. This typically indicates that people in these neighborhoods are more budget-free and they tend to spend more money for the leisure purpose since they have loose budgets.

It is also important to keep in mind that these data and information are only for reference, since they only cover one particular year and come from one source. For the purpose of study and exercise the use of data science tools we have also made several assumptions to minimize and simplify the problem that was supposed to be very complicated. For these reasons, one may never simply make decision depend on these results, as sometime unforeseeable deviation from reality could lead to fatal outcomes.

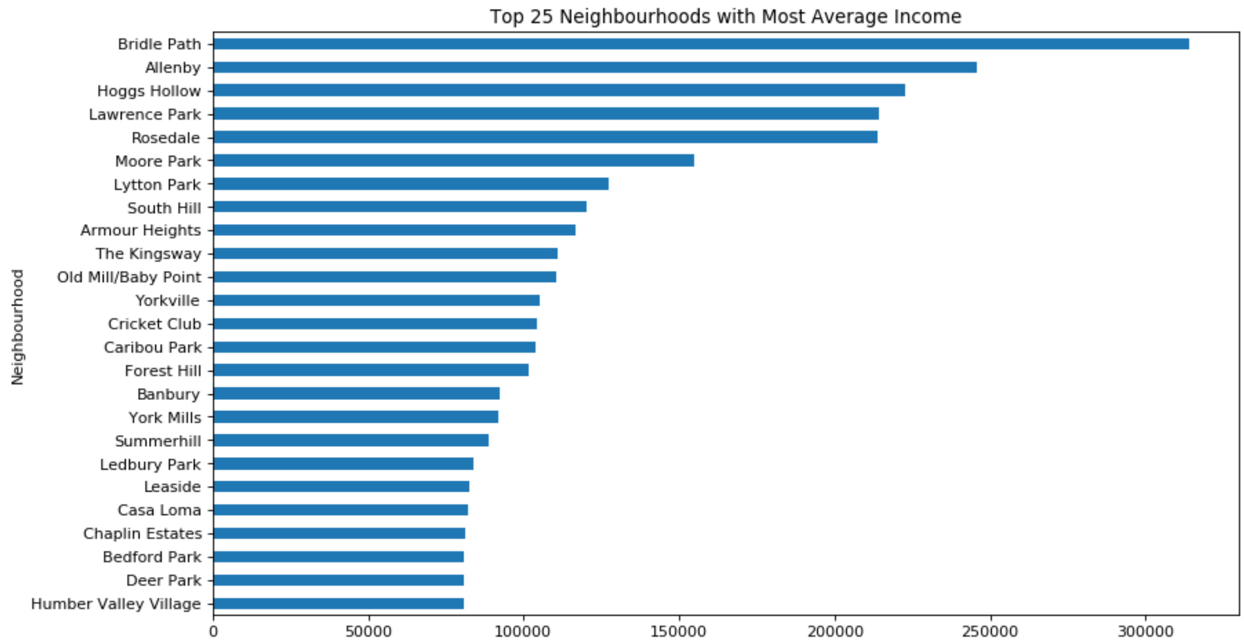

Fig.7 Top 25 neighborhoods with most average income

Part 4. Conclusion

Data science can be used as effective tools to collect and get useful results to help make decision when it comes to opening a video game store in the Toronto City. Clusters of neighborhoods of similar music profile, or any 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.

Making a decision is never easy, but it could be more accurate and less 'random' with the aid of reliable data. For investors who want to open a Gamestop store in Toronto City, they need to consider the competition with other brands and small stores, they also need to take the income level and population of local people. It is also important to keep in mind that these data and information are only for reference, since they only cover one particular year and come from one source. For the purpose of study and exercise the use of data science tools we have also made several assumptions to minimize and simplify the problem that was supposed to be very complicated. Thus one may never simply make decision depend on these results, as sometime unforeseeable deviation from reality could lead to fatal outcomes.

This project is by no means finished and could be expanded on in a number of different ways. Foursquare's API could be further interrogated to retrieve and consider more videogame-related venues in Toronto City. New datasets of videogame-related venues can be acquired and potentially merged with what was retrieved from Foursquare. One may also use cluster algorithm to further analyze the data, for the purpose of getting more insights. ¶

Part 5. Acknowledgement

My project benefited from these works and tools:
https://towardsdatascience.com/tuning-in-to-nycs-music-neighborhoods-efb7ae77a4cd¶
https://developer.foursquare.com/docs/resources/categories
https://en.wikipedia.org/wiki/Demographics_of_Toronto_neighbourhoods

Part 6. Appendix

Appendix 1: the function to call Google geocodes

```
# this function gets coordination info of each neighbourhood in df, and if no info is returned from
# geocode, the neighbourhood is recored in add.

def getLL(dataframe):
    add = []
    lat = []
    lng = []

    for address in df.Name:
        try:
            #address = dataframe.iloc[i,0] + ', Toronto'
            inputAddress = address + ', Toronto'
            geolocator = Nominatim(user_agent="foursquare_agent")
            #my understanding is that one call to the serve usually get timed out.
            #setting timeout to 15 makes less 'serve timed out' errors.
            location = geolocator.geocode(inputAddress, timeout =15)
            latitude = location.latitude
            longitude = location.longitude
            lat.append(latitude)
            lng.append(longitude)
        except Exception as e:
            print('Error, skipping address: ' + address, e)
            add.append(address)

    df_geocodes = pd.DataFrame({'Latitude':lat, 'Longitude':lng})

    return df_geocodes, add
```

Appendix 2: the function to remove neighborhoods with no coordination

```
# remove neighbourhoods with no coordination info.
def removeNeighbourhood(dataframe, add):
    delarry = []
    for name in add:
        for i in range (0,len(dataframe)):
            if name == dataframe['Name'].iloc[i]:
                delarry.append(i)

    # drop the rows
    dataframe.drop(dataframe.index[delarry], axis = 0, inplace = True)

    return dataframe
```

Appendix 3: the function to request data from Foursquare

```python
def getCompetitors(dataframe):
    endpoint = 'https://api.foursquare.com/v2/venues/search?'
    categoryIds = ['4bf58dd8d48988d122951735','4bf58dd8d48988d10b951735']
    categoryId = ','.join(categoryIds)
    radius = 1000
    limit = 50

    # create an empty list to collect venues
    venue_list = []
    count_list = []
    for i in range(0, len(df)):
        lat = df.iloc[i,3]
        lng = df.iloc[i,4]

        url = createURL(endpoint, CLIENT_ID, CLIENT_SECRET, VERSION, lat, lng, radius, categoryId, limit)
        results = requests.get(url).json()['response']['venues']

        # c stores total venue number before this neighbourhood
        c = len(venue_list)
        for item in results:
            venue_name = item['name']
            venue_category = item['categories'][0]['name']
            venue_lat = item['location']['lat']
            venue_lng = item['location']['lng']
            # put N/A if a venue has no such info
            try:
                venue_city = item['location']['city']
            except:
                venue_city = 'N/A'

            try:
                venue_state = item['location']['state']
            except:
                venue_state = 'N/A'

            venue_list.append([df.iloc[i,0],
                               df.iloc[i,3],
                               df.iloc[i,4],
                               venue_name,
                               venue_category,
                               venue_lat,
                               venue_lng,
                               venue_city,
                               venue_state
                              ])

        #this is the number of venues in this neighbourhood
        count = len(venue_list) - c
        count_list.append([df.iloc[i,0], count])

    nearby_venues = pd.DataFrame(venue_list,
                                 columns = ['Neighbourhood',
                                            'Latitude',
                                            'Longitude',
                                            'Venue Name',
                                            'Venue Category',
                                            'Venue Latitude',
                                            'Venue Longitude',
                                            'Venue City',
                                            'Venue State'
                                           ]
                                )

    venue_count = pd.DataFrame(count_list, columns = ['Neighbourhood','count'])

    return nearby_venues, venue_count
```