Business Location Explore in Toronto

This is Peer-graded Assignment for Course Applied Data Science Capstone (https://www.coursera.org/learn/applied-data-science-capstone/home/welcome), Week 4/5

Contents

1. Introduction

1.1 Business Requests

The location problem has been the challenge for many businesses starts for a long time. Many academic and industrial approaches focus on this problem. In this project we're trying to answer the question, where is the realistic location to start a new business based on existing data. We use Chinese restaurant as the business category to apply machine learning to help investors to make a better decision of location choice in downtown Toronto.

We assume that an investor wants to start a new business to serve Chinese food in downtown Toronto due to its density of population, higher average income, as well as the diversity of culture.

1.2 Analytic Approach

A good location should satisfy the two criteria at the same time: (1) Sufficient demand (2) Insufficient support

To address the first criteria, we could assume that if in some locations exists many restaurants business, there should have a good demand for foodservice in that location.

As per the second criteria, if we could hardly find a Chinese restaurant in the area, then we could say that the support is insufficient.

In summary, we need to gain the data of the food services venue information in the area, as well as their categories.

More specifically, we want to know how many restaurants in specific areas, how many of them provide Chinese food or Asian food. Based on this information we want to find out the most interesting area which has sufficient demand and insufficient support for Chinese food.

2. Data Collection

Majority we will use data provided by FourSqare (https://foursquare.com/) to perform our analysis.

FourSquare is a location technology platform to allow developers to fetch the location data, as well as venues information. With the free account one can make 100K calls per day to access their 105M+ points of interest data.

Aiming to the business request described above, we will collect all restrants information in downtown Toronto, and find out their distribution and rating, etc.

2.1 Scrape location info in Toronto

We use pandas function read_html to get postal code list in Toronto, as well as the neibourhoods.

In [3]:
```
import pandas as pd
import numpy as np
import requests
import pickle
import folium
import re
```

Step(1) Fetch postal code in Toronto

We get the list of postal codes in Toronto from the Wiki page: List_of_postal_codes_of_Canada (https://en.wikipedia.org/wiki/List_of_postal_codes_of_Canada:_M), and perform some simple data cleaning job.

In [4]:
```
url = 'https://en.wikipedia.org/wiki/List_of_postal_codes_of_Canada:_M'
dfs = pd.read_html(url)
for idx, df in enumerate(dfs):
    print('DataFrame[{}]:{}'.format(idx, df.shape))

dfs[0].head()
```

```
DataFrame[0]:(180, 3)
DataFrame[1]:(4, 18)
DataFrame[2]:(2, 18)
```

Out[4]:

	Postal code	Borough	Neighborhood
0	M1A	Not assigned	NaN
1	M2A	Not assigned	NaN
2	M3A	North York	Parkwoods
3	M4A	North York	Victoria Village
4	M5A	Downtown Toronto	Regent Park / Harbourfront

Oberviously the first data frame is what we need.

Lets remove those 'Not assigned' rows as per column Borough, and check the duplications for Postal Code.

In [5]:
```
df = dfs[0]
df = df[df['Borough'] != 'Not assigned']
print(df.shape)
len(df['Postal code'].unique())
```

```
(103, 3)
```

Out[5]:
```
103
```

We are good there's no duplication in column Postal Code.

Chose Postal code as index.

In [6]:
```
df.set_index('Postal code', inplace=True)
df.head()
```

Out[6]:

Postal code	Borough	Neighborhood
M3A	North York	Parkwoods
M4A	North York	Victoria Village
M5A	Downtown Toronto	Regent Park / Harbourfront
M6A	North York	Lawrence Manor / Lawrence Heights
M7A	Downtown Toronto	Queen's Park / Ontario Provincial Government

Step(2) Attaching geo info for each postal code

We could get geospatial information for each postal code via the online csv file: [Geospatial_data (http://cocl.us/Geospatial_data)](http://cocl.us/Geospatial_data), then attach it to existing data set.

In [7]:
```
url = 'http://cocl.us/Geospatial_data'
geo_info = pd.read_csv(url)
geo_info.set_index('Postal Code', inplace=True)
print(geo_info.shape)
geo_info.head()
```

```
(103, 2)
```

Out[7]:

Postal Code	Latitude	Longitude
M1B	43.806686	-79.194353
M1C	43.784535	-79.160497
M1E	43.763573	-79.188711
M1G	43.770992	-79.216917
M1H	43.773136	-79.239476

Now we can merge these two data set into one.

In [8]:
```
df = df.merge( geo_info, left_index= True, right_index = True)

df.index.name='Postal Code'
df.head()
```

Out[8]:

Postal Code	Borough	Neighborhood	Latitude	Longitude
M3A	North York	Parkwoods	43.753259	-79.329656
M4A	North York	Victoria Village	43.725882	-79.315572
M5A	Downtown Toronto	Regent Park / Harbourfront	43.654260	-79.360636
M6A	North York	Lawrence Manor / Lawrence Heights	43.718518	-79.464763
M7A	Downtown Toronto	Queen's Park / Ontario Provincial Government	43.662301	-79.389494

Step (3) Visualization areas on map

We can have a general idea of the area by visulize these data on map.

First we get the center point of the map:

In [9]:
```
lat, lng = df[['Latitude','Longitude']].max() + df[['Latitude','Longitude']].min()

lat, lng  = lat /2, lng /2
lat, lng
```

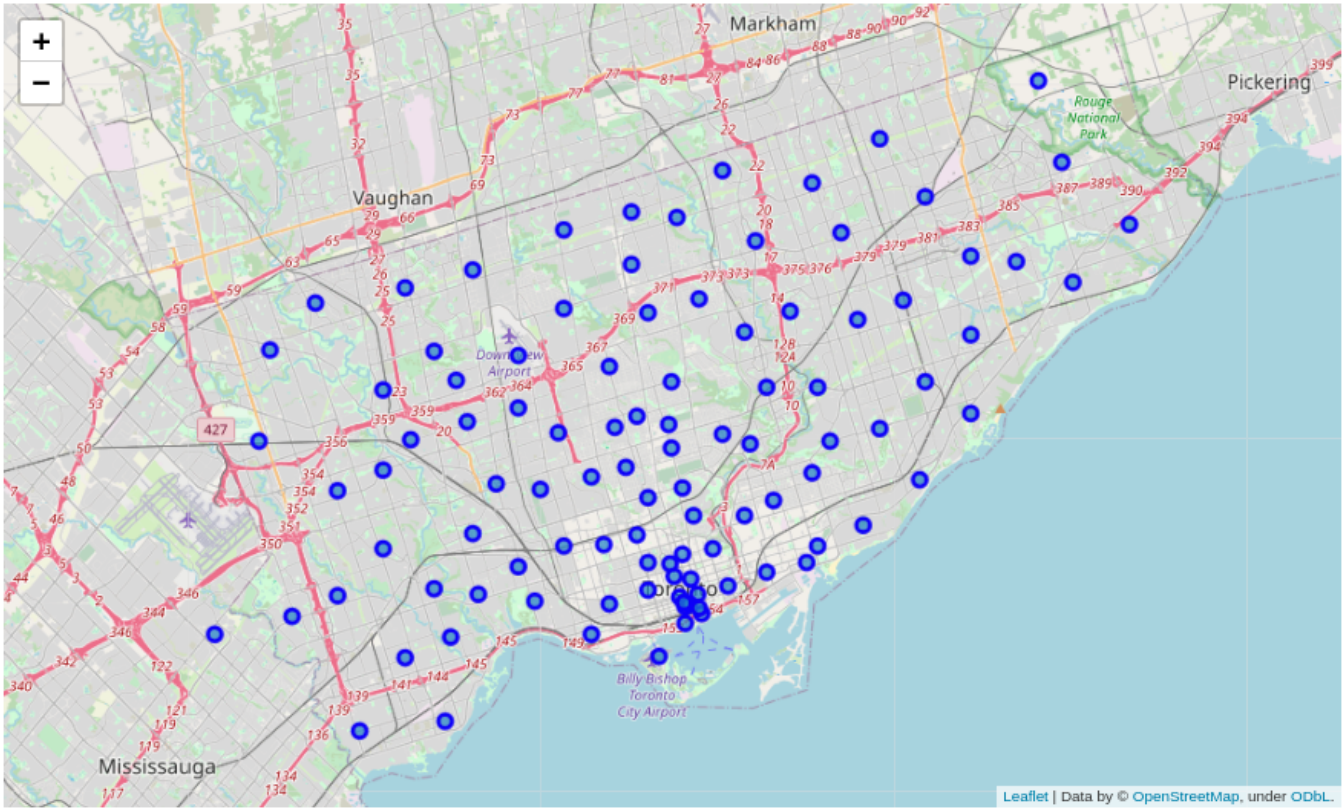
Out[9]:
```
(43.71926920000001, -79.38815804999999)
```

Then we can illustrate them on the map:

In [ ]:
```
# create map of Toronto using latitude and longitude values
map_toronto = folium.Map(location=[lat, lng], zoom_start=11)

# add markers to map
for idx, r  in df.iterrows():
    lat, lng, bor, postalcode  =  r['Latitude'], r['Longitude'],r['Borough'], r.name
    label = '{}, {}'.format(bor, postalcode)
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius=5,
        popup=label,
        color='blue',
        fill=True,
        fill_color='#3186cc',
        fill_opacity=0.7,
        parse_html=False).add_to(map_toronto)

map_toronto
```



2.2 Fetch all 'FOOD' venues in Toronto

In this step we will employee FourSquare API to fetch all venues under category 'FOOD' in Toronto.

Step(1) First we set API credentials

In [11]:
```
#hide this cell while exporting#
api_credentials = {'CLIENT_ID':'Y5FK5TTSXY24B0DDCUJBGCWCL2B01DYMXZRFOXROSYNCSSYJ',
                   'CLIENT_SECRET':'TUOXNZ2M4NVKE4BLA1N0XOV5CC54GIWOD0D4RF4A3CFMR3MV
                   'VERSION':'20180605'}
api_credentials
```

Out[11]:
```
{'CLIENT_ID': 'Y5FK5TTSXY24B0DDCUJBGCWCL2B01DYMXZRFOXROSYNCSSYJ',
 'CLIENT_SECRET': 'TUOXNZ2M4NVKE4BLA1N0XOV5CC54GIWOD0D4RF4A3CFMR3MV',
 'VERSION': '20180605'}
```

Step(2) Fetch data via FourSquare API

From FourSquare API doc Venue Categories (https://developer.foursquare.com/docs/build-with-foursquare/categories/), we can tell the following Foursquare Venue Category Hierarchy, as well as their ID.

- Category: Food: 4d4b7105d754a06374d81259
  - Asian Restaurant: 4bf58dd8d48988d142941735
    - Chinese Restaurant: 4bf58dd8d48988d145941735

In [12]:
```
CATEGORYID = '4d4b7105d754a06374d81259'
```

In [36]:
```
# search venues at specific postcal code
# return datafrom with column : id, name, lat, lng, id, primaryCategory, categories

def venuelist(postcode, df=df, radius = 1200 ):
    name, _, lat, lng  =  df.loc[postcode]
    url = '{}{}{}{}'.format('https://api.foursquare.com/v2/venues/search?',
          '&client_id={id}&client_secret={pw}&v={v}'.format(id = api_credentials['CLI
                                                           pw = api_credentials['CLI
                                                           v  = api_credentials['VER
            '&ll={},{}&radius={}'.format(lat, lng, radius),
            '&categoryId={}'.format(CATEGORYID)
            )
    results = requests.get(url).json()['response']['venues']
    d = pd.json_normalize(results)
    for idx, row in d.iterrows():
        for catidx, cat in enumerate(row['categories']):
            if cat['primary']:
                d.loc[idx, 'PrimaryCategory'] = cat['name']
            else:
                d.loc[idx, 'Category-{}'.format(catidx)] = cat['name']

    d.drop(['categories', 'referralId', 'hasPerk', 'location.cc', 'location.crossStre
             'location.distance','location.city', 'location.state',
            'location.formattedAddress', 'location.neighborhood'
           ], axis =1, errors = 'ignore', inplace=True)

    with open( postcode, 'wb') as f:
        pickle.dump(d, f)

    return d
```

In [ ]:
```
venues = pd.DataFrame()

for i in range(df.shape[0]):
    postcode = df.iloc[i].name
#     print('Process..#{}:{}'.format(i, postcode))

#     with open(postcode, 'rb') as f:
#         d = pickle.load(f)
#         venues = venues.append(d, ignore_index=True)
     venues = vennus.append( venuelist( df.iloc[i].name ), ignore_index=True)
```

Removing duplicated venues by id

In [43]:
```
venues.drop_duplicates('id', inplace=True)

with open('Venues', 'wb') as f:
    pickle.dump(venues, f)
print(venues.shape)
```

```
(1879, 9)
```

Step(3) Review venues contains **Restaurant** in their Name

Lets' foucs on those Restaurants in the venues list, since the stackholder/investor's purpose is to open a restaurnat.

In [52]:
```
len(venues['PrimaryCategory'].unique())
```

Out[52]: 132

In [54]:
```
# Find the most frequency categories by Grouping by category and sorting by count des
categories_counts = venues[['id','PrimaryCategory']].groupby('PrimaryCategory').count
                                                    .sort_values(by='id', ascending=F

categories_counts.rename({'id':'count'}, axis =1, inplace=True)

categories_counts.head(10)
```

Out[54]:

PrimaryCategory	count
Coffee Shop	318
Pizza Place	123
Fast Food Restaurant	109
Bakery	95
Restaurant	88
Grocery Store	70
Café	69
Chinese Restaurant	63
Sandwich Place	53
Caribbean Restaurant	48

There are lots of categories under **FOOD**, most of them are caffee shop, Pizza Place, even many Grocery Stores are included in the search result.

Let's focust on those **real** Restaurants.

In [64]:
```
restaurants = venues[venues['PrimaryCategory'].str.contains('Restaurant')]
restaurants.shape
```

Out[64]: (775, 9)

Now lets see how many Asian/Chinese Restaurant here:

In [65]:
```
categories_counts.loc[['Asian Restaurant','Chinese Restaurant']]
```

Out[65]:

PrimaryCategory	count
Asian Restaurant	22
Chinese Restaurant	63

Seems like the category Hierarchy is not well defined.

In [69]:
```
restaurants_counts = restaurants.groupby('PrimaryCategory').count().sort_values('id'
restaurants_counts.rename({'id':'count'}, axis =1, inplace=True)
restaurants_counts.head(10)
```

Out[69]:

PrimaryCategory	count
Fast Food Restaurant	109
Restaurant	88
Chinese Restaurant	63
Caribbean Restaurant	48
Italian Restaurant	40
Indian Restaurant	35
Middle Eastern Restaurant	33
Sushi Restaurant	31
Vietnamese Restaurant	30
Japanese Restaurant	27

By reviewing the whole list, we setup a mapping on top of the current category hierarchy. We will use the mapping for further analysis.

In [80]:
```
AsianRestaurants = ['Asian Restaurant','Burmese Restaurant','Sushi Restaurant', 'Viet
                    'Japanese Restaurant','Korean Restaurant','Thai Restaurant',
                    'Japanese Curry Restaurant', 'Indian Chinese Restaurant',
                   ]
ChineseRestaurants = ['Chinese Restaurant','Cantonese Restaurant','Hakka Restaurant'
                      'Tibetan Restaurant','Taiwanese Restaurant','Udon Restaurant',
                      'Hong Kong Restaurant','Hotpot Restaurant',
                     ]
```

2.3 Show venues on map

Step (1) Find the center of the map

In [86]:
```
lat, lng = restaurants[['location.lat','location.lng']].max() + restaurants[['locatio
lat, lng  = lat /2, lng /2
lat, lng
```

Out[86]:
```
(43.709245211612455, -79.38525389221354)
```

Step (2) Mark the venues on maps

We use three colors in the visulization:

- Red: Chinese Restaurants
- Blue: Asian Restaurants, excluding Chinese Restaurants
- Green: All other restaurants

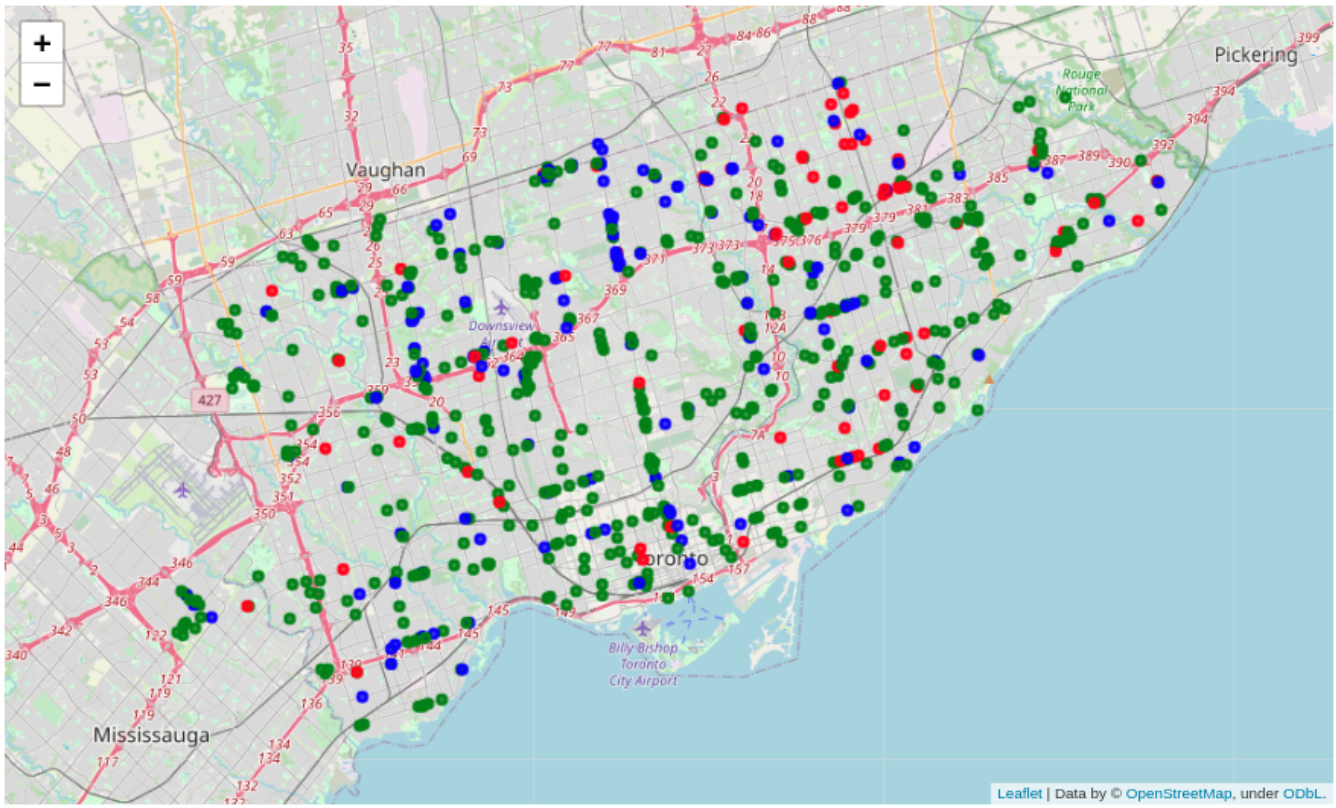
In [ ]:
```
# create map of New York using latitude and longitude values
map_all_venues = folium.Map(location=[lat, lng], zoom_start=11)

# add markers to map
for idx, r  in restaurants.iterrows():
    lat, lng, name, category =  r['location.lat'], r['location.lng'],r['name'], r['Pr
    if category in ChineseRestaurants:
        color = 'red'
    elif category in AsianRestaurants:
        color = 'blue'
    else:
        color = 'green'

    label = '{}, {}'.format(name, category)
    label = folium.Popup(label, parse_html=True)

    folium.CircleMarker(
        [lat, lng],
        radius=3,
        popup=label,
        color=color,
        fill=True,
        fill_color= color, #'#3186cc',
        fill_opacity=0.7,
        parse_html=False).add_to(map_all_venues)

map_all_venues
```



3. Methodology

In [1]:
```
import pandas as pd
import numpy as np
import requests
```

4. Result

To Be Continue....

5. Discussion

To Be Continue....

6. Conclusion

To Be Continue....