Capstone: Find the best neighborhood in Toronto to open a Restaurant Supply Store

1. Load all the Data from all the various sources.

1.1 Toronto neighborhoods broken down by postal code.

In [1]:

```
# Load the required libraries
import numpy as np
import pandas as pd
import requests
from bs4 import BeautifulSoup

# Found the table using beautifulsoup and used Pandas to read it in.
#res = requests.get('https://en.wikipedia.org/wiki/List_of_postal_codes_of_Canada:_M')
#soup = BeautifulSoup(res.content,'lxml')
#table = soup.find_all('table')[0]
#df = pd.read_html(str(table))


# WRANGLE/Transform THE DATA
# Convert the list back into a dataframe
#data = pd.DataFrame(df[0])
data=pd.read_excel("canada.xlsx")
# Rename the columns as instructed
#data = data.rename(columns={0:'PostalCode', 1:'Bourough', 2:'Neighborhood'})

# Get rid of the first row which contained the table headers from the webpage
data = data.iloc[1:]


# Only process the cells that have an assigned borough. Ignore cells with a borough tha
t is Not assigned.
data = data[~data['Bourough'].str.contains('Not assigned')]


# More than one neighborhood can exist in one postal code area.
#For example, in the table on the Wikipedia page, you will notice
#that M5A is listed twice and has two neighborhoods: Harbourfront
#and Regent Park. These two rows will be combined into one row with
#the neighborhoods separated with a comma
df2=data.groupby(['PostalCode', 'Bourough']).apply(lambda group: ', '.join(group['Neigh
borhood']))


# Convert the Series back into a DataFrame and put the 'Neighbourhood' column label bac
k in
df2=df2.to_frame().reset_index()
df2 = df2.rename(columns={0:'Neighborhood'})

# If a cell has a borough but a Not assigned neighborhood, then the neighborhood will b
e the same as the borough.
df2.loc[df2.Neighborhood == 'Not assigned', 'Neighborhood' ] = df2.Bourough

# Display the DataFrame
df2.head()
```

Out[1]:

	PostalCode	Bourough	Neighborhood
0	M1B	Scarborough	Malvern, Rouge
1	M1C	Scarborough	Rouge Hill, Port Union, Highland Creek
2	M1E	Scarborough	Guildwood, Morningside, West Hill
3	M1G	Scarborough	Woburn
4	M1H	Scarborough	Cedarbrae

In [2]:

```
#1.1.1 Load Toronto geospatial cooridinates and merge to Toronto Postal Code Data
gf = pd.read_csv('to_geo_space.csv')

#rename the coloumns so the match
gf = gf.rename(columns={'Postal Code':'PostalCode'})

#Merge the Toronto data with geo cooridinate data
gf_new = pd.merge(df2, gf, on='PostalCode', how='inner')

# display the new dataframe
gf_new.head()
```

Out[2]:

	PostalCode	Bourough	Neighborhood	Latitude	Longitude
0	M1B	Scarborough	Malvern, Rouge	43.806686	-79.194353
1	M1C	Scarborough	Rouge Hill, Port Union, Highland Creek	43.784535	-79.160497
2	M1E	Scarborough	Guildwood, Morningside, West Hill	43.763573	-79.188711
3	M1G	Scarborough	Woburn	43.770992	-79.216917
4	M1H	Scarborough	Cedarbrae	43.773136	-79.239476

In [3]:

```
#1.2 Toronto neighborhoods populations broken down by postal code
# Load this data from Stats Canada
df_pop = pd.read_csv('https://www12.statcan.gc.ca/census-recensement/2016/dp-pd/hlt-fst/pd-pl/Tables/File.cfm?T=1201&SR=1&RPP=9999&PR=0&CMA=0&CSD=0&S=22&O=A&Lang=Eng&OFT=CSV',encoding = 'unicode_escape')
# Rename the columns appropiatley
df_pop = df_pop.rename(columns={'Geographic code':'PostalCode', 'Geographic name':'PostalCod2', 'Province or territory':'Province', 'Incompletely enumerated Indian reserves and Indian settlements, 2016':'Incomplete', 'Population, 2016':'Population_2016', 'Total private dwellings, 2016':'TotalPrivDwellings', 'Private dwellings occupied by usual residents, 2016':'PrivDwellingsOccupied'})
df_pop= df_pop.drop(columns=['PostalCod2', 'Province', 'Incomplete', 'TotalPrivDwellings', 'PrivDwellingsOccupied'])

# Get rid of the first row
df_pop = df_pop.iloc[1:]
df_pop.head()
```

Out[3]:

	PostalCode	Population_2016
1	A0A	46587.0
2	A0B	19792.0
3	A0C	12587.0
4	A0E	22294.0
5	A0G	35266.0

In [4]:

```
#1.2.1 Merge Toronto Neighbourhood populations data with Toronto Postal Code data
#Merge the Toronto Pop data with geo postalcode data
gf_new
gf_new = pd.merge(df_pop, gf_new, on='PostalCode', how='right')
# sort on population
gf_new = gf_new.sort_values(by=['Population_2016'], ascending=False)

# display the new dataframe
gf_new.head()
```

Out[4]:

	PostalCode	Population_2016	Bourough	Neighborhood	Latitude	Longitude
22	M2N	75897.0	North York	Willowdale, Willowdale East	43.770120	-79.408493
0	M1B	66108.0	Scarborough	Malvern, Rouge	43.806686	-79.194353
18	M2J	58293.0	North York	Fairview, Henry Farm, Oriole	43.778517	-79.346556
100	M9V	55959.0	Etobicoke	South Steeles, Silverstone, Humbergate, Jamest...	43.739416	-79.588437
14	M1V	54680.0	Scarborough	Milliken, Agincourt North, Steeles East, L'Amo...	43.815252	-79.284577

In [5]:

```
#1.3 Toronto neighborhoods average after tax income broken down by postal code
# It was easier to extract this data manually from Stats Canada and load it then it was
to scrape it.
# It was only accessible from indeividual queries per postal code on the statscan web s
ite.
df_income = pd.read_csv('TorontoAvgIncomeByPC.csv',encoding = 'unicode_escape')
# Rename the after tax income column to a more maanageable name
df_income = df_income.rename(columns={"after-tax income of households in 2015":"AfterTa
xIncome2015"})
df_income.head()
```

Out[5]:

	PostalCode	AfterTaxIncome2015
0	M2P	115237
1	M5M	111821
2	M4N	109841
3	M5R	108271
4	M8X	97210

In [6]:

```
#1.3.1 Merge Toronto Neighbourhood income data with Toronto Postal Code data
#Merge the Toronto Income data with geo postalcode data

gf_new = pd.merge(df_income, gf_new, on='PostalCode', how='right')
# get rid of the Nulls
gf_new = gf_new.replace('Null', 0)
#gf_new cast as float
gf_new['AfterTaxIncome2015'] = gf_new['AfterTaxIncome2015'].astype('float64')
# Sort on Income
gf_new = gf_new.sort_values(by=['AfterTaxIncome2015'], ascending=False)

# display the new dataframe
gf_new.to_csv('TO_Affluence.csv')
gf_new.head(10)
```

Out[6]:

	PostalCode	AfterTaxIncome2015	Population_2016	Bourough	Neighborhood	Latitude
0	M2P	115237.0	7843.0	North York	York Mills West	43.752758
1	M5M	111821.0	25975.0	North York	Bedford Park, Lawrence Manor East	43.733283
2	M4N	109841.0	15330.0	Central Toronto	Lawrence Park	43.728020
3	M5R	108271.0	26496.0	Central Toronto	The Annex, North Midtown, Yorkville	43.672710
4	M8X	97210.0	10787.0	Etobicoke	The Kingsway, Montgomery Road, Old Mill North	43.653654
5	M2L	96512.0	11717.0	North York	York Mills, Silver Hills	43.757490
6	M4G	94853.0	19076.0	East York	Leaside	43.709060
7	M1C	93943.0	35626.0	Scarborough	Rouge Hill, Port Union, Highland Creek	43.784535
8	M9B	91110.0	32400.0	Etobicoke	West Deane Park, Princess Gardens, Martin Grov...	43.650943
9	M3B	90841.0	13324.0	North York	Don Mills	43.745906

1.4 What is the Canadian National Average After Tax Income. Again obtained from the Stats Canada Website Canadian families and unattached individuals had a median after-tax income of $57,000 in 2016.

1.5 Toronto list of Restaurants or Venues that could potentially use Restaurant Equipment.

In [7]:

```
#FourSquare Credentials

CLIENT_ID = 'LCTZXNUJDOVCVNYUPN33GJVILULU1WFXWBW3AN151N3ZGQOV' # your Foursquare ID


CLIENT_SECRET = 'ICKMDW5H1RKQUB1TVQ4MOVDHUYBE2VKHX4OJOI43Q44SDBC5' # your Foursquare Se
cret


VERSION = '20180605' # Foursquare API version
```

In [8]:

```
#Let's explore neighborhoods in our dataframe.
import requests # library to handle requests
from pandas.io.json import json_normalize # tranform JSON file into a pandas dataframe

LIMIT = 200 # limit of number of venues returned by Foursquare API

radius = 500 # define radius

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

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

        # 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)
```

In [9]:

```
# Toronto Bouroughs
TO_data = gf_new
TO_data.head()
```

Out[9]:

	PostalCode	AfterTaxIncome2015	Population_2016	Bourough	Neighborhood	Latitude	L
0	M2P	115237.0	7843.0	North York	York Mills West	43.752758	-7
1	M5M	111821.0	25975.0	North York	Bedford Park, Lawrence Manor East	43.733283	-7
2	M4N	109841.0	15330.0	Central Toronto	Lawrence Park	43.728020	-7
3	M5R	108271.0	26496.0	Central Toronto	The Annex, North Midtown, Yorkville	43.672710	-7
4	M8X	97210.0	10787.0	Etobicoke	The Kingsway, Montgomery Road, Old Mill North	43.653654	-7

In [10]:

```
#1.5.1 Get all the Venues in Toronto.
# Get all of the Venues
TO_venues = getNearbyVenues(names=TO_data['Neighborhood'],
                                    latitudes=TO_data['Latitude'],
                                    longitudes=TO_data['Longitude']
                                   )
```

```
York Mills West
Bedford Park, Lawrence Manor East
Lawrence Park
The Annex, North Midtown, Yorkville
The Kingsway, Montgomery Road, Old Mill North
York Mills, Silver Hills
Leaside
Rouge Hill, Port Union, Highland Creek
West Deane Park, Princess Gardens, Martin Grove, Islington, Cloverdale
Don Mills
CN Tower, King and Spadina, Railway Lands, Harbourfront West, Bathurst Qua
y, South Niagara, Island airport
East Toronto, Broadview North (Old East York)
Upper Rouge
Birch Cliff, Cliffside West
Little Portugal, Trinity
Rosedale
Moore Park, Summerhill East
Roselawn
The Beaches
Mimico NW, The Queensway West, South of Bloor, Kingsway Park South West, R
oyal York South West
North Toronto West, Lawrence Park
Runnymede, Swansea
Berczy Park
Forest Hill North & West, Forest Hill Road Park
Woodbine Heights
Summerhill West, Rathnelly, South Hill, Forest Hill SE, Deer Park
Harbourfront East, Union Station, Toronto Islands
Richmond, Adelaide, King
St. James Town
India Bazaar, The Beaches West
Eringate, Bloordale Gardens, Old Burnhamthorpe, Markland Wood
Kingsview Village, St. Phillips, Martin Grove Gardens, Richview Gardens
Parkdale, Roncesvalles
Islington Avenue, Humber Valley Village
Fairview, Henry Farm, Oriole
Studio District
Hillcrest Village
Westmount
Davisville
Christie
Alderwood, Long Branch
Bathurst Manor, Wilson Heights, Downsview North
Old Mill South, King's Mill Park, Sunnylea, Humber Bay, Mimico NE, The Que
ensway East, Royal York South East, Kingsway Park South East
Malvern, Rouge
Golden Mile, Clairlea, Oakridge
Humewood-Cedarvale
Cliffside, Cliffcrest, Scarborough Village West
Milliken, Agincourt North, Steeles East, L'Amoreaux East
Northwest, West Humber - Clairville
Bayview Village
High Park, The Junction South
Steeles West, L'Amoreaux West
Glencairn
Humber Summit
Agincourt
Parkwoods
The Danforth West, Riverdale
Humberlea, Emery
```

```
Downsview
Dufferin, Dovercourt Village
Guildwood, Morningside, West Hill
Downsview
South Steeles, Silverstone, Humbergate, Jamestown, Mount Olive, Beaumond H
eights, Thistletown, Albion Gardens
Willowdale, Willowdale East
Caledonia-Fairbanks
Wexford, Maryvale
Willowdale, Newtonbrook
Parkview Hill, Woodbine Gardens
Cedarbrae
New Toronto, Mimico South, Humber Bay Shores
Davisville North
Runnymede, The Junction North
Dorset Park, Wexford Heights, Scarborough Town Centre
Don Mills
Willowdale, Willowdale West
Woburn
North Park, Maple Leaf Park, Upwood Park
Scarborough Village
Downsview
Northwood Park, York University
Victoria Village
Clarks Corners, Tam O'Shanter, Sullivan
Lawrence Manor, Lawrence Heights
Kennedy Park, Ionview, East Birchmount Park
Del Ray, Mount Dennis, Keelsdale and Silverthorn
Brockton, Parkdale Village, Exhibition Place
University of Toronto, Harbord
Regent Park, Harbourfront
Church and Wellesley
Weston
Garden District, Ryerson
Downsview
Central Bay Street
Thorncliffe Park
Kensington Market, Chinatown, Grange Park
St. James Town, Cabbagetown
Commerce Court, Victoria Hotel
Canada Post Gateway Processing Centre
Queen's Park, Ontario Provincial Government
First Canadian Place, Underground city
Toronto Dominion Centre, Design Exchange
Business reply mail Processing Centre, South Central Letter Processing Pla
nt Toronto
Stn A PO Boxes
```

In [11]:

```
#Let's count the number of Venues per Neighorhood
TO_venues.groupby('Neighborhood').count()
```

Out[11]:

Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
Agincourt	5	5	5	5	5	5
Alderwood, Long Branch	7	7	7	7	7	7
Bathurst Manor, Wilson Heights, Downsview North	21	21	21	21	21	21
Bayview Village	4	4	4	4	4	4
Bedford Park, Lawrence Manor East	22	22	22	22	22	22
...	...	...	...	...	...	...
Willowdale, Willowdale West	5	5	5	5	5	5
Woburn	4	4	4	4	4	4
Woodbine Heights	8	8	8	8	8	8
York Mills West	2	2	2	2	2	2
York Mills, Silver Hills	1	1	1	1	1	1

96 rows × 6 columns

In [12]:

```
# Let's pick out restaurants from Venue Categories

print('Unique Venue Categories:')
list(TO_venues['Venue Category'].unique())
```

Unique Venue Categories:

Out[12]:

```
['Convenience Store',
 'Park',
 'Café',
 'Restaurant',
 'Sushi Restaurant',
 'Comfort Food Restaurant',
 'Thai Restaurant',
 'Indian Restaurant',
 'Italian Restaurant',
 'Coffee Shop',
 'Pub',
 'Juice Bar',
 'Greek Restaurant',
 'Pharmacy',
 'Grocery Store',
 'Sandwich Place',
 'Butcher',
 'American Restaurant',
 'Liquor Store',
 'Pizza Place',
 'Locksmith',
 'Swim School',
 'Bus Line',
 'Middle Eastern Restaurant',
 'BBQ Joint',
 'Burger Joint',
 'Donut Shop',
 'History Museum',
 'Pool',
 'River',
 'Martial Arts School',
 'Sports Bar',
 'Sporting Goods Shop',
 'Fish & Chips Shop',
 'Bike Shop',
 'Supermarket',
 'Pet Store',
 'Smoothie Shop',
 'Shopping Mall',
 'Department Store',
 'Dessert Shop',
 'Bank',
 'Brewery',
 'Beer Store',
 'Breakfast Spot',
 'Furniture / Home Store',
 'Electronics Store',
 'Mexican Restaurant',
 'Bagel Shop',
 'Bar',
 'Construction & Landscaping',
 'Print Shop',
 'Caribbean Restaurant',
 'Gym',
 'Japanese Restaurant',
 'Airport',
 'Airport Lounge',
 'Harbor / Marina',
 'Airport Food Court',
```

```
    'Airport Gate',
    'Plane',
    'Boutique',
    'Rental Car Location',
    'Airport Terminal',
    'Airport Service',
    'Sculpture Garden',
    'Boat or Ferry',
    'Intersection',
    'General Entertainment',
    'Skating Rink',
    'College Stadium',
    'Ice Cream Shop',
    'Wine Bar',
    'Asian Restaurant',
    'Cuban Restaurant',
    'Korean Restaurant',
    'Cocktail Bar',
    'Art Gallery',
    'New American Restaurant',
    'Theater',
    'Yoga Studio',
    'French Restaurant',
    'Record Shop',
    'Vietnamese Restaurant',
    'Vegetarian / Vegan Restaurant',
    "Men's Store",
    'Bakery',
    'Malay Restaurant',
    'Cupcake Shop',
    'Gift Shop',
    'Miscellaneous Shop',
    'Diner',
    'Dog Run',
    'Southern / Soul Food Restaurant',
    'Playground',
    'Trail',
    'Garden',
    'Music Venue',
    'Health Food Store',
    'Neighborhood',
    'Wings Joint',
    'Discount Store',
    'Supplement Shop',
    'Fast Food Restaurant',
    'Hardware Store',
    'Social Club',
    'Tanning Salon',
    'Kids Store',
    'Thrift / Vintage Store',
    'Burrito Place',
    'Salon / Barbershop',
    'Clothing Store',
    'Spa',
    'Chinese Restaurant',
    'Bookstore',
    'Falafel Restaurant',
    'Latin American Restaurant',
    'Gourmet Shop',
    'Indie Movie Theater',
    'Concert Hall',
```

```
    'Museum',
    'Beer Bar',
    'Farmers Market',
    'Fountain',
    'Basketball Stadium',
    'Bistro',
    'Seafood Restaurant',
    'Fish Market',
    'Jazz Club',
    'Tailor Shop',
    'Creperie',
    'Cheese Shop',
    'Beach',
    'Steakhouse',
    'Hotel',
    'Eastern European Restaurant',
    'Jewelry Store',
    'Curling Ice',
    'Dance Studio',
    'Athletics & Sports',
    'Fried Chicken Joint',
    'Light Rail Station',
    'Salad Place',
    'Lake',
    'Performing Arts Venue',
    'Plaza',
    'Deli / Bodega',
    'Bubble Tea Shop',
    'IT Services',
    'Roof Deck',
    'Aquarium',
    'Lounge',
    'Train Station',
    'Monument / Landmark',
    'Scenic Lookout',
    'Baseball Stadium',
    'Tea Room',
    'Hotel Bar',
    'Office',
    'Speakeasy',
    'Gym / Fitness Center',
    'Smoke Shop',
    'Food Court',
    'Colombian Restaurant',
    'Gastropub',
    'General Travel',
    'Mediterranean Restaurant',
    'Brazilian Restaurant',
    'Gluten-free Restaurant',
    'Cosmetics Shop',
    'Art Museum',
    "Women's Store",
    'Poke Place',
    'Modern European Restaurant',
    'Noodle House',
    'Building',
    'Event Space',
    'Nightclub',
    'Toy / Game Store',
    'Food Truck',
    'Camera Store',
```

```
    'Movie Theater',
    'German Restaurant',
    'Lingerie Store',
    'Moroccan Restaurant',
    'Belgian Restaurant',
    'Irish Pub',
    'Kitchen Supply Store',
    'Shopping Plaza',
    'Chocolate Shop',
    'Video Game Store',
    'Mobile Phone Shop',
    'Shoe Store',
    'Luggage Store',
    'Bus Station',
    'Metro Station',
    'Baseball Field',
    'Gay Bar',
    'Stationery Store',
    'Coworking Space',
    'Golf Course',
    'Gas Station',
    'Indoor Play Area',
    'Candy Store',
    'Baby Store',
    'Bridal Shop',
    'Frozen Yogurt Shop',
    'Soccer Field',
    'Field',
    'Tennis Court',
    'Hockey Arena',
    'Motel',
    'Drugstore',
    'Garden Center',
    'Flea Market',
    'Antique Shop',
    'Arts & Crafts Store',
    'Cajun / Creole Restaurant',
    'Food & Drink Shop',
    'Fruit & Vegetable Store',
    'Medical Center',
    'Ramen Restaurant',
    'Auto Garage',
    'Accessories Store',
    'Hakka Restaurant',
    'Dim Sum Restaurant',
    'Korean BBQ Restaurant',
    'Home Service',
    'Business Service',
    'Massage Studio',
    'Portuguese Restaurant',
    'Hobby Shop',
    'Turkish Restaurant',
    'Climbing Gym',
    'Stadium',
    'College Gym',
    'College Arts Building',
    'Distribution Center',
    'Historic Site',
    'Theme Restaurant',
    'Escape Room',
    'Ethiopian Restaurant',
```

```
    'Sake Bar',
    'Health & Beauty Service',
    'Afghan Restaurant',
    'Strip Club',
    'Comic Shop',
    'College Rec Center',
    'Other Great Outdoors',
    'Poutine Place',
    'Hookah Bar',
    'Warehouse Store',
    'Organic Grocery',
    'Dumpling Restaurant',
    'Gaming Cafe',
    'Doner Restaurant',
    'Filipino Restaurant',
    'Hospital',
    'Bed & Breakfast',
    'Snack Place',
    'Taiwanese Restaurant',
    'Market',
    'College Auditorium',
    'College Cafeteria',
    'Opera House',
    'Taco Place',
    'Skate Park',
    'Auto Workshop',
    'Recording Studio',
    'Molecular Gastronomy Restaurant',
    'Optical Shop',
    'Church',
    'Hostel',
    'Soup Place']
```

In [13]:

```
#1.5.2 Only add Restaurants as Venue Categories
# Here we manually pick out restaurants or 'features' from the unique venue list and th
at we want to examine for similiarity during clustering
rest_list = ['Steakhouse', 'Coffee Shop', 'Café', 'Ramen Restaurant', 'Indonesian Resta
urant', 'Restaurant', 'Japanese Restaurant',
             'Fast Food Restaurant', 'Sushi Restaurant', 'Vietnamese Restaurant', 'Pizz
a Place', 'Sandwich Place', 'Middle Eastern Restaurant',
             'Burger Joint', 'American Restaurant', 'Food Court', 'Wings Joint', 'Burri
to Place', 'Asian Restaurant', 'Deli / Bodega',
             'Greek Restaurant', 'Fried Chicken Joint', 'Airport Food Court', 'Chinese
 Restaurant', 'Breakfast Spot', 'Mexican Restaurant',
             'Indian Restaurant', 'Latin American Restaurant', 'Bar', 'Pub', 'Italian R
estaurant', 'French Restaurant', 'Ice Cream Shop',
             'Caribbean Restaurant', 'Gastropub', 'Thai Restaurant', 'Cajun / Creole Re
staurant', 'Diner', 'Dim Sum Restaurant', 'Seafood Restaurant',
             'Food & Drink Shop', 'Noodle House', 'Food', 'Fish & Chips Shop', 'Falafel
Restaurant', 'Gourmet Shop', 'Vegetarian / Vegan Restaurant',
             'South American Restaurant', 'Korean Restaurant', 'Cuban Restaurant', 'New
American Restaurant', 'Malay Restaurant', 'Mac & Cheese Joint',
             'Bistro', 'Southern / Soul Food Restaurant', 'Tapas Restaurant',  'Sports
 Bar', 'Polish Restaurant', 'Ethiopian Restaurant',
             'Creperie', 'Sake Bar', 'Persian Restaurant', 'Afghan Restaurant','Mediter
ranean Restaurant', 'BBQ Joint', 'Jewish Restaurant',
             'Comfort Food Restaurant',  'Hakka Restaurant', 'Food Truck', 'Taiwanese R
estaurant',  'Snack Place', 'Eastern European Restaurant',
             'Dumpling Restaurant', 'Belgian Restaurant', 'Arepa Restaurant', 'Taco Pla
ce', 'Doner Restaurant', 'Filipino Restaurant',
             'Hotpot Restaurant', 'Poutine Place', 'Salad Place',  'Portuguese Restaura
nt', 'Modern European Restaurant', 'Empanada Restaurant',
             'Irish Pub', 'Molecular Gastronomy Restaurant', 'German Restaurant', 'Braz
ilian Restaurant', 'Gluten-free Restaurant', 'Soup Place']

rest_pd = pd.DataFrame(rest_list)
#rest_pd
#rename the coloumns so the match
rest_pd = rest_pd.rename(columns={0:'Venue Category'})

#Join the 2 dataframes as instructed
TO_new = pd.merge(TO_venues, rest_pd, on='Venue Category', how='right')

# display the new dataframe
#TO_new

TO_new.groupby('Neighborhood').count()
```

Out[13]:

Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
Agincourt	2	2	2	2	2	2
Alderwood, Long Branch	5	5	5	5	5	5
Bathurst Manor, Wilson Heights, Downsview North	12	12	12	12	12	12
Bayview Village	3	3	3	3	3	3
Bedford Park, Lawrence Manor East	16	16	16	16	16	16
...	...	...	...	...	...	...
Westmount	4	4	4	4	4	4
Wexford, Maryvale	2	2	2	2	2	2
Willowdale, Willowdale East	21	21	21	21	21	21
Willowdale, Willowdale West	2	2	2	2	2	2
Woburn	3	3	3	3	3	3

78 rows × 6 columns

In [14]:

```
#1.5.3 OneHot encode and count restaurants
# one hot encoding
TO_new_onehot = pd.get_dummies(TO_new[['Venue Category']], prefix="", prefix_sep="")

# add neighborhood column back to dataframe
TO_new_onehot['Neighborhood'] = TO_new['Neighborhood']


# move neighborhood column to the first column
fixed_columns = [TO_new_onehot.columns[-1]] + list(TO_new_onehot.columns[:-1])
TO_new_onehot = TO_new_onehot[fixed_columns]

TO_new_onehot.head()
```

Out[14]:

	Neighborhood	Afghan Restaurant	Airport Food Court	American Restaurant	Arepa Restaurant	Asian Restaurant	BBQ Joint	Bar	Bel Restau
0	Bedford Park, Lawrence Manor East	0	0	0	0	0	0	0	
1	The Annex, North Midtown, Yorkville	0	0	0	0	0	0	0	
2	The Annex, North Midtown, Yorkville	0	0	0	0	0	0	0	
3	The Annex, North Midtown, Yorkville	0	0	0	0	0	0	0	
4	Don Mills	0	0	0	0	0	0	0	

5 rows × 91 columns

In [15]:

```
#Analyze each neighbourhood



TO_grouped = TO_new_onehot.groupby('Neighborhood').mean().reset_index()
TO_grouped.shape


TO_grouped.head()
```

Out[15]:

	Neighborhood	Afghan Restaurant	Airport Food Court	American Restaurant	Arepa Restaurant	Asian Restaurant	BBQ Joint	Bar	Bel Restau
0	Agincourt	0.0	0.0	0.0000	0.0	0.0	0.0	0.0	
1	Alderwood, Long Branch	0.0	0.0	0.0000	0.0	0.0	0.0	0.0	
2	Bathurst Manor, Wilson Heights, Downsview North	0.0	0.0	0.0000	0.0	0.0	0.0	0.0	
3	Bayview Village	0.0	0.0	0.0000	0.0	0.0	0.0	0.0	
4	Bedford Park, Lawrence Manor East	0.0	0.0	0.0625	0.0	0.0	0.0	0.0	

5 rows × 91 columns

2. Begin to Cluster

Use silhouette score to find optimal number of clusters to segment the data

In [16]:

```
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
import numpy as np


TO_grouped_clustering = TO_grouped.drop('Neighborhood', 1)

# Use silhouette score to find optimal number of clusters to segment the data
kclusters = np.arange(2,10)
results = {}
for size in kclusters:
    model = KMeans(n_clusters = size).fit(TO_grouped_clustering)
    predictions = model.predict(TO_grouped_clustering)
    results[size] = silhouette_score(TO_grouped_clustering, predictions)

best_size = max(results, key=results.get)
best_size
```

Out[16]:

```
5
```

In [17]:

```
#2.1 Run K means and segment data into clusters and generate labels
#import k-means from clustering stage
from sklearn.cluster import KMeans

# set number of clusters
kclusters = best_size


# run k-means clustering
kmeans = KMeans(n_clusters=kclusters, random_state=0).fit(TO_grouped_clustering)

# check cluster labels generated for each row in the dataframe
kmeans.labels_[0:10]
```

Out[17]:

```
array([3, 2, 0, 0, 0, 0, 0, 0, 0, 0])
```

In [18]:

```
def return_most_common_venues(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 = 10

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('{}{} Most Common Venue'.format(ind+1, indicators[ind]))
    except:
        columns.append('{}th Most Common Venue'.format(ind+1))

# create a new dataframe
neighborhoods_venues_sorted = pd.DataFrame(columns=columns)
neighborhoods_venues_sorted['Neighborhood'] = TO_grouped['Neighborhood']

for ind in np.arange(TO_grouped.shape[0]):
    neighborhoods_venues_sorted.iloc[ind, 1:] = return_most_common_venues(TO_grouped.il
oc[ind, :], num_top_venues)

neighborhoods_venues_sorted.head()
```

Out[18]:

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th M Comm Ven
0	Agincourt	Breakfast Spot	Latin American Restaurant	Food	Doner Restaurant	Dumpling Restaurant	Eastern European Restaurant	Empana Restaur
1	Alderwood, Long Branch	Pizza Place	Pub	Coffee Shop	Sandwich Place	Falafel Restaurant	Dim Sum Restaurant	Di
2	Bathurst Manor, Wilson Heights, Downsview North	Coffee Shop	Pizza Place	Sandwich Place	Diner	Middle Eastern Restaurant	Chinese Restaurant	Restaur
3	Bayview Village	Chinese Restaurant	Japanese Restaurant	Café	Wings Joint	Fish & Chips Shop	Dumpling Restaurant	East Europe Restaur
4	Bedford Park, Lawrence Manor East	Sandwich Place	Italian Restaurant	Coffee Shop	Pizza Place	Comfort Food Restaurant	Café	F

In [19]:

```
#2.2 Merge the Toronto data with geo coordinates data and make sure it's the right shap
e
#Merge the Toronto data with geo cooridinate data and make sure it's the right shape
TO_labels = pd.merge(TO_data,TO_grouped, on='Neighborhood', how='right')
TO_labels.shape


TO_labels = TO_labels.drop(columns=['Steakhouse', 'Coffee Shop', 'Café', 'Ramen Restaur
ant', 'Indonesian Restaurant', 'Restaurant', 'Japanese Restaurant',
            'Fast Food Restaurant', 'Sushi Restaurant', 'Vietnamese Restaurant', 'Pizz
a Place', 'Sandwich Place', 'Middle Eastern Restaurant',
            'Burger Joint', 'American Restaurant', 'Food Court', 'Wings Joint', 'Burri
to Place', 'Asian Restaurant', 'Deli / Bodega',
            'Greek Restaurant', 'Fried Chicken Joint', 'Airport Food Court', 'Chinese
 Restaurant', 'Breakfast Spot', 'Mexican Restaurant',
            'Indian Restaurant', 'Latin American Restaurant', 'Bar', 'Pub', 'Italian R
estaurant', 'French Restaurant', 'Ice Cream Shop',
            'Caribbean Restaurant', 'Gastropub', 'Thai Restaurant', 'Cajun / Creole Re
staurant', 'Diner', 'Dim Sum Restaurant', 'Seafood Restaurant',
            'Food & Drink Shop', 'Noodle House', 'Food', 'Fish & Chips Shop', 'Falafel
Restaurant', 'Gourmet Shop', 'Vegetarian / Vegan Restaurant',
            'South American Restaurant', 'Korean Restaurant', 'Cuban Restaurant', 'New
American Restaurant', 'Malay Restaurant', 'Mac & Cheese Joint',
            'Bistro', 'Southern / Soul Food Restaurant', 'Tapas Restaurant',  'Sports
 Bar', 'Polish Restaurant', 'Ethiopian Restaurant',
            'Creperie', 'Sake Bar', 'Persian Restaurant', 'Afghan Restaurant','Mediter
ranean Restaurant', 'BBQ Joint', 'Jewish Restaurant',
            'Comfort Food Restaurant',  'Hakka Restaurant', 'Food Truck', 'Taiwanese R
estaurant',  'Snack Place', 'Eastern European Restaurant',
            'Dumpling Restaurant', 'Belgian Restaurant', 'Arepa Restaurant', 'Taco Pla
ce', 'Doner Restaurant', 'Filipino Restaurant',
            'Hotpot Restaurant', 'Poutine Place', 'Salad Place',  'Portuguese Restaura
nt', 'Modern European Restaurant', 'Empanada Restaurant',
            'Irish Pub', 'Molecular Gastronomy Restaurant', 'German Restaurant', 'Braz
ilian Restaurant', 'Gluten-free Restaurant', 'Soup Place'])
TO_labels.head()
```

Out[19]:

	PostalCode	AfterTaxIncome2015	Population_2016	Bourough	Neighborhood	Latitude
0	M5M	111821.0	25975.0	North York	Bedford Park, Lawrence Manor East	43.733283
1	M5R	108271.0	26496.0	Central Toronto	The Annex, North Midtown, Yorkville	43.672710
2	M4G	94853.0	19076.0	East York	Leaside	43.709060
3	M1C	93943.0	35626.0	Scarborough	Rouge Hill, Port Union, Highland Creek	43.784535
4	M3B	90841.0	13324.0	North York	Don Mills	43.745906

In [22]:

```
print(neighborhoods_venues_sorted.shape)
print(TO_labels.shape)
print(TO_data.shape,TO_grouped.shape)
```

```
(78, 11)
(82, 7)
(103, 7) (78, 91)
```

In [40]:

```
#2.3 Add the KMeans Labels
TO_merged = TO_labels.iloc[:78]
print(TO_merged.shape) #82 * 7
print(kmeans.labels_.shape)
#neighborhoods_venues_sorted=neighborhoods_venues_sorted.iloc[:,[:8]]

# add clustering labels
TO_merged['Cluster Labels'] = kmeans.labels_

# merge toronto_grouped with toronto_data to add latitude/longitude for each neighborho
od
TO_merged = TO_merged.join(neighborhoods_venues_sorted.set_index('Neighborhood'), on='N
eighborhood')

TO_merged.head() # check the last columns!
```

```
(78, 7)
(78,)
```

```
<ipython-input-40-298ecff391b7>:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  TO_merged['Cluster Labels'] = kmeans.labels_
```

Out[40]:

	PostalCode	AfterTaxIncome2015	Population_2016	Bourough	Neighborhood	Latitude
0	M5M	111821.0	25975.0	North York	Bedford Park, Lawrence Manor East	43.733283
1	M5R	108271.0	26496.0	Central Toronto	The Annex, North Midtown, Yorkville	43.672710
2	M4G	94853.0	19076.0	East York	Leaside	43.709060
3	M1C	93943.0	35626.0	Scarborough	Rouge Hill, Port Union, Highland Creek	43.784535
4	M3B	90841.0	13324.0	North York	Don Mills	43.745906

In [41]:

```
TO_merged_new1 = TO_merged.loc[TO_merged['Cluster Labels'] == 0, TO_merged.columns[[3,
4] + list(range(5, TO_merged.shape[1]))]]
TO_merged_new1.shape
```

Out[41]:

```
(63, 15)
```

In [42]:

```
TO_merged_new2 = TO_merged.loc[TO_merged['Cluster Labels'] == 1, TO_merged.columns[[3,
4] + list(range(5, TO_merged.shape[1]))]]
TO_merged_new2.shape
```

Out[42]:

```
(2, 15)
```

3. Cluster 2 Contains the highest cluster density. We need to find the geographic centroid for this cluster. This is the optimum location for a new Restaurant Supply Store.

In [43]:

```
# Find the geographic center of the most dense or like cluster.
Cluster_0_coorid = TO_merged_new2[['Latitude', 'Longitude']]
Cluster_0_coorid = list(Cluster_0_coorid.values)
lat = []
long = []



for l in Cluster_0_coorid:
  lat.append(l[0])
  long.append(l[1])



Blatitude = sum(lat)/len(lat)
Blongitude = sum(long)/len(long)
print(Blatitude)
print(Blongitude)
```

```
43.71837895
-79.3379867
```

In [45]:

```
#3.1 Install opencage to reverse lookup the coordinates
# Intstall opencage to reverse lookup the cooridinates
!pip install opencage
from opencage.geocoder import OpenCageGeocode
from pprint import pprint

key = '935fec9601954fe8be52e5b979eef6b4'
geocoder = OpenCageGeocode(key)

results = geocoder.reverse_geocode(Blatitude, Blongitude)
pprint(results)
```

```
Requirement already satisfied: opencage in c:\anaconda\lib\site-packages
(1.2.2)
Requirement already satisfied: six>=1.4.0 in c:\anaconda\lib\site-packages
(from opencage) (1.15.0)
Requirement already satisfied: backoff>=1.10.0 in c:\anaconda\lib\site-pac
kages (from opencage) (1.10.0)
Requirement already satisfied: pyopenssl>=0.15.1 in c:\anaconda\lib\site-p
ackages (from opencage) (19.1.0)
Requirement already satisfied: Requests>=2.2.0 in c:\anaconda\lib\site-pac
kages (from opencage) (2.24.0)
Requirement already satisfied: cryptography>=2.8 in c:\anaconda\lib\site-p
ackages (from pyopenssl>=0.15.1->opencage) (2.9.2)
Requirement already satisfied: certifi>=2017.4.17 in c:\anaconda\lib\site-
packages (from Requests>=2.2.0->opencage) (2020.6.20)
Requirement already satisfied: chardet<4,>=3.0.2 in c:\anaconda\lib\site-p
ackages (from Requests>=2.2.0->opencage) (3.0.4)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
c:\anaconda\lib\site-packages (from Requests>=2.2.0->opencage) (1.25.9)
Requirement already satisfied: idna<3,>=2.5 in c:\anaconda\lib\site-packag
es (from Requests>=2.2.0->opencage) (2.10)
Requirement already satisfied: cffi!=1.11.3,>=1.8 in c:\anaconda\lib\site-
packages (from cryptography>=2.8->pyopenssl>=0.15.1->opencage) (1.14.0)
Requirement already satisfied: pycparser in c:\anaconda\lib\site-packages
(from cffi!=1.11.3,>=1.8->cryptography>=2.8->pyopenssl>=0.15.1->opencage)
(2.20)
[{'annotations': {'DMS': {'lat': "43° 43' 4.71000'' N",
                          'lng': "79° 20' 15.65520'' W"},
                  'MGRS': '17TPJ3390541892',
                  'Maidenhead': 'FN03hr92lh',
                  'Mercator': {'x': -8831830.361, 'y': 5392361.125},
                  'OSM': {'edit_url': 'https://www.openstreetmap.org/edit?
way=401009336#map=17/43.71798/-79.33768',
                          'note_url': 'https://www.openstreetmap.org/note/
new#map=17/43.71798/-79.33768&layers=N',
                          'url': 'https://www.openstreetmap.org/?mlat=43.7
1798&mlon=-79.33768#map=17/43.71798/-79.33768'},
                  'UN_M49': {'regions': {'AMERICAS': '019',
                                         'CA': '124',
                                         'NORTHERN_AMERICA': '021',
                                         'WORLD': '001'},
                             'statistical_groupings': ['MEDC']},
                  'callingcode': 1,
                  'currency': {'alternate_symbols': ['C$', 'CAD$'],
                               'decimal_mark': '.',
                               'disambiguate_symbol': 'C$',
                               'html_entity': '$',
                               'iso_code': 'CAD',
                               'iso_numeric': '124',
                               'name': 'Canadian Dollar',
                               'smallest_denomination': 5,
                               'subunit': 'Cent',
                               'subunit_to_unit': 100,
                               'symbol': '$',
                               'symbol_first': 1,
                               'thousands_separator': ','},
                  'flag': 'CA',
                  'geohash': 'dpz8dws5dj0wcr8v1gss',
                  'qibla': 54.61,
                  'roadinfo': {'drive_on': 'right',
                               'road': 'Rochefort Drive',
                               'speed_in': 'km/h'},
```

```
                  'sun': {'rise': {'apparent': 1605269460,
                                   'astronomical': 1605263520,
                                   'civil': 1605267600,
                                   'nautical': 1605265500},
                          'set': {'apparent': 1605304320,
                                  'astronomical': 1605310260,
                                  'civil': 1605306180,
                                  'nautical': 1605308280}},
                  'timezone': {'name': 'America/Toronto',
                               'now_in_dst': 0,
                               'offset_sec': -18000,
                               'offset_string': '-0500',
                               'short_name': 'EST'},
                  'what3words': {'words': 'from.twilight.limbs'}},
  'bounds': {'northeast': {'lat': 43.718025, 'lng': -79.337632},
             'southwest': {'lat': 43.717925, 'lng': -79.337732}},
  'components': {'ISO_3166-1_alpha-2': 'CA',
                 'ISO_3166-1_alpha-3': 'CAN',
                 '_category': 'building',
                 '_type': 'building',
                 'city': 'Toronto',
                 'city_district': 'North York',
                 'continent': 'North America',
                 'country': 'Canada',
                 'country_code': 'ca',
                 'house_number': '7',
                 'postcode': 'M3C 1C3',
                 'quarter': 'Don Valley East',
                 'road': 'Rochefort Drive',
                 'state': 'Ontario',
                 'state_code': 'ON',
                 'state_district': 'Golden Horseshoe'},
  'confidence': 10,
  'formatted': '7 Rochefort Drive, Toronto, ON M3C 1C3, Canada',
  'geometry': {'lat': 43.717975, 'lng': -79.337682}}]
```


In [46]:

```
#Obtain the popupstring of the best location
popstring = TO_data[TO_data['PostalCode'].str.contains('M4S')]

def str_join(*args):
    return ''.join(map(str, args))

popstring_new = str_join('The Best Neighbourhood to locate a Restaurant Supply Store is
in: ', popstring['Neighborhood'].values,  ' in ' ,  popstring['Bourough'].values)


print(popstring_new)
```

```
The Best Neighbourhood to locate a Restaurant Supply Store is in: ['Davisv
ille'] in ['Central Toronto']
```

In [54]:

```
# Let's get the coordinates for Toronto
import geopy
from geopy.geocoders import Nominatim
address = 'Toronto, ON'

geolocator = Nominatim(user_agent='abhishekkumaragrawal1819@gmail.com')
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude
print('The geograpical coordinate of Toronto are {}, {}.'.format(latitude, longitude))
```

```
The geograpical coordinate of Toronto are 43.6534817, -79.3839347.
```

4. Results

4.1 Plot the clusters on a Map of the Toronto and Super Impose the best location of a Store

In [55]:

```
# getfolium
import folium
# Matplotlib and associated plotting modules
import matplotlib.cm as cm
import matplotlib.colors as colors

# create map
map_clusters = folium.Map(location=[latitude, longitude], zoom_start=11)

# set color scheme for the clusters
x = np.arange(kclusters)
ys = [i+x+(i*x)**2 for i in range(kclusters)]
colors_array = cm.rainbow(np.linspace(0, 1, len(ys)))
rainbow = [colors.rgb2hex(i) for i in colors_array]

# add markers to the map
markers_colors = []
for lat, lon, poi, cluster in zip(TO_merged['Latitude'], TO_merged['Longitude'], TO_mer
ged['Neighborhood'], TO_merged['Cluster Labels']):
    label = folium.Popup(str(poi) + ' Cluster ' + str(cluster), parse_html=True)
    folium.CircleMarker(
        [lat, lon],
        radius=5,
        popup=label,
        color=rainbow[cluster-1],
        fill=True,
        fill_color=rainbow[cluster-1],
        fill_opacity=0.7).add_to(map_clusters)

folium.CircleMarker([Blatitude, Blongitude],
                    radius=50,
                    popup='Toronto',
                    color='red',
                    ).add_to(map_clusters)

# Interactive marker
map_clusters.add_child(folium.ClickForMarker(popup=popstring_new))

#map_clusters
map_clusters.save('map_clusters.html')
```

In [57]:

```
map_clusters
```

Out[57]:

Make this Notebook Trusted to load map: File -> Trust Notebook

In [58]:

```
#4.2 Exact Address of desired Location
print('The exact Address to locate would be: 268 Balliol Street, ON M4S 1C2, Canada or
 lat: 43.6991598, lng: -79.3878871')
```

```
The exact Address to locate would be: 268 Balliol Street, ON M4S 1C2, Cana
da or lat: 43.6991598, lng: -79.3878871
```

Discussion:

5.1 Explaining the results

As we built our list of neighborhoods with Restaurant venues exclusively we discovered most neighborhoods were similar and the greatest concentration of restaurants was in Central Toronto and downtown Toronto. This might seem obvious but it would also appear that these are some of the most affluent neighborhoods in Toronto so there appears to be correlation. By Locating in the general vicinity of the Exact location my friend could be geographically centered in this cluster and poised to service his restaurant customer base with the greatest efficiency.

When we built our our K-Means dataset we used Silhouette analysis to tell us there was a lot of similarity between neighborhoods and the most common restaurants contained with in. Really there was only 2 types of cluster or neighborhoods in greater Toronto. The vast majority of those were in 1 cluster. So Toronto restaurants might be many but they are very homogeneously located near the center of Toronto.

Of the 103 Toronto Neighborhoods gathered only 55.3% or 57 Neighborhoods are above the median after-tax income. 37.8% or 39 Neighborhoods are below he median after-tax income. 6.7% or 7 neighborhoods did not register as it appears their populations are too low. It appears that the greatest concentration of affluence is near central Toronto. We decided to keep all neighborhoods in the dataset regardless of income of population as the majority were close enough.

Conclusion:

I feel confident with the recommendation I have given my friend as it is backed up with demonstrated data analysis. While nothing can ever be 100% certain he will certainly be better informed than he was prior to asking for my help.

Much more inference can be obtained with more work. A potential side business for my friend might be assisting new restaurant owners where they might locate a new restaurant, who their competition is and who their clientele might be.

In [ ]: