# intalling beaitifulsoup library

!conda install -c conda-forge beautifulsoup4 --yes

!conda install -c conda-forge lxml --yes

!conda install -c conda-forge requests --yes

!conda install -c conda-forge geocoder --yes

!conda install -c conda-forge geopy --yes

!conda install -c conda-forge folium=0.5.0 –yes

---------------# importing necessary modules

import requests

import pandas as pd

import numpy as np

import lxml

import matplotlib.cm as cm

import matplotlib.colors as colors

import folium

# libraries for displaying images

from IPython.display import Image

from IPython.core.display import HTML

from sklearn.cluster import KMeans

from bs4 import BeautifulSoup

from pandas.io.json import json\_normalize

from geopy.geocoders import Nominatim

# url for the wikipedia page

url = 'https://en.wikipedia.org/wiki/List\_of\_postal\_codes\_of\_Canada:\_M'

source = requests.get(url).text

# cretaing a beautifulsoup obect

soup = BeautifulSoup(source,'lxml')

# creating & loading data into dataframe

pcode\_df = pd.DataFrame()

i = 0

j = 0

table\_pcode = soup.find('table')

for table\_data in table\_pcode.find\_all('td'):

col\_data = table\_data.text

pcode\_df.loc[i,j] = col\_data

j = j + 1

if (j==3):

i = i + 1

j = 0

# adding column names to the data frame

pcode\_df.columns = ('PostalCode','Borough','Neighborhood')

# removing the \n haracter from the end of the string

pcode\_df['Neighborhood'] = pcode\_df['Neighborhood'].str.slice(stop = -1)

# removing the rows with borough value as not assigned

pcode\_df=pcode\_df[pcode\_df.Borough != 'Not assigned']

pcode\_df = pcode\_df.reset\_index(drop=True)

# concatenating neighborhoods with same Postal Code

new\_df = pd.DataFrame()

new\_df = pcode\_df.groupby('PostalCode').agg({'Neighborhood':lambda x: tuple(x)})

new\_df.reset\_index(inplace=True)

# converting the tuple into a string data type

new\_df['Neighborhood'] = new\_df['Neighborhood'].astype(str)

#removing the end and begenning braces and the single quotes

new\_df['Neighborhood'] = new\_df['Neighborhood'].str.slice(stop = -2)

new\_df['Neighborhood'] = new\_df['Neighborhood'].str.slice(start = 2)

new\_df['Neighborhood'] = new\_df['Neighborhood'].str.replace("'","")

# creating a data frame of unique postal code and borough combination

df\_borough = pcode\_df

df\_borough = df\_borough.drop('Neighborhood',axis=1)

df\_borough.drop\_duplicates(inplace=True)

# creation of the final data frame by combining two intermediate data frames using Postal code as key

final\_pcode\_df = pd.DataFrame()

final\_pcode\_df = df\_borough.merge(new\_df, on='PostalCode')

# updating the data for rows missing Neighborhood values

loc\_Neigh\_NA = final\_pcode\_df.Neighborhood == 'Not assigned'

final\_pcode\_df.loc[loc\_Neigh\_NA,'Neighborhood'] = final\_pcode\_df.loc[loc\_Neigh\_NA,'Borough']

final\_pcode\_df.head()

# shape of the final dataframe

print('Shape of the Final DataFrame',final\_pcode\_df.shape)

# reading the csv file with latitude and longitude value

lat\_long\_df = pd.read\_csv('http://cocl.us/Geospatial\_data')

# modify column names inorder to perform the merge

lat\_long\_df.columns = ('PostalCode','Latitude','Longitude')

# merging data to create the new data set with latitude and longitude data

final\_lat\_long\_df = final\_pcode\_df.merge(lat\_long\_df , on='PostalCode')

final\_lat\_long\_df.head(2)

final\_lat\_long\_df.shape

# @hidden\_cell

# Initializing Foursquare Credentials

CLIENT\_ID = 'Q3OILHOPS1VYPYSLHI4OQY4LFZ3KC04J2UTW3ON3JCOXXQFM'

CLIENT\_SECRET = 'K5S3ABHIJCCNZY5HMHJS1ZHYQUGYQGZ3C0YO1HRVGNVEKYE0'

VERSION = '20180605'

# creating a function to retrive venue information from Forsquare

def getNearbyVenues(names, latitudes, longitudes,search\_query='restaurant',radius=500):

neigh\_venues\_list=pd.DataFrame(columns=('Neighborhood','Neigh\_Latitude','Neigh\_Longitude','ID','Venue','Venue\_Latitude', \

'Venue\_Longitude','Venue\_Category'))

j = 0

if (neigh\_venues\_list.shape[0] == 0):

i = 0

else:

i = neigh\_venues\_list.shape[0]

for name, lat, lng in zip(names, latitudes, longitudes):

# create the API request URL

url ='https://api.foursquare.com/v2/venues/search?client\_id={}&client\_secret={}&ll={},{}&v={}&query={}&radius={}&limit={}'.format(

CLIENT\_ID,

CLIENT\_SECRET,

lat,

lng,

VERSION,

search\_query,

radius,

50)

# make the GET request

results = requests.get(url).json()["response"]['venues']

# store the relevant information for each nearby venue in adate frame

for v in results:

try:

neigh\_venues\_list.loc[i,'ID'] = v['id']

neigh\_venues\_list.loc[i,'Venue'] = v['name']

neigh\_venues\_list.loc[i,'Venue\_Latitude'] = v['location']['lat']

neigh\_venues\_list.loc[i,'Venue\_Longitude'] = v['location']['lng']

neigh\_venues\_list.loc[i,'Venue\_Category'] = v['categories'][0]['name']

except:

neigh\_venues\_list.loc[i,'ID'] = np.nan

neigh\_venues\_list.loc[i,'Venue'] = np.nan

neigh\_venues\_list.loc[i,'Venue\_Latitude'] = np.nan

neigh\_venues\_list.loc[i,'Venue\_Longitude'] = np.nan

neigh\_venues\_list.loc[i,'Venue\_Category'] = np.nan

neigh\_venues\_list.loc[i,'Neighborhood'] = names[j]

neigh\_venues\_list.loc[i,'Neigh\_Latitude'] = latitudes[j]

neigh\_venues\_list.loc[i,'Neigh\_Longitude'] = longitudes[j]

i = i + 1

j = j + 1

# return the data frame

return(neigh\_venues\_list)

# creating and storing the venues information returned by the user defined funtion

toronto\_venues = pd.DataFrame()

toronto\_venues = getNearbyVenues(names=final\_lat\_long\_df['Neighborhood'],

latitudes=final\_lat\_long\_df['Latitude'],

longitudes=final\_lat\_long\_df['Longitude']

)

toronto\_venues.head(2)

toronto\_venues.dropna(axis=0,how='any',inplace=True)

toronto\_venues.reset\_index(drop=True,inplace=True)

print(toronto\_venues.shape)

toronto\_venues.head()

# list out the unique restaurant categories

toronto\_venues['Venue\_Category'].unique()

toronto\_venues[toronto\_venues['Venue\_Category'] == 'Food'].head(15)

toronto\_venues[toronto\_venues['Venue\_Category'] == 'Hakka Restaurant'].head(15)

toronto\_venues[toronto\_venues['Venue\_Category'] == 'Asian'].head(15)

toronto\_venues[toronto\_venues['Venue\_Category'] == 'Restaurant'].head(15)

toronto\_venues[toronto\_venues['Venue\_Category'] == 'Noodle House'].head(15)

toronto\_venues['Venue\_Category'] = toronto\_venues['Venue\_Category'].str.replace(" Restaurant","")

toronto\_venues['Venue\_Category'].unique()

for i in range(0,toronto\_venues.shape[0]):

if (toronto\_venues.loc[i,'Venue\_Category'] in ('Chinese','Korean','Thai','Asian','Japanese','Sushi', \

'Dim Sum','Hakka','Vietnamese','Mongolian','Szechuan', \

'Cantonese','Dumpling','Noodle House')):

toronto\_venues.loc[i,'Venue\_Cat\_Group'] = 'East Asian'

elif (toronto\_venues.loc[i,'Venue\_Category'] in ('Indian')):

toronto\_venues.loc[i,'Venue\_Cat\_Group'] = 'South Asian'

elif (toronto\_venues.loc[i,'Venue\_Category'] in ('Ethiopian','African',)):

toronto\_venues.loc[i,'Venue\_Cat\_Group'] = 'African'

elif (toronto\_venues.loc[i,'Venue\_Category'] in ('American','New American','Steakhouse','Diner')):

toronto\_venues.loc[i,'Venue\_Cat\_Group'] = 'North American'

elif (toronto\_venues.loc[i,'Venue\_Category'] in ('Mexican','Latin American')):

toronto\_venues.loc[i,'Venue\_Cat\_Group'] = 'Latin American'

elif (toronto\_venues.loc[i,'Venue\_Category'] in ('Bar','Nightclub','Wine Bar','Beer Bar','Gastropub', \

'Pub','Brewery','Nightlife Spot','Lounge')):

toronto\_venues.loc[i,'Venue\_Cat\_Group'] = 'Beverage Bar'

elif (toronto\_venues.loc[i,'Venue\_Category'] in ('Italian','Pizza Place','Spanish','Tapas', \

'French','Portuguese','Modern European','Eastern European')):

toronto\_venues.loc[i,'Venue\_Cat\_Group'] = 'European'

elif (toronto\_venues.loc[i,'Venue\_Category'] in ('Middle Eastern','Greek','Mediterranean')):

toronto\_venues.loc[i,'Venue\_Cat\_Group'] = 'Mediterranean'

elif (toronto\_venues.loc[i,'Venue\_Category'] in ('Caribbean','Cuban')):

toronto\_venues.loc[i,'Venue\_Cat\_Group'] = 'Caribbean'

else:

toronto\_venues.loc[i,'Venue\_Cat\_Group'] = (toronto\_venues.loc[i,'Venue\_Category'])

toronto\_venues['Venue\_Cat\_Group'].unique()

# delet rows taht eitehr are not true culinary establishments e.g. Deli, Movie Tehater

#or do not have any specific categorization e.g. - Food, Restaurant

delet\_cat\_group = ('Restaurant','General Entertainment','Hotel','Food','Deli / Bodega', \

'Movie Theater','Music Venue','Furniture / Home Store')

toronto\_venues = toronto\_venues[~toronto\_venues.Venue\_Cat\_Group.isin(delet\_cat\_group)]

toronto\_venues.reset\_index(drop=True,inplace=True)

toronto\_venues['Venue\_Cat\_Group'].unique()

toronto\_venues.shape

toronto\_venues.columns

toronto\_venues\_no\_dup = pd.DataFrame

toronto\_venues\_no\_dup = toronto\_venues.copy()

toronto\_venues\_no\_dup.drop(['Neighborhood','Neigh\_Latitude','Neigh\_Longitude'],axis=1,inplace=True)

toronto\_venues\_no\_dup.drop\_duplicates(inplace=True)

toronto\_venues\_no\_dup.reset\_index(drop=True,inplace=True)

print(toronto\_venues\_no\_dup.shape)

toronto\_venues\_no\_dup.head(2)

toronto\_venues\_no\_dup['Venue\_Rating'] = np.nan

toronto\_venues\_no\_dup.columns

toronto\_venues\_no\_dup.shape

toronto\_venues\_1 = toronto\_venues\_no\_dup.copy()

toronto\_venues\_1.head(2)

toronto\_venues\_1.shape

def getVenueRatings(venue\_id):

url = 'https://api.foursquare.com/v2/venues/{}?client\_id={}&client\_secret={}&v={}'.format(

venue\_id, CLIENT\_ID, CLIENT\_SECRET, VERSION)

results = requests.get(url).json()

try:

venue\_rating = results['response']['venue']['rating']

except:

venue\_rating = 0

return(venue\_rating)

#good code

for i in range(0,toronto\_venues\_1.shape[0]):

toronto\_venues\_1.loc[i,'Venue\_Rating'] = getVenueRatings(toronto\_venues\_1.loc[i,'ID'])

toronto\_venues\_1.head(3)

toronto\_venues\_1['Venue\_Rating'].unique()

toront\_venue\_1\_rate\_nonzero = pd.DataFrame()

toront\_venue\_1\_rate\_nonzero = toronto\_venues\_1[toronto\_venues\_1['Venue\_Rating'] != 0]

toront\_venue\_1\_rate\_mean = toront\_venue\_1\_rate\_nonzero.groupby('Venue\_Cat\_Group').agg({'Venue':'count','Venue\_Rating':'sum'})

toront\_venue\_1\_rate\_mean.sort\_values('Venue\_Rating',ascending =False,inplace=True)

toront\_venue\_1\_rate\_mean.head()

toront\_venue\_1\_rate\_mean['Avg\_Rating'] = (toront\_venue\_1\_rate\_mean['Venue\_Rating'] \* \

toront\_venue\_1\_rate\_mean['Venue']) / sum(toront\_venue\_1\_rate\_mean['Venue'])

toront\_venue\_1\_rate\_mean.sort\_values('Avg\_Rating',ascending=False, inplace=True)

toront\_venue\_1\_rate\_mean.head()

toront\_venue\_1\_rate\_mean.reset\_index(inplace=True)

toront\_venue\_1\_rate\_mean.head()

toronto\_neigh\_cat = pd.DataFrame()

toronto\_neigh\_cat = toronto\_venues.drop(['Neigh\_Latitude','Neigh\_Longitude','ID', \

'Venue', 'Venue\_Latitude', 'Venue\_Longitude', \

'Venue\_Category'],axis=1).copy()

toronto\_neigh\_cat\_2 = pd.DataFrame()

toronto\_neigh\_cat\_2 = toronto\_neigh\_cat.drop\_duplicates().copy()

toronto\_neigh\_cat\_2.reset\_index(inplace=True)

toronto\_neigh\_cat\_2.drop('index',axis=1,inplace = True)

toronto\_neigh\_cat\_2.head()

toronto\_neigh\_cat\_3 = toronto\_neigh\_cat\_2.groupby('Neighborhood')['Venue\_Cat\_Group'].apply(tuple)

toronto\_neigh\_cat\_3 = pd.DataFrame(toronto\_neigh\_cat\_3)

toronto\_neigh\_cat\_3.reset\_index(inplace=True)

toronto\_neigh\_cat\_3['Venue\_Cat\_Group'] = toronto\_neigh\_cat\_3['Venue\_Cat\_Group'].astype(str)

toronto\_neigh\_cat\_3['Venue\_Cat\_Group'] = toronto\_neigh\_cat\_3['Venue\_Cat\_Group'].str.slice(stop = -2)

toronto\_neigh\_cat\_3['Venue\_Cat\_Group'] = toronto\_neigh\_cat\_3['Venue\_Cat\_Group'].str.slice(start = 2)

toronto\_neigh\_cat\_3['Venue\_Cat\_Group'] = toronto\_neigh\_cat\_3['Venue\_Cat\_Group'].str.replace("'","")

toronto\_neigh\_cat\_3.head()

def getNextRestaurant(toronto\_neigh\_cat\_3):

next\_rest = pd.DataFrame(columns=('Neighborhood','Next\_Rest\_Cat\_Group'))

i = 0

for i in range(0,toronto\_neigh\_cat\_3.shape[0]):

next\_rest.loc[i,'Neighborhood'] = toronto\_neigh\_cat\_3.loc[i,'Neighborhood']

j = 0

for j in range(0,toront\_venue\_1\_rate\_mean.shape[0]):

if (toront\_venue\_1\_rate\_mean.loc[j,'Venue\_Cat\_Group'] in toronto\_neigh\_cat\_3.loc[i,'Venue\_Cat\_Group']):

j = j + 1

else:

next\_rest.loc[i,'Next\_Rest\_Cat\_Group'] = toront\_venue\_1\_rate\_mean.loc[j,'Venue\_Cat\_Group']

break

i = i + 1

return(next\_rest)

next\_rest = getNextRestaurant(toronto\_neigh\_cat\_3)

next\_rest.head(20)

next\_rest.shape

next\_loc = next\_rest.groupby('Next\_Rest\_Cat\_Group')['Neighborhood'].apply(tuple)

next\_loc = pd.DataFrame(next\_loc)

next\_loc.reset\_index(inplace=True)

next\_loc['Neighborhood'] = next\_loc['Neighborhood'].astype(str)

next\_loc['Neighborhood'] = next\_loc['Neighborhood'].str.slice(stop = -2)

next\_loc['Neighborhood'] = next\_loc['Neighborhood'].str.slice(start = 2)

next\_loc['Neighborhood'] = next\_loc['Neighborhood'].str.replace("'","")

next\_loc.head(25)