Segmenting and Clustering

Neighborhoods in Toronto

Peer-graded Assignment: Github_Segmenting and Clustering Neighborhoods in Toronto_Linda

Table of Contents

Question 1:

1.1. Notebook book created

with the basic dependencies.

In [1]:
```
import numpy as np # library to handle data in a vectorized manner
import pandas as pd # library for data analsysis
import requests # Library for web scraping

print('Libraries imported.')
```

```
Libraries imported.
```

1.2. Web page scraped

About the Data, Wikipedia page, https://en.wikipedia.org/wiki/List_of_postal_codes_of_Canada:_M (https://en.wikipedia.org/wiki/List_of_postal_codes_of_Canada:_M),

- is a list of postal codes in Canada where the first letter is M. Postal codes beginning with M are located within the city of Toronto in the province of Ontario.
- Scraping table from HTML using BeautifulSoup, write a Python program similar to scrape.py,from:

Corey Schafer Python Programming Tutorial:

The code from this video can be found at: https://github.com/CoreyMSchafer/code (https://github.com/CoreyMSchafer/code)...

In [2]:

```
# To run this, you can install BeautifulSoup
# https://pypi.python.org/pypi/beautifulsoup4

# Or download the file
# http://beautiful-soup-4
# and unzip it in the same directory as this file
import requests
from urllib.request import urlopen
from bs4 import BeautifulSoup
import ssl
import csv

print('BeautifulSoup  & csv imported.')
```

```
BeautifulSoup  & csv imported.
```

In [3]:

```
# Ignore SSL certificate errors
ctx = ssl.create_default_context()
ctx.check_hostname = False
ctx.verify_mode = ssl.CERT_NONE

print('SSL certificate errors ignored.')
```

```
SSL certificate errors ignored.
```

In [4]:

```
source = requests.get('https://en.wikipedia.org/wiki/List_of_postal_codes_of_Canada:_M').text

soup = BeautifulSoup(source, 'lxml')

#print(soup.prettify())
print('soup ready')
```

```
soup ready
```

In [5]:

```
table = soup.find('table',{'class':'wikitable sortable'})
#table
```

In [6]:

```
table_rows = table.find_all('tr')

#table_rows
```

In [7]:

```
data = []
for row in table_rows:
    data.append([t.text.strip() for t in row.find_all('td')])

df = pd.DataFrame(data, columns=['PostalCode', 'Borough', 'Neighbourhood'])
df = df[~df['PostalCode'].isnull()]  # to filter out bad rows

#print(df.head(5))
#print('***')
#print(df.tail(5))
```

1.3. Data transformed into pandas dataframe

In [8]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 288 entries, 1 to 288
Data columns (total 3 columns):
PostalCode       288 non-null object
Borough          288 non-null object
Neighbourhood    288 non-null object
dtypes: object(3)
memory usage: 9.0+ KB
```

In [9]:

```
df.shape
```

Out[9]:

```
(288, 3)
```

1.4. Dataframe cleaned and notebook annotate

Only process the cells that have an assigned borough, we can ignore cells with 'Not assigned' boroughs, like in rows 1 & 2.

In [10]:

```
import pandas
import requests
from bs4 import BeautifulSoup
website_text = requests.get('https://en.wikipedia.org/wiki/List_of_postal_codes_of_Canada:_M').text
soup = BeautifulSoup(website_text,'lxml')

table = soup.find('table',{'class':'wikitable sortable'})
table_rows = table.find_all('tr')

data = []
for row in table_rows:
    data.append([t.text.strip() for t in row.find_all('td')])

df = pandas.DataFrame(data, columns=['PostalCode', 'Borough', 'Neighbourhood'])
df = df[~df['PostalCode'].isnull()]  # to filter out bad rows

#df.head(15)
```

In [11]:

```
df.drop(df[df['Borough']=="Not assigned"].index,axis=0, inplace=True)
#df.head()
```

The dataframe can be reindex as follows:

In [12]:

```
df1 = df.reset_index()
#df1.head()
```

In [13]:

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 211 entries, 0 to 210
Data columns (total 4 columns):
index            211 non-null int64
PostalCode       211 non-null object
Borough          211 non-null object
Neighbourhood    211 non-null object
dtypes: int64(1), object(3)
memory usage: 6.7+ KB
```

In [14]:

```
df1.shape
```

Out[14]:

```
(211, 4)
```

More than one neighborhood can exist in one postal code area, 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 using groupby, see:

https://pandas-docs.github.io/pandas-docs-travis/user_guide/groupby.html (https://pandas-docs.github.io/pandas-docs-travis/user_guide/groupby.html)

```
In [15]: df2= df1.groupby('PostalCode').agg(lambda x: ','.join(x))

         #df2.head()
```

```
In [16]: df2.info()

         <class 'pandas.core.frame.DataFrame'>
         Index: 103 entries, M1B to M9W
         Data columns (total 2 columns):
         Borough          103 non-null object
         Neighbourhood    103 non-null object
         dtypes: object(2)
         memory usage: 2.4+ KB
```

```
In [17]: df2.shape

Out[17]: (103, 2)
```

There are also cells that have an assigned neighbouhoods,like M7A, lets assign their boroughs as their neighbourhood, as follows:

```
In [18]: df2.loc[df2['Neighbourhood']=="Not assigned",'Neighbourhood']=df2.loc[
         df2['Neighbourhood']=="Not assigned",'Borough']

         #df2.head()
```

```
In [19]: df3 = df2.reset_index()
         #df3.head()
```

Now we can remove the duplicate boroughts as follows:

```
In [20]: df3['Borough']= df3['Borough'].str.replace('nan|[{}\s]','').str.split(
         ',').apply(set).str.join(',').str.strip(',').str.replace(",{2,}",",")
```

In [21]:
```
df3.head()
```

Out[21]:

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

In [22]:
```
df3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 103 entries, 0 to 102
Data columns (total 3 columns):
PostalCode       103 non-null object
Borough          103 non-null object
Neighbourhood    103 non-null object
dtypes: object(3)
memory usage: 2.5+ KB
```

In [23]:
```
df3.shape
```

Out[23]:
```
(103, 3)
```

1.5. Q1_notebook on Github repository. (10 marks)

Question 2:

2.1. Used the Geocoder Package

In [24]:
```
pip install geopy
```

```
Requirement already satisfied: geopy in /home/jupyterlab/conda/lib/p
ython3.6/site-packages (1.11.0)
Note: you may need to restart the kernel to use updated packages.
```

In [25]:

```
from  geopy.geocoders import Nominatim
geolocator = Nominatim()
city ="London"
country ="Uk"
loc = geolocator.geocode(city+','+ country)
print("latitude is :-" ,loc.latitude,"\nlongtitude is:-" ,loc.longitud
e)
```

```
latitude is :- 51.5073219
longtitude is:- -0.1276474
```

In [26]:

```
from  geopy.geocoders import Nominatim
geolocator = Nominatim()
location = geolocator.geocode("Toronto, North York, Parkwoods")

print(location.address)
print('')
print((location.latitude, location.longitude))
print('')
print(location.raw)
```

```
Parkwoods Village Drive, Parkway East, Don Valley East, North York,
Toronto, Ontario, M3A 1Z5, Canada

(43.7611243, -79.3240594)

{'place_id': 112261812, 'licence': 'Data © OpenStreetMap contributor
s, ODbL 1.0. https://osm.org/copyright', 'osm_type': 'way', 'osm_id'
: 160406962, 'boundingbox': ['43.761106', '43.7612191', '-79.3242996
', '-79.3239088'], 'lat': '43.7611243', 'lon': '-79.3240594', 'displ
ay_name': 'Parkwoods Village Drive, Parkway East, Don Valley East, N
orth York, Toronto, Ontario, M3A 1Z5, Canada', 'class': 'highway', '
type': 'secondary', 'importance': 0.51}
```

In [27]:

```
import pandas as pd
#df3.head()
```

In [28]:

```
import pandas as pd
df_geopy = pd.DataFrame({'PostalCode': ['M3A', 'M4A', 'M5A'],
                         'Borough': ['North York', 'North York', 'Down
town Toronto'],
                         'Neighbourhood': ['Parkwoods', 'Victoria Vill
age', 'Harbourfront'],})

from geopy.geocoders import Nominatim
geolocator = Nominatim()
```

In [29]:
```
df_geopy1 = df3
#df_geopy1
```

In [30]:
```
from geopy.geocoders import Nominatim
geolocator = Nominatim()

df_geopy1['address'] = df3[['PostalCode', 'Borough', 'Neighbourhood']]
.apply(lambda x: ', '.join(x), axis=1 )
df_geopy1.head()
```

Out[30]:

	PostalCode	Borough	Neighbourhood	address
0	M1B	Scarborough	Rouge,Malvern	M1B, Scarborough, Rouge,Malvern
1	M1C	Scarborough	Highland Creek,Rouge Hill,Port Union	M1C, Scarborough, Highland Creek,Rouge Hill,Po...
2	M1E	Scarborough	Guildwood,Morningside,West Hill	M1E, Scarborough, Guildwood,Morningside,West Hill
3	M1G	Scarborough	Woburn	M1G, Scarborough, Woburn
4	M1H	Scarborough	Cedarbrae	M1H, Scarborough, Cedarbrae

In [31]:
```
df_geopy1 = df3
```

In [32]:
```
df_geopy1.shape
```

Out[32]:
```
(103, 4)
```

In [33]:
```
df_geopy1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 103 entries, 0 to 102
Data columns (total 4 columns):
PostalCode       103 non-null object
Borough          103 non-null object
Neighbourhood    103 non-null object
address          103 non-null object
dtypes: object(4)
memory usage: 3.3+ KB
```

In [34]:

```
df_geopy1.drop(df_geopy1[df_geopy1['Borough']=="Notassigned"].index,ax
is=0, inplace=True)
#df_geopy1
# code holds true up until i=102
df_geopy1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 103 entries, 0 to 102
Data columns (total 4 columns):
PostalCode       103 non-null object
Borough          103 non-null object
Neighbourhood    103 non-null object
address          103 non-null object
dtypes: object(4)
memory usage: 4.0+ KB
```

In [35]:

```
#df_geopy1.head()
```

In [36]:

```
df_geopy1.shape
```

Out[36]:

```
(103, 4)
```

In [37]:

```
df_geopy1.to_csv('geopy1.csv')
# no data for location after row 75
```

Now let's test for location = 'M1G, Scarborough, Woburn'

In [38]:

```
from  geopy.geocoders import Nominatim
geolocator = Nominatim()
location = geolocator.geocode("M1G, Scarborough, Woburn")


#print(location.address)

#print((location.latitude, location.longitude))

#print(location.raw)
```

```
In [39]: pip install geocoder
```

```
Collecting geocoder
  Downloading https://files.pythonhosted.org/packages/4f/6b/13166c90
9ad2f2d76b929a4227c952630ebaf0d729f6317eb09cbceccbab/geocoder-1.38.1
-py2.py3-none-any.whl (98kB)
    100% |████████████████████████████████| 102kB 17.7MB/s
Requirement already satisfied: click in /home/jupyterlab/conda/lib/p
ython3.6/site-packages (from geocoder) (7.0)
Requirement already satisfied: requests in /home/jupyterlab/conda/li
b/python3.6/site-packages (from geocoder) (2.21.0)
Collecting ratelim (from geocoder)
  Downloading https://files.pythonhosted.org/packages/f2/98/7e6d147f
d16a10a5f821db6e25f192265d6ecca3d82957a4fdd592cad49c/ratelim-0.1.6-p
y2.py3-none-any.whl
Requirement already satisfied: future in /home/jupyterlab/conda/lib/
python3.6/site-packages (from geocoder) (0.17.1)
Requirement already satisfied: six in /home/jupyterlab/conda/lib/pyt
hon3.6/site-packages (from geocoder) (1.12.0)
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in /home/jupyte
rlab/conda/lib/python3.6/site-packages (from requests->geocoder) (3.
0.4)
Requirement already satisfied: certifi>=2017.4.17 in /home/jupyterla
b/conda/lib/python3.6/site-packages (from requests->geocoder) (2019.
3.9)
Requirement already satisfied: urllib3<1.25,>=1.21.1 in /home/jupyte
rlab/conda/lib/python3.6/site-packages (from requests->geocoder) (1.
24.1)
Requirement already satisfied: idna<2.9,>=2.5 in /home/jupyterlab/co
nda/lib/python3.6/site-packages (from requests->geocoder) (2.8)
Requirement already satisfied: decorator in /home/jupyterlab/conda/l
ib/python3.6/site-packages (from ratelim->geocoder) (4.4.0)
Installing collected packages: ratelim, geocoder
Successfully installed geocoder-1.38.1 ratelim-0.1.6
Note: you may need to restart the kernel to use updated packages.
```

Bonus _ Used Geopy & OpenStreetMap to create Dataframe

```
In [40]: df3.to_csv('geopy.csv')
```

In [41]:

```
import csv

with open('geopy.csv') as csvfile:
     reader = csv.DictReader(csvfile)
     #for row in reader:
         #print(row['PostalCode'],row['Borough'], row['Neighbourhood']
)
```

In [42]:

```
from  geopy.geocoders import Nominatim
geolocator = Nominatim()
location = geolocator.geocode("M1B Scarborough Rouge,Malvern")

#print(location.address)

#print((location.latitude, location.longitude))

#print(location.raw)
```

In [43]:

```
from  geopy.geocoders import Nominatim
geolocator = Nominatim()
location = geolocator.geocode("Toronto, Highland Creek")

#print(location.address)

#print((location.latitude, location.longitude))

#print(location.raw)

#M1C Scarborough Highland Creek,Rouge Hill,Port Union = no address
```

In [44]:

```
from  geopy.geocoders import Nominatim
geolocator = Nominatim()
location = geolocator.geocode("Toronto, Morningside")

#print(location.address)

#print((location.latitude, location.longitude))

#print(location.raw)

#M1E Scarborough Guildwood,Morningside,West Hill = no address
```

Bonus how to create a csv file.

In [45]:

```
import numpy as np
import csv



PostalCode = None
Borough = None
Neighbourhood = None
latData = None
longData = None

LAT_Woburn = 43.7598243
LONG_Woburn = -79.2252908
LAT_Malvern = 43.8091955
LONG_Malvern = -79.2217008
LAT_Highland_Creek = 43.7901172
LONG_Highland_Creek = -79.1733344
LAT_Morningside = 43.7826012
LONG_Morningside = -79.2049579



PostalCode = np.array(['M1H','M1B','M1C','M1G '])
Borough = np.array(['Scarborough','Scarborough','Scarborough','Scarborough'])
Neighbourhood = np.array(['Woburn','Malvern','Highland_Creek','Morningside'])
latData = np.array([43.7598243,43.8091955, 43.7901172 , 43.7826012])
longData = np.array([-79.2252908,-79.2217008,-79.1733344, -79.2049579])

with open('data.csv', 'w') as file:
    writer = csv.writer(file, delimiter=',')
    writer.writerow('ABXYZ')
    for a,b,x,y,z in np.nditer([ PostalCode.T, Borough.T, Neighbourhood.T, latData.T, longData.T], order='C'):
        writer.writerow([a,b,x,y,z])
```

In [46]:

```
import csv

with open('data.csv') as csvfile:
     reader = csv.DictReader(csvfile)
     for row in reader:
         print(row['A'], row['B'],row['X'], row['Y'], row['Z'])
```

```
M1H Scarborough Woburn 43.7598243 -79.2252908
M1B Scarborough Malvern 43.8091955 -79.2217008
M1C Scarborough Highland_Creek 43.7901172 -79.1733344
M1G  Scarborough Morningside 43.7826012 -79.2049579
```

In [47]:

```
pd.read_csv('data.csv')
```

Out[47]:

	A	B	X	Y	Z
0	M1H	Scarborough	Woburn	43.759824	-79.225291
1	M1B	Scarborough	Malvern	43.809196	-79.221701
2	M1C	Scarborough	Highland_Creek	43.790117	-79.173334
3	M1G	Scarborough	Morningside	43.782601	-79.204958

Retrieved coordinates with lambda equation

In [48]:

```
import pandas, os
#os.listdir()
```

In [49]:

```
df_geopy=df3
#df_geopy.head()
```

In [50]:

```
import geopy
#dir(geopy)
```

In [51]:

```
type(df_geopy)
```

Out[51]:

```
pandas.core.frame.DataFrame
```

In [52]:

```
df_geopy.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 103 entries, 0 to 102
Data columns (total 4 columns):
PostalCode       103 non-null object
Borough          103 non-null object
Neighbourhood    103 non-null object
address          103 non-null object
dtypes: object(4)
memory usage: 4.0+ KB
```

Import GeoPy:

In [53]:

```
pip install geopy
```

```
Requirement already satisfied: geopy in /home/jupyterlab/conda/lib/p
ython3.6/site-packages (1.11.0)
Note: you may need to restart the kernel to use updated packages.
```

In [54]:

```
from geopy.geocoders import Nominatim
print('Nominatim imported')
```

```
Nominatim imported
```

Set connection to OpenStreeMap

In [55]:

```
df_geopy['address']=df_geopy['PostalCode'] + ',' + df_geopy['Borough']
+ ','+ df_geopy['Neighbourhood']
df_geopy.head()
```

Out[55]:

	PostalCode	Borough	Neighbourhood	
0	M1B	Scarborough	Rouge,Malvern	M1B,Scarborough,Rouge,Malv
1	M1C	Scarborough	Highland Creek,Rouge Hill,Port Union	M1C,Scarborough,Highland Cr Hill,Port...
2	M1E	Scarborough	Guildwood,Morningside,West Hill	M1E,Scarborough,Guildwood,M Hill
3	M1G	Scarborough	Woburn	M1G,Scarborough,Woburn
4	M1H	Scarborough	Cedarbrae	M1H,Scarborough,Cedarbrae

In [56]:

```
nom = Nominatim()
```

In [57]:

```
n=nom.geocode('M1B, Scarborough, Rouge,Malvern')
n
```

Out[57]:

```
Location(Malvern, Scarborough—Rouge Park, Scarborough, Toronto, Gold
en Horseshoe, Ontario, M1B 4Y7, Canada, (43.8091955, -79.2217008, 0.
0))
```

In [58]:

```
n.latitude
```

Out[58]:

```
43.8091955
```

In [59]:
```
type(n)
```

Out[59]:
```
geopy.location.Location
```

Watch out for None values

In [60]:
```
n2=nom.geocode('M1E Scarborough Guildwood,Morningside,West Hill')
print(n2)
```

```
None
```

In [64]:
```
df_geopy['Coordinates'] =df_geopy['address'].apply(nom.geocode)
df_geopy.head()
```

Out[64]:

	PostalCode	Borough	Neighbourhood	
0	M1B	Scarborough	Rouge,Malvern	M1B,Scarborough,Rouge,Malv
1	M1C	Scarborough	Highland Creek,Rouge Hill,Port Union	M1C,Scarborough,Highland Cr Hill,Port...
2	M1E	Scarborough	Guildwood,Morningside,West Hill	M1E,Scarborough,Guildwood,I Hill
3	M1G	Scarborough	Woburn	M1G,Scarborough,Woburn
4	M1H	Scarborough	Cedarbrae	M1H,Scarborough,Cedarbrae

location objects created at 'Coordinates'

In [65]:

```
df_geopy.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 103 entries, 0 to 102
Data columns (total 5 columns):
PostalCode       103 non-null object
Borough          103 non-null object
Neighbourhood    103 non-null object
address          103 non-null object
Coordinates      5 non-null object
dtypes: object(5)
memory usage: 4.8+ KB
```

In [66]:

```
df_geopy.Coordinates[0]
```

Out[66]:

```
Location(Malvern, Scarborough—Rouge Park, Scarborough, Toronto, Gold
en Horseshoe, Ontario, M1B 4Y7, Canada, (43.8091955, -79.2217008, 0.
0))
```

In [67]:

```
print(df_geopy.Coordinates[1])
```

```
None
```

In [68]:
```
df_geopy['latitude']=df_geopy['Coordinates'].apply(lambda x: x.latitude if x !=None else None)
df_geopy['longitude']=df_geopy['Coordinates'].apply(lambda x: x.longitude if x !=None else None)
df_geopy.head()
```

Out[68]:

	PostalCode	Borough	Neighbourhood	
0	M1B	Scarborough	Rouge,Malvern	M1B,Scarborough,Rouge,Malv
1	M1C	Scarborough	Highland Creek,Rouge Hill,Port Union	M1C,Scarborough,Highland Cr Hill,Port...
2	M1E	Scarborough	Guildwood,Morningside,West Hill	M1E,Scarborough,Guildwood,M Hill
3	M1G	Scarborough	Woburn	M1G,Scarborough,Woburn
4	M1H	Scarborough	Cedarbrae	M1H,Scarborough,Cedarbrae

In [69]:
```
df_geopy.to_csv('geo_loc_py.csv')
```

As just 5 addresses were fruitful, we will go on to use the given geo-location data.

In [70]:
```
print('The latitude of', df_geopy.address[0],  'is', df_geopy.latitude[0], 'and its longitude is',df_geopy.longitude[0])
```

```
The latitude of M1B,Scarborough,Rouge,Malvern is 43.8091955 and its longitude is -79.2217008
```

2.2. Used the csv file to create the requested dataframe

In [71]:

```
# Load the Pandas libraries with alias 'pd'
import pandas as pd
# Read data from file 'filename.csv'
# (in the same directory that your python process is based)
# Control delimiters, rows, column names with read_csv (see later)
data2 = pd.read_csv("geopy.csv")
# Preview the first 5 lines of the loaded data
data2.head()
```

Out[71]:

	Unnamed: 0	PostalCode	Borough	Neighbourhood	
0	0	M1B	Scarborough	Rouge,Malvern	M1B, Scarborough Rouge,Malvern
1	1	M1C	Scarborough	Highland Creek,Rouge Hill,Port Union	M1C, Scarborough Creek,Rouge Hill,P
2	2	M1E	Scarborough	Guildwood,Morningside,West Hill	M1E, Scarborough Guildwood,Mornin Hill
3	3	M1G	Scarborough	Woburn	M1G, Scarborough
4	4	M1H	Scarborough	Cedarbrae	M1H, Scarborough Cedarbrae

In [72]:

```
data3 = pd.read_csv("Geospatial_Coordinates.csv")
# Preview the first 5 lines of the loaded data
data3.head()
```

Out[72]:

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

- Rename 'Postal Code'

In [73]:
```
data3.rename(columns={'Postal Code': 'PostalCode'}, inplace=True)
#data3.head()
```

In [74]:
```
data1 = pd.merge(data3, data2, how='inner', on=None, left_on=None, rig
ht_on=None,
         left_index=False, right_index=False, sort=True,
         suffixes=('_x', '_y'), copy=True, indicator=False,
         validate=None)

data1.head()
```

Out[74]:

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

In [75]:
```
data1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 103 entries, 0 to 102
Data columns (total 7 columns):
PostalCode       103 non-null object
Latitude         103 non-null float64
Longitude        103 non-null float64
Unnamed: 0       103 non-null int64
Borough          103 non-null object
Neighbourhood    103 non-null object
address          103 non-null object
dtypes: float64(2), int64(1), object(4)
memory usage: 6.4+ KB
```

- Rearrange columns and drop foreign key:

In [76]:

```
cols = data1.columns.tolist()
cols
```

Out[76]:

```
['PostalCode',
 'Latitude',
 'Longitude',
 'Unnamed: 0',
 'Borough',
 'Neighbourhood',
 'address']
```

In [77]:

```
new_column_order = ['PostalCode',
 'Borough',
 'Neighbourhood',
 'Latitude',
 'Longitude']
new_column_order
```

Out[77]:

```
['PostalCode', 'Borough', 'Neighbourhood', 'Latitude', 'Longitude']
```

In [78]:

```
data1 = data1[new_column_order]
#data1.head()
```

- Sort dataframe to match example:

In [79]:

```
sorted_df = data1.sort_values([ 'Neighbourhood', 'Latitude'], ascending=[True, True])
#sorted_df.head()
# no idea how to get it exacly like the exqample :(
```

In [80]:

```
sorted_df.reset_index(inplace=True)
#sorted_df.head()
```

In [81]:

```
sorted_cols =sorted_df.columns.tolist()
#sorted_cols
```

In [82]:

```
new_column_order2 = ['PostalCode',
 'Borough',
 'Neighbourhood',
 'Latitude',
 'Longitude']
new_column_order2
```

Out[82]:

```
['PostalCode', 'Borough', 'Neighbourhood', 'Latitude', 'Longitude']
```

In [83]:

```
sorted_dataframe = sorted_df[new_column_order]
sorted_dataframe.head()
```

Out[83]:

	PostalCode	Borough	Neighbourhood	Latitude	Longitude
0	M5H	DowntownToronto	Adelaide,King,Richmond	43.650571	-79.384568
1	M1S	Scarborough	Agincourt	43.794200	-79.262029
2	M1V	Scarborough	Agincourt North,L'Amoreaux East,Milliken,Steel...	43.815252	-79.284577
3	M9V	Etobicoke	Albion Gardens,Beaumond Heights,Humbergate,Jam...	43.739416	-79.588437
4	M8W	Etobicoke	Alderwood,Long Branch	43.602414	-79.543484

2.3. Q2_ notebook on Github repository. (2 marks)

In [84]:

```
sorted_dataframe.to_csv('sorted_geoloc.csv')
```

Question 3:

3.1. Build a test set with boroughs in Toronto

Import dependencies that we will need.

In [85]:

```
import numpy as np # library to handle data in a vectorized manner

import pandas as pd # library for data analsysis
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)

import json # library to handle JSON files

#!conda install -c conda-forge geopy --yes # uncomment this line if you haven't completed the Foursquare API lab
from geopy.geocoders import Nominatim # convert an address into latitude and longitude values

import requests # library to handle requests
from pandas.io.json import json_normalize # tranform JSON file into a pandas dataframe

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

# import k-means from clustering stage
from sklearn.cluster import KMeans

#!conda install -c conda-forge folium=0.5.0 --yes # uncomment this line if you haven't completed the Foursquare API lab
import folium # map rendering library

print('Libraries imported.')
```

```
Libraries imported.
```

bonus_loading data into json:

In [86]:

```
# library to handle JSON files

import pandas as pd

import json

sorted_dataframe.to_json(path_or_buf='geo_toronto.json', orient='table')
```

In [87]:

```
with open('geo_toronto.json') as json_data:
    Toronto_data = json.load(json_data)
```

In [88]:

```
#Toronto_data
# Data is in the 'data' field
```

In [89]:

```
neighborhoods_data = Toronto_data['data']
neighborhoods_data[0]
#Let's take a look at the first item in this list.
```

Out[89]:

```
{'index': 0,
 'PostalCode': 'M5H',
 'Borough': 'DowntownToronto',
 'Neighbourhood': 'Adelaide,King,Richmond',
 'Latitude': 43.6505712,
 'Longitude': -79.3845675}
```

In [90]:

```
sorted_dataframe.info()
sorted_dataframe.shape
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 103 entries, 0 to 102
Data columns (total 5 columns):
PostalCode       103 non-null object
Borough          103 non-null object
Neighbourhood    103 non-null object
Latitude         103 non-null float64
Longitude        103 non-null float64
dtypes: float64(2), object(3)
memory usage: 4.1+ KB
```

Out[90]:

```
(103, 5)
```

In [91]:

```
sorted_dataframe.head()
```

Out[91]:

	PostalCode	Borough	Neighbourhood	Latitude	Longitude
0	M5H	DowntownToronto	Adelaide,King,Richmond	43.650571	-79.384568
1	M1S	Scarborough	Agincourt	43.794200	-79.262029
2	M1V	Scarborough	Agincourt North,L'Amoreaux East,Milliken,Steel...	43.815252	-79.284577
3	M9V	Etobicoke	Albion Gardens,Beaumond Heights,Humbergate,Jam...	43.739416	-79.588437
4	M8W	Etobicoke	Alderwood,Long Branch	43.602414	-79.543484

```
In [92]: print('The dataframe has {} boroughs and {} neighborhoods.'.format(
                 len(sorted_dataframe['Borough'].unique()),
                 sorted_dataframe.shape[0]
             )
         )
```

```
The dataframe has 11 boroughs and 103 neighborhoods.
```

How to get coordonates:

```
In [93]: address = 'Adelaide'

         geolocator = Nominatim()
         location = geolocator.geocode(address)
         latitude = location.latitude
         longitude = location.longitude
         print('The geograpical coordinate of Adelaide are {}, {}.'.format(lati
         tude, longitude))
```

```
The geograpical coordinate of Adelaide are -34.9281805, 138.5999312.
```

How to map with Folium:

```
In [94]: !conda install -c conda-forge folium=0.5.0
```

```
Collecting package metadata: done
Solving environment: -
The environment is inconsistent, please check the package plan caref
ully
The following packages are causing the inconsistency:

  - defaults/linux-64::anaconda==5.3.1=py37_0
  - defaults/linux-64::astropy==3.0.4=py37h14c3975_0
  - defaults/linux-64::bkcharts==0.2=py37_0
  - defaults/linux-64::blaze==0.11.3=py37_0
  - defaults/linux-64::bokeh==0.13.0=py37_0
  - defaults/linux-64::bottleneck==1.2.1=py37h035aef0_1
  - defaults/linux-64::dask==0.19.1=py37_0
  - defaults/linux-64::datashape==0.5.4=py37_1
  - defaults/linux-64::mkl-service==1.1.2=py37h90e4bf4_5
  - defaults/linux-64::numba==0.39.0=py37h04863e7_0
  - defaults/linux-64::numexpr==2.6.8=py37hd89afb7_0
  - defaults/linux-64::odo==0.5.1=py37_0
  - defaults/linux-64::pytables==3.4.4=py37ha205bf6_0
  - defaults/linux-64::pytest-arraydiff==0.2=py37h39e3cac_0
  - defaults/linux-64::pytest-astropy==0.4.0=py37_0
  - defaults/linux-64::pytest-doctestplus==0.1.3=py37_0
  - defaults/linux-64::pywavelets==1.0.0=py37hdd07704_0
  - defaults/linux-64::scikit-image==0.14.0=py37hf484d3e_1
done

# All requested packages already installed.
```

Ready to generate maps, open them on your browser!

- if you cannot generate the maps open PGA*map**.html from the zip file

In [95]:
```
import pandas as pd
import folium

print('imported pandas & folium')
```

```
imported pandas & folium
```

Map generated with folium default markers:

In [96]:

```
import pandas as pd
import folium

#grab a random sample from df
subset_of_df = sorted_dataframe.sample(n=11)
map_test = folium.Map(location=[subset_of_df['Latitude'].mean(),
                                subset_of_df['Longitude'].mean()],
                      zoom_start=10)
#creating a Marker for each point in df_sample. Each point will get a
popup with their zip
for row in subset_of_df.itertuples():if you cannot
    map_test.add_child(folium.Marker(location=[row.Latitude ,row.Longi
tude],
           popup=row.Borough))


#map_test

#open map_test.html in browser
map_test.save("map_test.html")

# if you cannot generate the maps open PGA_map_*.html from the zip fil
e
```

Test on Borough data, map with MarkerClusters:

In [97]:

```
from folium.plugins import MarkerCluster
map_borough = folium.Map(location=[subset_of_df['Latitude'].mean(),
 subset_of_df['Longitude'].mean()],
 zoom_start=10)
mc = MarkerCluster()
#creating a Marker for each point in df_sample. Each point will get a
popup with their zip
for row in subset_of_df.itertuples():
    mc.add_child(folium.Marker(location=[row.Latitude,  row.Longitude]
,
                 popup=row.Borough))
    map_borough.add_child(mc)


#map_borough

#open in map_borough.html browser
map_borough.save("map_borough.html")

#if you cannot generate the maps open PGA_map_*.html from the zip file
```

3.2. Replicate the same analysis with the neighborhoods in Toronto.

In [98]:
```
import pandas as pd
import folium



#grab a random sample from df
toronto_n = sorted_dataframe.sample(n=20)
map_toronto = folium.Map(location=[toronto_n['Latitude'].mean(),
                                toronto_n['Longitude'].mean()],
                      zoom_start=10)
#creating a Marker for each point in df_sample. Each point will get a popup with their zip
for row in toronto_n.itertuples():
    map_toronto.add_child(folium.Marker(location=[row.Latitude ,row.Longitude],
           popup=row.Neighbourhood))


map_toronto

#open map_toronto.html in browser

map_toronto.save("map_toronto20.html")

#if you cannot generate the maps open PGA_map_*.html from the zip file
```

great got 20 neighbourhoods...could not get more :(

3.3 Used the Foursquare API to explore neighborhoods in Toronto.

-let's check the dataframe...

```
In [104]: sorted_dataframe.head()
```

Out[104]:

	PostalCode	Borough	Neighbourhood	Latitude	Longitude
0	M5H	DowntownToronto	Adelaide,King,Richmond	43.650571	-79.384568
1	M1S	Scarborough	Agincourt	43.794200	-79.262029
2	M1V	Scarborough	Agincourt North,L'Amoreaux East,Milliken,Steel...	43.815252	-79.284577
3	M9V	Etobicoke	Albion Gardens,Beaumond Heights,Humbergate,Jam...	43.739416	-79.588437
4	M8W	Etobicoke	Alderwood,Long Branch	43.602414	-79.543484

```
In [110]: sorted_dataframe.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 103 entries, 0 to 102
Data columns (total 5 columns):
PostalCode       103 non-null object
Borough          103 non-null object
Neighbourhood    103 non-null object
Latitude         103 non-null float64
Longitude        103 non-null float64
dtypes: float64(2), object(3)
memory usage: 4.1+ KB
```

```
In [105]: print('The dataframe has {} boroughs and {} neighborhoods.'.format(
                  len(sorted_dataframe['Borough'].unique()),
                  sorted_dataframe.shape[0]
              )
          )
```

```
The dataframe has 11 boroughs and 103 neighborhoods.
```

Use geopy library to get the latitude and longitude values of Toronto.

-In order to define an instance of the geocoder, leave out user_agent.

In [108]:
```
address = 'Toronto, CA'

geolocator = Nominatim()
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 43.653963, -79.387207.
```

Create a map of Toronto

- with its neighborhoods superimposed on top.
- if you cannot replicate the lab open PGA*map**.html from the zip file

In [114]:
```
# create map of Toronto using latitude and longitude values
map_toronto_neighbourhoods = folium.Map(location=[latitude, longitude], zoom_start=10)

# add markers to map
for lat, lng, borough, neighbourhood in zip(sorted_dataframe['Latitude'], sorted_dataframe['Longitude'], sorted_dataframe['Borough'], sorted_dataframe['Neighbourhood']):
    label = '{}, {}'.format(neighbourhood, borough)
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius=2,
        popup=label,
        color='blue',
        fill=True,
        fill_color='#3186cc',
        fill_opacity=0.7,
        parse_html=False).add_to(map_toronto_neighbourhoods)

map_toronto_neighbourhoods

map_toronto_neighbourhoods.save("map_toronto_neighbourhoods.html")

#open map_toronto_neighbourhoods.html in browser
#if you cannot generate the maps open PGA_map_*.html from the zip file
```

To kick off, let's focus on just one borough_...'York' for example:

In [115]:

```
address = 'York, Toronto'

geolocator = Nominatim()
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude
print('The geograpical coordinates of York, Toronto are {}, {}.'.forma
t(latitude, longitude))
```

```
The geograpical coordinates of York, Toronto are 43.6896191, -79.479
188.
```

Lets repeat as above, but for York, Toronto only:

In [119]:

```
york_data = sorted_dataframe[sorted_dataframe['Borough'] == 'York'].re
set_index(drop=True)
york_data
```

Out[119]:

	PostalCode	Borough	Neighbourhood	Latitude	Longitude
0	M6E	York	Caledonia-Fairbanks	43.689026	-79.453512
1	M6M	York	Del Ray,Keelesdale,Mount Dennis,Silverthorn	43.691116	-79.476013
2	M6C	York	Humewood-Cedarvale	43.693781	-79.428191
3	M6N	York	The Junction North,Runnymede	43.673185	-79.487262
4	M9N	York	Weston	43.706876	-79.518188

In [118]:

```
york_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 5 columns):
PostalCode       5 non-null object
Borough          5 non-null object
Neighbourhood    5 non-null object
Latitude         5 non-null float64
Longitude        5 non-null float64
dtypes: float64(2), object(3)
memory usage: 280.0+ bytes
```

In [120]:

```
# create map of Manhattan using latitude and longitude values
map_york_toronto = folium.Map(location=[latitude, longitude], zoom_start=11)

# add markers to map
for lat, lng, label in zip(york_data['Latitude'], york_data['Longitude'], york_data['Neighbourhood']):
    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_york_toronto)

map_york_toronto

map_york_toronto.save("map_york_toronto.html")

#open map_york_toronto.html in browser
#if you cannot generate the maps open PGA_map_*.html from the zip file
```

Foursquare:

@hidden_cell

-https://dataplatform.cloud.ibm.com/docs/content/wsj/analyze-data/hide_code.html (https://dataplatform.cloud.ibm.com/docs/content/wsj/analyze-data/hide_code.html)

Enter your Foursquare credentials:

- CLIENT_ID = 'cif' # your Foursquare ID
- CLIENT_SECRET = 'csf' # your Foursquare Secret
- VERSION = 'ymd' # Foursquare API version
- print('My credentails: ***')
- print('CLIENT_ID: ' + CLIENT_ID)
- print('CLIENT_SECRET:' + CLIENT_SECRET)

In [140]:

```
# The code was removed by Watson Studio for sharing.
```

```
My credentails: ***
```

Let's explore the first neighborhood in our dataframe.

- Get the neighborhood's name.

In [144]:

```
york_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 5 columns):
PostalCode       5 non-null object
Borough          5 non-null object
Neighbourhood    5 non-null object
Latitude         5 non-null float64
Longitude        5 non-null float64
dtypes: float64(2), object(3)
memory usage: 280.0+ bytes
```

In [145]:

```
neighbourhood_latitude = york_data.loc[0, 'Latitude'] # neighborhood l
atitude value
neighbourhood_longitude = york_data.loc[0, 'Longitude'] # neighborhood
longitude value

neighbourhood_name = york_data.loc[0, 'Neighbourhood'] # neighborhood
name

print('Latitude and longitude values of {} are {}, {}.'.format(neighbo
urhood_name,
                                                               neighbo
urhood_latitude,
                                                               neighbo
urhood_longitude))
```

```
Latitude and longitude values of Caledonia-Fairbanks are 43.6890256,
-79.453512.
```

Now, let's get the top 100 venues that are in Caledonia-Fairbanks within a radius of 500 meters.

- First, let's create the GET request URL. Name your URL url.

In [149]:
```
# The code was removed by Watson Studio for sharing.
```

your code should look like this:

LIMIT = 100

radius = 500

url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={},{}&radius={}&limit={}'.format((https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={},{}&radius={}&limit={}'.format()

```
    CLIENT_ID,

    CLIENT_SECRET,

    VERSION,

    neighbourhood_latitude,

    neighbourhood_longitude,

    radius,

    LIMIT)
```

url

In [154]:
```
york_results = requests.get(url).json()
#york_results
```

3.4. Get the most common venue categories in each neighborhood

- use get_category_type function from the Foursquare lab

In [153]:

```
# function that extracts the category of the venue
def get_category_type(row):
    try:
        categories_list = row['categories']
    except:
        categories_list = row['venue.categories']

    if len(categories_list) == 0:
        return None
    else:
        return categories_list[0]['name']
```

In [158]:

```
york_venues = york_results['response']['groups'][0]['items']

york_nearby_venues = json_normalize(york_venues) # flatten JSON

# filter columns
york_filtered_columns = ['venue.name', 'venue.categories', 'venue.loca
tion.lat', 'venue.location.lng']
york_nearby_venues = york_nearby_venues.loc[:, york_filtered_columns]

# filter the category for each row
york_nearby_venues['venue.categories'] = york_nearby_venues.apply(get_
category_type, axis=1)

# clean columns
york_nearby_venues.columns = [col.split(".")[-1] for col in york_nearb
y_venues.columns]

york_nearby_venues.head()
```

Out[158]:

	name	categories	lat	lng
0	Shoppers Drug Mart	Pharmacy	43.690651	-79.456310
1	KFC	Fast Food Restaurant	43.690647	-79.456326
2	Nairn Park	Park	43.690654	-79.456300
3	Maximum Woman	Women's Store	43.690651	-79.456333
4	Walmart	Market	43.690660	-79.456317

In [159]:

```
york_nearby_venues.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 4 columns):
name          6 non-null object
categories    6 non-null object
lat           6 non-null float64
lng           6 non-null float64
dtypes: float64(2), object(2)
memory usage: 272.0+ bytes
```

In [160]:

```
print('{} venues were returned by Foursquare.'.format(york_nearby_venues.shape[0]))
```

```
6 venues were returned by Foursquare.
```

Explore Neighborhoods in York:

- Let's create a function to repeat the same process to all the neighborhoods in York

In [170]:

```
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]['i
tems']

        # 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 f
or item in venue_list])
    nearby_venues.columns = ['Neighbourhood',
                  'Neighbourhood Latitude',
                  'Neighbourhood Longitude',
                  'Venue',
                  'Venue Latitude',
                  'Venue Longitude',
                  'Venue Category']

    return(nearby_venues)
```

In [171]:

```
york_venues = getNearbyVenues(names=york_data['Neighbourhood'],
                                   latitudes=york_data['Latitude'],
                                   longitudes=york_data['Longitude']
                                  )
```

```
Caledonia-Fairbanks
Del Ray,Keelesdale,Mount Dennis,Silverthorn
Humewood-Cedarvale
The Junction North,Runnymede
Weston
```

In [172]:

```
york_venues.head()
```

Out[172]:

	Neighbourhood	Neighbourhood Latitude	Neighbourhood Longitude	Venue	Venue Latitude	Venue Longitude
0	Caledonia-Fairbanks	43.689026	-79.453512	Shoppers Drug Mart	43.690651	-79.456310
1	Caledonia-Fairbanks	43.689026	-79.453512	KFC	43.690647	-79.456326
2	Caledonia-Fairbanks	43.689026	-79.453512	Nairn Park	43.690654	-79.456300
3	Caledonia-Fairbanks	43.689026	-79.453512	Maximum Woman	43.690651	-79.456333
4	Caledonia-Fairbanks	43.689026	-79.453512	Walmart	43.690660	-79.456317

In [173]:

```
york_venues.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20 entries, 0 to 19
Data columns (total 7 columns):
Neighbourhood              20 non-null object
Neighbourhood Latitude     20 non-null float64
Neighbourhood Longitude    20 non-null float64
Venue                      20 non-null object
Venue Latitude             20 non-null float64
Venue Longitude            20 non-null float64
Venue Category             20 non-null object
dtypes: float64(4), object(3)
memory usage: 1.2+ KB
```

Let's see how many venues were returned for each neighborhood:

```
In [174]: york_venues.groupby('Neighbourhood').count()
```

Out[174]:

	Neighbourhood Latitude	Neighbourhood Longitude	Venue	Venue Latitude	Venue Longitude	C
Neighbourhood						
Caledonia-Fairbanks	6	6	6	6	6	6
Del Ray,Keelesdale,Mount Dennis,Silverthorn	4	4	4	4	4	4
Humewood-Cedarvale	4	4	4	4	4	4
The Junction North,Runnymede	4	4	4	4	4	4
Weston	2	2	2	2	2	2

Now let's see how many types of venues there are in York:

```
In [176]: print('There are {} uniques categories.'.format(len(york_venues['Venue
          Category'].unique())))
```

```
There are 17 uniques categories.
```

3.5. Used these features to group the neighborhoods into clusters

- use one hot coding to analyse each of the neighbourhoods in York, Torono:

In [179]:

```
# one hot encoding
york_onehot = pd.get_dummies(york_venues[['Venue Category']], prefix="
", prefix_sep="")

# add neighborhood column back to dataframe
york_onehot['Neighbourhood'] = york_venues['Neighbourhood']

# move neighborhood column to the first column
york_fixed_columns = [york_onehot.columns[-1]] + list(york_onehot.colu
mns[:-1])
york_onehot = york_onehot[york_fixed_columns]

york_onehot.head()
```

Out[179]:

	Neighbourhood	Bus Line	Check Cashing Service	Convenience Store	Discount Store	Fast Food Restaurant	Field	Grocery Store
0	Caledonia-Fairbanks	0	0	0	0	0	0	0
1	Caledonia-Fairbanks	0	0	0	0	1	0	0
2	Caledonia-Fairbanks	0	0	0	0	0	0	0
3	Caledonia-Fairbanks	0	0	0	0	0	0	0
4	Caledonia-Fairbanks	0	0	0	0	0	0	0

```
In [180]: york_onehot.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20 entries, 0 to 19
Data columns (total 18 columns):
Neighbourhood            20 non-null object
Bus Line                 20 non-null uint8
Check Cashing Service    20 non-null uint8
Convenience Store        20 non-null uint8
Discount Store           20 non-null uint8
Fast Food Restaurant     20 non-null uint8
Field                    20 non-null uint8
Grocery Store            20 non-null uint8
Hockey Arena             20 non-null uint8
Market                   20 non-null uint8
Park                     20 non-null uint8
Pharmacy                 20 non-null uint8
Pizza Place              20 non-null uint8
Restaurant               20 non-null uint8
Sandwich Place           20 non-null uint8
Tennis Court             20 non-null uint8
Trail                    20 non-null uint8
Women's Store            20 non-null uint8
dtypes: object(1), uint8(17)
memory usage: 580.0+ bytes
```

Let's group by neighbourhoods:

In [181]:
```
york_grouped = york_onehot.groupby('Neighbourhood').mean().reset_index
()
york_grouped.head()
```

Out[181]:

	Neighbourhood	Bus Line	Check Cashing Service	Convenience Store	Discount Store	Fast Food Restaurant	Field	Gro S
0	Caledonia-Fairbanks	0.00	0.00	0.00	0.00	0.166667	0.00	0.00
1	Del Ray,Keelesdale,Mount Dennis,Silverthorn	0.00	0.25	0.00	0.25	0.000000	0.00	0.00
2	Humewood-Cedarvale	0.00	0.00	0.00	0.00	0.000000	0.25	0.00
3	The Junction North,Runnymede	0.25	0.00	0.25	0.00	0.000000	0.00	0.25
4	Weston	0.00	0.00	0.00	0.00	0.000000	0.00	0.00

In [182]:
```
york_grouped.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 18 columns):
Neighbourhood            5 non-null object
Bus Line                 5 non-null float64
Check Cashing Service    5 non-null float64
Convenience Store        5 non-null float64
Discount Store           5 non-null float64
Fast Food Restaurant     5 non-null float64
Field                    5 non-null float64
Grocery Store            5 non-null float64
Hockey Arena             5 non-null float64
Market                   5 non-null float64
Park                     5 non-null float64
Pharmacy                 5 non-null float64
Pizza Place              5 non-null float64
Restaurant               5 non-null float64
Sandwich Place           5 non-null float64
Tennis Court             5 non-null float64
Trail                    5 non-null float64
Women's Store            5 non-null float64
dtypes: float64(17), object(1)
memory usage: 800.0+ bytes
```

Let's find the top venues:

In [186]:

```
num_top_venues = 3

for hood in york_grouped['Neighbourhood']:
    print("----"+hood+"----")
    york_temp = york_grouped[york_grouped['Neighbourhood'] == hood].T.
reset_index()
    york_temp.columns = ['venue','freq']
    york_temp = york_temp.iloc[1:]
    york_temp['freq'] = york_temp['freq'].astype(float)
    york_temp = york_temp.round({'freq': 2})
    print(york_temp.sort_values('freq', ascending=False).reset_index(d
rop=True).head(num_top_venues))
    print('\n')
```

```
----Caledonia-Fairbanks----
                  venue  freq
0                  Park  0.33
1                Market  0.17
2  Fast Food Restaurant  0.17


----Del Ray,Keelesdale,Mount Dennis,Silverthorn----
            venue  freq
0  Discount Store  0.25
1  Sandwich Place  0.25
2      Restaurant  0.25


----Humewood-Cedarvale----
          venue  freq
0         Trail  0.25
1  Tennis Court  0.25
2         Field  0.25


----The Junction North,Runnymede----
               venue  freq
0           Bus Line  0.25
1  Convenience Store  0.25
2      Grocery Store  0.25


----Weston----
      venue  freq
0      Park   1.0
1  Bus Line   0.0
2     Trail   0.0
```

Let's put that into a pandas dataframe

- write a function to sort the venues in descending order.

In [187]:

```
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]
```

- Now let's create the new dataframe and display the top 7 venues for each neighborhood.

In [211]:

```
num_top_venues = 17

indicators = ['st', 'nd', 'rd']

# create columns according to number of top venues
columns = ['Neighbourhood']
for ind in np.arange(num_top_venues):
    try:
        columns.append('{}{} Most Common Venue'.format(ind+1, indicato
rs[ind]))
    except:
        columns.append('{}th Most Common Venue'.format(ind+1))

# create a new dataframe
york_neighbourhoods_venues_sorted = pd.DataFrame(columns=columns)

york_neighbourhoods_venues_sorted
```

Out[211]:

	Neighbourhood	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 Comn Ve

In [213]:

```
york_neighbourhoods_venues_sorted['Neighbourhood'] = york_grouped['Nei
ghbourhood']

york_neighbourhoods_venues_sorted.head(2)
```

Out[213]:

	Neighbourhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue
0	Caledonia-Fairbanks	NaN	NaN	NaN	NaN	NaN	NaN
1	Del Ray,Keelesdale,Mount Dennis,Silverthorn	NaN	NaN	NaN	NaN	NaN	NaN

In [214]:

```
for ind in np.arange(york_grouped.shape[0]):
    york_neighbourhoods_venues_sorted.iloc[ind, 1:] = return_most_comm
on_venues(york_grouped.iloc[ind, :], num_top_venues)

york_neighbourhoods_venues_sorted.head(2)
```

Out[214]:

	Neighbourhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Mo Comm Venu
0	Caledonia-Fairbanks	Park	Women's Store	Pharmacy	Fast Food Restaurant	Market	Pizza Place
1	Del Ray,Keelesdale,Mount Dennis,Silverthorn	Check Cashing Service	Sandwich Place	Restaurant	Discount Store	Women's Store	Grocery Store

3.4. Used the Folium library to generated maps to visualize neighborhoods on and how they cluster together.

Run k-means to cluster the neighborhood into 2 clusters.

In [215]:

```
# set number of clusters
kclusters = 2

york_grouped_clustering = york_grouped.drop('Neighbourhood', 1)

# run k-means clustering
york_kmeans = KMeans(n_clusters=kclusters, random_state=0).fit(york_gr
ouped_clustering)

# check cluster labels generated for each row in the dataframe
york_kmeans.labels_[0:5]
```

Out[215]:

```
array([1, 1, 1, 1, 0], dtype=int32)
```

now lets merge the clusters & sorted venue table:

In [216]:

```
# add clustering labels
york_neighbourhoods_venues_sorted.insert(0, 'Cluster Labels', york_kme
ans.labels_)

york_merged = york_data

# merge toronto_grouped with toronto_data to add latitude/longitude fo
r each neighborhood
york_merged = york_merged.join(york_neighbourhoods_venues_sorted.set_i
ndex('Neighbourhood'), on='Neighbourhood')

york_merged
```

Out[216]:

	PostalCode	Borough	Neighbourhood	Latitude	Longitude	Cluster Labels	1st Mo Commo Venu
0	M6E	York	Caledonia-Fairbanks	43.689026	-79.453512	1	Park
1	M6M	York	Del Ray,Keelesdale,Mount Dennis,Silverthorn	43.691116	-79.476013	1	Check Cashing Service
2	M6C	York	Humewood-Cedarvale	43.693781	-79.428191	1	Tennis Court
3	M6N	York	The Junction North,Runnymede	43.673185	-79.487262	1	Bus Lin
4	M9N	York	Weston	43.706876	-79.518188	0	Park

Generate maps to visualize your neighborhoods and how they cluster together.

In [217]:

```
# create map
york_map_clusters = folium.Map(location=[latitude, longitude], zoom_st
art=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(york_merged['Latitude'], york_merged
['Longitude'], york_merged['Neighbourhood'], york_merged['Cluster Labe
ls']):
    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(york_map_clusters)

york_map_clusters

york_map_clusters.save("york_map_clusters.html")

#open york_map_clusters.html in browser
#if you cannot generate the maps open PGA_map_*.html from the zip file
```

Examine Clusters

- examen & determine the discriminating venue categories that distinguish each cluster.

- Red Cluster:

In [218]:
```
york_merged.loc[york_merged['Cluster Labels'] == 0, york_merged.column
s[[1] + list(range(5, york_merged.shape[1]))]]
```

Out[218]:

	Borough	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue
4	York	0	Park	Women's Store	Hockey Arena	Check Cashing Service	Convenience Store	Discount Store

- Blue Cluster:

The neigbourhood Westson standsout from the four others in the York borough of Toronto, with an important Park and as can be seen on the map is close to the highway, its other imporant venues are women's stores and hockey areana.

In [219]:
```
york_merged.loc[york_merged['Cluster Labels'] == 1, york_merged.column
s[[1] + list(range(5, york_merged.shape[1]))]]
```

Out[219]:

	Borough	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Mos Commo Venu
0	York	1	Park	Women's Store	Pharmacy	Fast Food Restaurant	Market	Pizza Place
1	York	1	Check Cashing Service	Sandwich Place	Restaurant	Discount Store	Women's Store	Grocery Store
2	York	1	Tennis Court	Field	Trail	Hockey Arena	Women's Store	Grocery Store
3	York	1	Bus Line	Convenience Store	Pizza Place	Grocery Store	Hockey Arena	Check Cashing Service

The other neigbourhoods, also have parks but are are futher away from the highway, as can be seen on the map, their most common venues are check cashing services, tennis courts and a bus line etc. As this cluster is in a greener area, I'd prefer to live there!

3.5. Q3_ notebook on Github repository. (3 marks)

This notebook is an assignment for a course on **Coursera** called *Applied Data Science Capstone*, you can take this course online by clicking here (http://cocl.us/DP0701EN_Coursera_Week3_LAB2).