Segmenting and Clustering

Neighborhoods in Toronto

Peer-graded Assignment: Github_Segmenting and Clustering Neighborhoods in Toronto_Linda

In addion to the github repository with the full notebook, data set and html outputs of the maps, zipped:

Here’s a link to the full notebook on Watson:

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_code
s_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', 'Neighbourhoo
d'])
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_posta
l_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', 'Neighbour
hood'])
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 to get the coordinates of a few neighborhoods

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

```
Requirement already satisfied: geopy in /home/jupyterlab/conda/lib/pyth
on3.6/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.longitude)
```

```
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, Tor
onto, Ontario, M3A 1Z5, Canada

(43.7611243, -79.3240594)

{'place_id': 112261812, 'licence': 'Data © OpenStreetMap contributors,
ODbL 1.0. https://osm.org/copyright', 'osm_type': 'way', 'osm_id': 1604
06962, 'boundingbox': ['43.761106', '43.7612191', '-79.3242996', '-79.3
239088'], 'lat': '43.7611243', 'lon': '-79.3240594', 'display_name': 'P
arkwoods Village Drive, Parkway East, Don Valley East, North York, Toro
nto, Ontario, M3A 1Z5, Canada', 'class': 'highway', 'type': 'secondar
y', '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', 'Downto
wn Toronto'],
                         'Neighbourhood': ['Parkwoods', 'Victoria Villag
e', '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,axis
=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/13166c909ad
2f2d76b929a4227c952630ebaf0d729f6317eb09cbceccbab/geocoder-1.38.1-py2.p
y3-none-any.whl (98kB)
    100% |████████████████████████████████| 102kB 17.7MB/s
Requirement already satisfied: click in /home/jupyterlab/conda/lib/pyth
on3.6/site-packages (from geocoder) (7.0)
Requirement already satisfied: requests in /home/jupyterlab/conda/lib/p
ython3.6/site-packages (from geocoder) (2.21.0)
Collecting ratelim (from geocoder)
  Downloading https://files.pythonhosted.org/packages/f2/98/7e6d147fd16
a10a5f821db6e25f192265d6ecca3d82957a4fdd592cad49c/ratelim-0.1.6-py2.py3
-none-any.whl
Requirement already satisfied: future in /home/jupyterlab/conda/lib/pyt
hon3.6/site-packages (from geocoder) (0.17.1)
Requirement already satisfied: six in /home/jupyterlab/conda/lib/python
3.6/site-packages (from geocoder) (1.12.0)
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in /home/jupyterla
b/conda/lib/python3.6/site-packages (from requests->geocoder) (3.0.4)
Requirement already satisfied: certifi>=2017.4.17 in /home/jupyterlab/c
onda/lib/python3.6/site-packages (from requests->geocoder) (2019.3.9)
Requirement already satisfied: urllib3<1.25,>=1.21.1 in /home/jupyterla
b/conda/lib/python3.6/site-packages (from requests->geocoder) (1.24.1)
Requirement already satisfied: idna<2.9,>=2.5 in /home/jupyterlab/cond
a/lib/python3.6/site-packages (from requests->geocoder) (2.8)
Requirement already satisfied: decorator in /home/jupyterlab/conda/lib/
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 csv file.

In [45]:

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

In [46]:

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

```
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]:

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

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 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/pyth
on3.6/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,Malvern
1	M1C	Scarborough	Highland Creek,Rouge Hill,Port Union	M1C,Scarborough,Highland Creek Hill,Port...
2	M1E	Scarborough	Guildwood,Morningside,West Hill	M1E,Scarborough,Guildwood,Mor 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, Golden
         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
```

Use 'address' to get geocode coordinates:

Geocoding (Latitude/Longitude Lookup) Required parameters in a geocoding request: address — The street address that you want to geocode, in the format used by the national postal service of the country concerned. Additional address elements such as business names and unit, suite or floor numbers should be avoided.

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,Malvern
1	M1C	Scarborough	Highland Creek,Rouge Hill,Port Union	M1C,Scarborough,Highland Creek Hill,Port...
2	M1E	Scarborough	Guildwood,Morningside,West Hill	M1E,Scarborough,Guildwood,Mor Hill
3	M1G	Scarborough	Woburn	M1G,Scarborough,Woburn
4	M1H	Scarborough	Cedarbrae	M1H,Scarborough,Cedarbrae

A few 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, Golden
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,Malvern
1	M1C	Scarborough	Highland Creek,Rouge Hill,Port Union	M1C,Scarborough,Highland Creek Hill,Port...
2	M1E	Scarborough	Guildwood,Morningside,West Hill	M1E,Scarborough,Guildwood,Mor 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 csv.

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 lon
gitude 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	a
0	0	M1B	Scarborough	Rouge,Malvern	M1B, Scarborough, Rouge,Malvern
1	1	M1C	Scarborough	Highland Creek,Rouge Hill,Port Union	M1C, Scarborough, H Creek,Rouge Hill,Po...
2	2	M1E	Scarborough	Guildwood,Morningside,West Hill	M1E, Scarborough, Guildwood,Morningsi Hill
3	3	M1G	Scarborough	Woburn	M1G, Scarborough, W
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, right
_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	Neighbourhoo
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,We 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

2.6. Submit a link to your Notebook on your Github repository. (2 marks)

In [84]:

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

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).