SegmentingToronto

February 17, 2021

1 Toronto Clustering Visualization

1.1 Clustering Toronto's Postal Codes into 5 City Boroughs

1.1.1 I combined the data from Wikipedia and a CSV to draw Toronto's clustered postal codes on a map.

Installing needed dependencies

[10]:
```
!pip install lxml
!pip install branca==0.3.1
print("done")
```

```
Requirement already satisfied: lxml in
/home/jupyterlab/conda/envs/python/lib/python3.6/site-packages (4.6.2)
Collecting branca==0.3.1
  Downloading https://files.pythonhosted.org/packages/63/36/1c93318e9653f4e414a2
e0c3b98fc898b4970e939afeedeee6075dd3b703/branca-0.3.1-py3-none-any.whl
Requirement already satisfied: six in
/home/jupyterlab/conda/envs/python/lib/python3.6/site-packages (from
branca==0.3.1) (1.15.0)
Requirement already satisfied: jinja2 in
/home/jupyterlab/conda/envs/python/lib/python3.6/site-packages (from
branca==0.3.1) (2.11.2)
Requirement already satisfied: MarkupSafe>=0.23 in
/home/jupyterlab/conda/envs/python/lib/python3.6/site-packages (from
jinja2->branca==0.3.1) (1.1.1)
Installing collected packages: branca
  Found existing installation: branca 0.4.1
    Uninstalling branca-0.4.1:
      Successfully uninstalled branca-0.4.1
Successfully installed branca-0.3.1
done
```

Scraping Wikipedia for the data.

If the code below returns: "lxml not found, please install it", run the code above and restart the Kernal

[11]:
```
import pandas as pd
url = 'https://en.wikipedia.org/wiki/List_of_postal_codes_of_Canada:_M'
df_list = pd.read_html(url)[0]
```

Cleaning up the data

[12]:
```
df_list = df_list[df_list['Borough']!='Not assigned']
df_list.shape
df_list=df_list.reset_index(drop=True)
df_list[0:5]
```

[12]:

	Postal Code	Borough	Neighbourhood
0	M3A	North York	Parkwoods
1	M4A	North York	Victoria Village
2	M5A	Downtown Toronto	Regent Park, Harbourfront
3	M6A	North York	Lawrence Manor, Lawrence Heights
4	M7A	Downtown Toronto	Queen's Park, Ontario Provincial Government

[13]:
```
!pip install geocoder
```

```
Requirement already satisfied: geocoder in
/home/jupyterlab/conda/envs/python/lib/python3.6/site-packages (1.38.1)
Requirement already satisfied: click in
/home/jupyterlab/conda/envs/python/lib/python3.6/site-packages (from geocoder)
(7.1.2)
Requirement already satisfied: requests in
/home/jupyterlab/conda/envs/python/lib/python3.6/site-packages (from geocoder)
(2.25.0)
Requirement already satisfied: six in
/home/jupyterlab/conda/envs/python/lib/python3.6/site-packages (from geocoder)
(1.15.0)
Requirement already satisfied: ratelim in
/home/jupyterlab/conda/envs/python/lib/python3.6/site-packages (from geocoder)
(0.1.6)
Requirement already satisfied: future in
/home/jupyterlab/conda/envs/python/lib/python3.6/site-packages (from geocoder)
(0.18.2)
Requirement already satisfied: chardet<4,>=3.0.2 in
/home/jupyterlab/conda/envs/python/lib/python3.6/site-packages (from
requests->geocoder) (3.0.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in
/home/jupyterlab/conda/envs/python/lib/python3.6/site-packages (from
requests->geocoder) (1.25.11)
Requirement already satisfied: certifi>=2017.4.17 in
/home/jupyterlab/conda/envs/python/lib/python3.6/site-packages (from
requests->geocoder) (2020.12.5)
Requirement already satisfied: idna<3,>=2.5 in
/home/jupyterlab/conda/envs/python/lib/python3.6/site-packages (from
```

```
requests->geocoder) (2.10)
Requirement already satisfied: decorator in
/home/jupyterlab/conda/envs/python/lib/python3.6/site-packages (from
ratelim->geocoder) (4.4.2)
```

[14]:
```
import geocoder # import geocoder


def coords(postal_code):
    # initialize your variable to None
    lat_lng_coords = None
    print('here')
    print(postal_code)

    # loop until you get the coordinates
    while(lat_lng_coords is None):
        print(lat_lng_coords)
        g = geocoder.google('{}, Toronto, Ontario'.format(str(postal_code)))
        print(g)
        lat_lng_coords = g.latlng
        print(lat_lng_coords)

    latitude = lat_lng_coords[0]
    longitude = lat_lng_coords[1]

    print(latitude)

    return latitude, longitude

# GEOCODER kept freezing
#df_list['latitude'] = df_list.apply(lambda row: coords(row['Postal Code'])[0],␣
 ↪axis=1)
#df_list['longitude'] = df_list.apply(lambda row: coords(row['Postal␣
 ↪Code'])[1], axis=1)
```

Geocoder kept freezing, so I used the CSV data

[15]:
```
df_coords = pd.read_csv("Geospatial_Coordinates.csv")
df_list1 = df_list.merge(df_coords, how='left', on='Postal Code')
df_list1[0:5]
```

[15]:

	Postal Code	Borough	Neighbourhood
0	M3A	North York	Parkwoods
1	M4A	North York	Victoria Village
2	M5A	Downtown Toronto	Regent Park, Harbourfront
3	M6A	North York	Lawrence Manor, Lawrence Heights
4	M7A	Downtown Toronto	Queen's Park, Ontario Provincial Government

```
    Latitude  Longitude
0  43.753259 -79.329656
1  43.725882 -79.315572
2  43.654260 -79.360636
3  43.718518 -79.464763
4  43.662301 -79.389494
```

1.1.2 Initial Visualization of the Postal Codes

[16]:
```
# create map of Manhattan using latitude and longitude values
import folium # map rendering library
map_toronto = folium.Map(location=[43.696343524107384, -79.40858261970551],␣
 ↪zoom_start=11)

# add markers to map
for lat, lng, borough, neighborhood in zip(df_list1['Latitude'],␣
 ↪df_list1['Longitude'], df_list1['Borough'], df_list1['Neighbourhood']):
    label = '{}, {}'.format(neighborhood, borough)
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius=5,
        popup=label,
        color='blue',
        fill=True,
        fill_color='#3186cc',
        fill_opacity=0.7,
        parse_html=False).add_to(map_toronto)

map_toronto
```

[16]:
```
<folium.folium.Map at 0x7f05888676d8>
```

1.1.3 Clustering and Visualizing

Since Toronto is kind of like the New York City of Canada, I wanted to split up Toronto into five 'boroughs' and see how they were clustered.

I used K means to cluster the Toronto Postal Codes into 5 clusters.

[17]:
```
from sklearn.cluster import KMeans
kclusters = 5

df_list2 = df_list1.drop(['Neighbourhood', 'Borough'], axis=1)
df_list2 = df_list2.set_index("Postal Code")
df_list2

kmeans = KMeans(n_clusters=kclusters, random_state=0).fit(df_list2)
```

```
kmeans.labels_
df_list2.insert(0, 'Cluster Labels', kmeans.labels_)
df_list2 = df_list2.drop(['Latitude', 'Longitude'], axis=1)
df_list3 = df_list1.merge(df_list2, on='Postal Code')
df_list3.head()
```

[17]:

	Postal Code	Borough	Neighbourhood	Latitude	Longitude	Cluster Labels
0	M3A	North York	Parkwoods	43.753259	-79.329656	4
1	M4A	North York	Victoria Village	43.725882	-79.315572	4
2	M5A	Downtown Toronto	Regent Park, Harbourfront	43.654260	-79.360636	2
3	M6A	North York	Lawrence Manor, Lawrence Heights	43.718518	-79.464763	3
4	M7A	Downtown Toronto	Queen's Park, Ontario Provincial Government	43.662301	-79.389494	2

Then I drew the clusters on the map using Folium.

[18]:
```
import numpy as np
import matplotlib.cm as cm
import matplotlib.colors as colors

map_clusters = folium.Map(location=[43.696343524107384, -79.40858261970551],␣
 ↪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(df_list3['Latitude'], df_list3['Longitude'],␣
 ↪df_list3['Neighbourhood'], df_list3['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)

map_clusters
```

[18]: <folium.folium.Map at 0x7f053660e550>

1.1.4 Thank you for taking the time to check out my Data Science mini project!