battle-2-part1

August 3, 2019

1 Capstone Project - The Battle of Neighborhoods (Week 2)

1.1 Introduction

New York City comprises 5 boroughs sitting where the Hudson River meets the Atlantic Ocean. At its core is Manhattan, a densely populated borough that's among the world's major commercial, financial and cultural centers. Its iconic sites include skyscrapers such as the Empire State Building and sprawling Central Park. Broadway theater is staged in neon-lit Times Square.

London, the capital of England and the United Kingdom, is a 21st-century city with history stretching back to Roman times. At its centre stand the imposing Houses of Parliament, the iconic 'Big Ben' clock tower and Westminster Abbey, site of British monarch coronations. Across the Thames River, the London Eye observation wheel provides panoramic views of the South Bank cultural complex, and the entire city.

1.2 Description of the problem

We will explored New York City and London and segmented and clustered their neighborhoods. Both cities are very diverse and are very similar. Both cities are a densely populated boroughs that's among the world's major commercial, financial and cultural centers. . We will to compare the neighborhoods of the two cities and determine how similar or dissimilar they are. We will define that people like to do more in the cities, which places are often visited. Knowing this information we can think of how to use this. For exemple, open a new restaurant or supermarket, entertainment center or gift shop. As we can see in the next task that although there are Mexican restaurants in London, but they are not popular, entertainment is centrally located and almost none in areas farther from the center. We may also use this information for advertising purposes, etc

1.3 Description of Data.

This project will rely on public data from Wikipedia and Foursquare.

London is the capital of and largest city in England and the United Kingdom. It is administered by the City of London and 32 London boroughs.

We will get information about the areas of London https://en.wikipedia.org/wiki/List_of_areas_of_London

I will use dataset https://geo.nyu.edu/catalog/nyu_2451_34572 for information about boroughs of NYC

In [1]:
```
# library for BeautifulSoup
from bs4 import BeautifulSoup

import numpy as np
import pandas as pd
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)

# library to handle JSON files
import json

!pip -q install geopy
# conda install -c conda-forge geopy --yes # uncomment this line if you haven't comple

# convert an address into latitude and longitude values
from geopy.geocoders import Nominatim

# library to handle requests
import requests

# tranform JSON file into a pandas dataframe
from pandas.io.json import json_normalize

# 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

# install the Geocoder
!pip -q install geocoder
import geocoder

# import time
import time

# !conda install -c conda-forge folium=0.5.0 --yes # uncomment this line if you haven'
!pip -q install folium
```

```
        import folium # map rendering library

        from PIL import Image # converting images into arrays

        %matplotlib inline

        import matplotlib as mpl
        import matplotlib.pyplot as plt

        mpl.style.use('ggplot') # optional: for ggplot-like style

        # check for latest version of Matplotlib
        #print ('Matplotlib version: ', mpl.__version__) # >= 2.0.0

        # install wordcloud
        !conda install -c conda-forge wordcloud==1.4.1 --yes
        from wordcloud import WordCloud, STOPWORDS

        print ('...Done')
```

```
Solving environment: done


==> WARNING: A newer version of conda exists. <==
  current version: 4.5.12
  latest version: 4.7.10

Please update conda by running

    $ conda update -n base conda



# All requested packages already installed.

...Done
```

1.3.1 London

```
In [133]: # download data and parse it:
          r = requests.get('https://en.wikipedia.org/wiki/List_of_areas_of_London')
          soup = BeautifulSoup(r.text, 'html.parser')
          table=soup.find('table', attrs={'class':'wikitable sortable'})

In [134]: #get headers:
          headers=table.findAll('th')
```

```
        for i, head in enumerate(headers): headers[i]=str(headers[i]).replace("<th>","").repl
        #headers
```

In [135]:
```
        #Find all items and skip first one:
        rows=table.findAll('tr')
        rows=rows[1:len(rows)]
        #rows
```

In [136]:
```
        # skip all meta symbols and line feeds between rows:
        for i, row in enumerate(rows): rows[i] = str(rows[i]).replace("\n</td></tr>","").repl
        #rows
```

In [137]:
```
        # make dataframe, expand rows and drop the old one:
        df=pd.DataFrame(rows)
        df[headers] = df[0].str.split("</td>\n<td>", n = 7, expand = True)
        df.drop(columns=[0],inplace=True)#

        df.rename(columns={'Location': 'neighborhoods', 'London\xa0borough': 'borough', 'Post
        df.drop(columns={'OS grid ref'},inplace=True)
        df.head(3)
```

Out[137]:

	neighborhoods	borough	posttown	postcode	Dialăcode
0	<a href="/wiki/Abbey_Wood" title="Abbey Wood">...	Bexley, Greenwich <sup class="reference" id="...	LONDON	SE2	020
1	<a href="/wiki/Acton,_London" title="Acton, Lo...	Ealing, Hammersmith and Fulham<sup class="refe...	LONDON	W3, W4	020
2	<a href="/wiki/Addington,_London" title="Addin...	Croydon<sup class="reference" id="cite_ref-mil...	CROYDON	CR0	020

In [138]:
```
        df.update(df.neighborhoods.loc[lambda x: x.str.contains('title')].str.extract('title=
        # delete Toronto annotation from Neighbourhood:
        df.update(df.neighborhoods.loc[lambda x: x.str.contains('London')].str.replace(", Lo
```

In [139]:
```
        for i in range(0, df.shape[0]-1):
            #print(df.borough.get_values()[i])
            c = df.borough.get_values()[i].split('<')[0]
            df.borough[i] = c

        df = df.drop('borough', axis=1).join(df['borough'].str.split(',', expand=True).stack
        df = df.drop('posttown', axis=1).join(df['posttown'].str.split(',', expand=True).sta

        df = df.drop('postcode', axis=1).join(df['postcode'].str.split(',', expand=True).sta
```

```
In [140]: df.head()
```

```
Out[140]:   neighborhoods Dialăcode borough posttown postcode
          0    Abbey Wood       020  Bexley   LONDON      SE2
          0    Abbey Wood       020  Bexley   LONDON      SE2
          0    Abbey Wood       020  Bexley   LONDON      SE2
          0    Abbey Wood       020  Bexley   LONDON      SE2
          0    Abbey Wood       020  Bexley   LONDON      SE2
```

```
In [141]: df.shape
```

```
Out[141]: (2856, 5)
```

```
In [144]: df.drop_duplicates(keep = False, inplace = True)
```

```
In [145]: df.head()
```

```
Out[145]:       neighborhoods Dialăcode      borough posttown postcode
          2         Addington       020      Croydon  CROYDON      CR0
          3        Addiscombe       020      Croydon  CROYDON      CR0
          5  Aldborough Hatch       020    Redbridge   ILFORD      IG2
          6           Aldgate       020         City   LONDON      EC3
          7           Aldwych       020  Westminster   LONDON      WC2
```

```
In [146]: df.shape
```

```
Out[146]: (578, 5)
```

Now, only the Boroughs with London Post-town will be used for our search of location. Therefore, all the non-post-town are dropped.

```
In [147]: df_london = df
          df_london = df_london[df_london['posttown'].str.contains('LONDON')]

          df_london.drop_duplicates(keep = False, inplace = True)
```

```
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  after removing the cwd from sys.path.
```

```
In [148]: df_london.head()
```

```
Out[148]:    neighborhoods Dialăcode      borough posttown postcode
          6        Aldgate       020         City   LONDON      EC3
          7        Aldwych       020  Westminster   LONDON      WC2
          9        Anerley       020      Bromley   LONDON     SE20
          10         Angel       020    Islington   LONDON      EC1
          10         Angel       020    Islington   LONDON       N1
```

Geocoder dont read my whole data, and i divide my dataset on smaller parts

In [149]:
```
# Defining a function to use --> get_latlng()'''
def get_latlng(arcgis_geocoder):

    # Initialize the Location (lat. and long.) to "None"
    lat_lng_coords = None

    # While loop helps to create a continous run until all the location coordinates
    while(lat_lng_coords is None):
        g = geocoder.arcgis('{}, London, United Kingdom'.format(arcgis_geocoder))
        lat_lng_coords = g.latlng
    return lat_lng_coords
# Geocoder ends here
```

In [150]:
```
# New dataframe for  postcodes started with "W"
df_w = df_london[df_london['postcode'].str.startswith(('W'))].reset_index(drop=True)
```

In [151]:
```
df_w.head()
```

Out[151]:

	neighborhoods	Dialăcode	borough	posttown	postcode
0	Aldwych	020	Westminster	LONDON	WC2
1	Bayswater	020	Westminster	LONDON	W2
2	Bedford Park	020	Ealing	LONDON	W4
3	Bloomsbury	020	Camden	LONDON	WC1
4	Charing Cross	020	Westminster	LONDON	WC2

In [152]:
```
postcode = df_w['postcode']
postcode
coordinates = [get_latlng(postcode) for postcode in postcode.tolist()]
```

In [153]:
```
df_with_coordinates = df_w

# The obtained coordinates (latitude and longitude) are joined with the dataframe as
df_with_coordinates = pd.DataFrame(coordinates, columns = ['Latitude', 'Longitude'])
df_w['Latitude'] = df_with_coordinates['Latitude']
df_w['Longitude'] = df_with_coordinates['Longitude']
```

In [154]:
```
df_w.head()
```

Out[154]:

	neighborhoods	Dialăcode	borough	posttown	postcode	Latitude	Longitude
0	Aldwych	020	Westminster	LONDON	WC2	51.51651	-0.11968
1	Bayswater	020	Westminster	LONDON	W2	51.51494	-0.18048
2	Bedford Park	020	Ealing	LONDON	W4	51.48944	-0.26194
3	Bloomsbury	020	Camden	LONDON	WC1	51.52450	-0.12273
4	Charing Cross	020	Westminster	LONDON	WC2	51.51651	-0.11968

In [155]:
```
# # New dataframe for  postcodes started with "S"
df_s = df_london[df_london['postcode'].str.startswith(('S'))].reset_index(drop=True)
```

```
In [156]: df_s.head()
```

Out[156]:

	neighborhoods	Dialăcode	borough	posttown	postcode
0	Anerley	020	Bromley	LONDON	SE20
1	Balham	020	Wandsworth	LONDON	SW12
2	Bankside	020	Southwark	LONDON	SE1
3	Barnes	020	Richmond upon Thames	LONDON	SW13
4	Battersea	020	Wandsworth	LONDON	SW11

```
In [157]: postcode = df_s['postcode']
          postcode
          coordinates = [get_latlng(postcode) for postcode in postcode.tolist()]

In [158]: df_with_coordinates_s = df_s

          # The obtained coordinates (latitude and longitude) are joined with the dataframe as
          df_with_coordinates_s = pd.DataFrame(coordinates, columns = ['Latitude', 'Longitude']
          df_s['Latitude'] = df_with_coordinates_s['Latitude']
          df_s['Longitude'] = df_with_coordinates_s['Longitude']

In [159]: df_s.head()
```

Out[159]:

	neighborhoods	Dialăcode	borough	posttown	postcode	Latitude	Longitude
0	Anerley	020	Bromley	LONDON	SE20	51.41009	-0.05683
1	Balham	020	Wandsworth	LONDON	SW12	51.44822	-0.14839
2	Bankside	020	Southwark	LONDON	SE1	51.49960	-0.09613
3	Barnes	020	Richmond upon Thames	LONDON	SW13	51.47457	-0.24212
4	Battersea	020	Wandsworth	LONDON	SW11	51.46760	-0.16290

```
In [160]: # df_london_allpart = df_s and df_w
          df_london_allpart = df_s.append(df_w, ignore_index=True)

In [161]: df_london_allpart.shape
```

Out[161]: (129, 7)

```
In [162]: # New dataframe for  postcodes started with "E"
          df_e = df_london[df_london['postcode'].str.startswith(('E'))].reset_index(drop=True)

In [163]: postcode = df_e['postcode']
          postcode
          coordinates = [get_latlng(postcode) for postcode in postcode.tolist()]
```

```
In [164]: f_with_coordinates_e = df_e

          # The obtained coordinates (latitude and longitude) are joined with the dataframe as
          df_with_coordinates_e = pd.DataFrame(coordinates, columns = ['Latitude', 'Longitude']
          df_e['Latitude'] = df_with_coordinates_e['Latitude']
          df_e['Longitude'] = df_with_coordinates_e['Longitude']

In [165]: df_london_allpart = df_london_allpart.append(df_e, ignore_index=True)

In [166]: # New dataframe for  postcodes started with "N"
          df_n = df_london[df_london['postcode'].str.startswith(('N'))].reset_index(drop=True)

In [167]: postcode = df_n['postcode']
          postcode
          coordinates = [get_latlng(postcode) for postcode in postcode.tolist()]

In [168]: df_with_coordinates_s = df_n

          # The obtained coordinates (latitude and longitude) are joined with the dataframe as
          df_with_coordinates_n = pd.DataFrame(coordinates, columns = ['Latitude', 'Longitude']
          df_n['Latitude'] = df_with_coordinates_n['Latitude']
          df_n['Longitude'] = df_with_coordinates_n['Longitude']

In [169]: df_london_allpart = df_london_allpart.append(df_n, ignore_index=True)

In [170]: #df_london_allpart.head(10)

In [171]: # New dataframe for  postcodes started with "d"
          df_d = df_london[df_london['postcode'].str.startswith(('D'))].reset_index(drop=True)

In [172]: postcode = df_d['postcode']
          postcode
          coordinates = [get_latlng(postcode) for postcode in postcode.tolist()]

In [173]: df_with_coordinates_s = df_d

          # The obtained coordinates (latitude and longitude) are joined with the dataframe as
          df_with_coordinates_d = pd.DataFrame(coordinates, columns = ['Latitude', 'Longitude']
          df_d['Latitude'] = df_with_coordinates_d['Latitude']
          df_d['Longitude'] = df_with_coordinates_d['Longitude']

In [174]: df_london_allpart = df_london_allpart.append(df_d, ignore_index=True)

In [175]: # New dataframe for  postcodes started with "I"/ same =E18
          df_i = df_london[df_london['postcode'].str.startswith(('I'))].reset_index(drop=True)

In [176]: postcode = df_i['postcode']
          postcode
          coordinates = [get_latlng(postcode) for postcode in postcode.tolist()]
```

```
In [177]: df_with_coordinates_s = df_i

          # The obtained coordinates (latitude and longitude) are joined with the dataframe as
          df_with_coordinates_i = pd.DataFrame(coordinates, columns = ['Latitude', 'Longitude']
          df_i['Latitude'] = df_with_coordinates_i['Latitude']
          df_i['Longitude'] = df_with_coordinates_i['Longitude']

In [178]: df_london_allpart = df_london_allpart.append(df_i, ignore_index=True)

In [179]: df_london_allpart.head()

Out[179]:   neighborhoods Dialăcode              borough posttown postcode  Latitude  \
          0       Anerley       020              Bromley   LONDON     SE20  51.41009
          1        Balham       020           Wandsworth   LONDON     SW12  51.44822
          2      Bankside       020            Southwark   LONDON      SE1  51.49960
          3        Barnes       020  Richmond upon Thames   LONDON     SW13  51.47457
          4     Battersea       020           Wandsworth   LONDON     SW11  51.46760

             Longitude
          0   -0.05683
          1   -0.14839
          2   -0.09613
          3   -0.24212
          4   -0.16290

In [180]: df_london_allpart['borough'].unique()

Out[180]: array(['Bromley', 'Wandsworth', 'Southwark', 'Richmond upon Thames',
                 'Westminster', 'Lewisham', 'Greenwich', 'Lambeth',
                 'Kensington and Chelsea', 'Merton', 'Bexley',
                 'Hammersmith and Fulham', 'Kingston upon Thames', 'Croydon',
                 'Ealing', 'Camden', 'Hounslow', 'Camden and Islington', 'City',
                 'Islington', 'Tower Hamlets', 'Waltham Forest', 'Newham', 'Hackney',
                 'Islington &amp;amp; City', 'Redbridge', 'Enfield', 'Haringey',
                 'Barnet', 'Brent', 'Haringey and Barnet', 'Dartford'], dtype=object)

In [181]: print('The dataframe has {} boroughs and {} neighborhoods.'.format(
                  len(df_london_allpart['borough'].unique()),
                  df_london_allpart.shape[0]
              )
          )

The dataframe has 32 boroughs and 285 neighborhoods.
```

Use geopy library to get the latitude and longitude values of London. In order to define an instance of the geocoder, we need to define a user_agent. We will name our agent ny_explorer, as shown below.

```
In [183]: address = 'London, uk'

          geolocator = Nominatim(user_agent="uk_explorer")
          location = geolocator.geocode(address)
          latitude = location.latitude
          longitude = location.longitude
          print('The geograpical coordinate of London, uk {}, {}.'.format(latitude, longitude)
```

```
The geograpical coordinate of London, uk 51.4893335, -0.144055084527687.
```

Create a map of London with borough superimposed on top.

```
In [184]: # create map of London using latitude and longitude values
          map_london = folium.Map(location=[latitude, longitude], zoom_start=10)

          # add markers to map
          for lat, lng, label in zip(df_london_allpart['Latitude'], df_london_allpart['Longitu
              label = '{}.format(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_london)

          map_london
```

```
Out[184]: <folium.folium.Map at 0x7f1b72486518>
```