Untitled8

August 10, 2019

1 Coursera Capstone Project

1.1 The Battle of Neighborhoods - Final Report (Week 1 and 2)

```
[1]: import numpy as np # library to handle data in a vectorized manner
     import time
     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
     import requests # library to handle requests
     from pandas.io.json import json_normalize # tranform JSON file into a pandas dataframe


     !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

     !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
     import folium # map rendering library
     from folium import plugins

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

     import seaborn as sns

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



     print('Libraries imported.')
```

```
Solving environment: done


==> WARNING: A newer version of conda exists. <==
  current version: 4.5.11
  latest version: 4.7.11

Please update conda by running

    $ conda update -n base -c defaults conda
```

```
## Package Plan ##

  environment location: /home/jupyterlab/conda/envs/python

  added / updated specs:
    - geopy


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    geopy-1.20.0               |             py_0          57 KB  conda-forge
    geographiclib-1.49         |             py_0          32 KB  conda-forge
    ------------------------------------------------------------
                                           Total:          90 KB

The following NEW packages will be INSTALLED:

    geographiclib: 1.49-py_0   conda-forge
    geopy:         1.20.0-py_0 conda-forge


Downloading and Extracting Packages
geopy-1.20.0         | 57 KB     |␣
↪########################################## | 100%
geographiclib-1.49   | 32 KB     |␣
↪########################################## | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
Solving environment: done


==> WARNING: A newer version of conda exists. <==
  current version: 4.5.11
  latest version: 4.7.11

Please update conda by running

    $ conda update -n base -c defaults conda



# All requested packages already installed.

Libraries imported.
```

1.2 1. Introduction Section :

1.2.1 1.1 Scenario and Background

I am a data scientist currently residing in Downtown Singapore. I currently live within walking distance to Downtown "Telok Ayer MRT metro station" therefore I have access to good public transportation to work. Likewise, I enjoy many ammenities in the neighborhood , such as international cousine restaurants, cafes, food shops and entertainment. I have been offered a great opportunity to work in Manhattan, NY. Although, I am very excited about it, I am a bit stress toward the process to secure a comparable place to live in Manhattan. Therefore, I decided to apply the learned skills during the Coursera course to explore ways to make sure my decision is factual and rewarding. Of course, there are alternatives to achieve the answer using available Google and Social media tools, but it rewarding doing it myself with learned tools.

1.2.2 1.2 Problem to be resolved:

The challenge to resolve is being able to find a rental apartment unit in Manhattan NY that offers similar characteristics and benefits to my current situation. Therefore, in order to set a basis for comparison, I want to find a renta unit subject to the following conditions:

Apartment with min 2 bedrooms with monthly rent not to exceed US$7000/month Unit located within walking distance (<=1.0 mile, 1.6 km) from a subway metro station in Manhattan Area with ammenities and venues similar to the ones described for current location (See item 2.1)

1.3 2. Data Section

1.3.1 2.1 Data of Current Situation

I Currently reside in the neighborhood of 'Mccallum Street' in Downtonw Singapore. I use Foursquare to identify the venues around the area of residence which are then shown in the Singapore map shown in methodology and execution in section 3.0 . It serves as a reference for comparison with the desired future location in Manhattan NY

1.3.2 2.2 Data Required to resolve the problem

In order to make a good choice of a similar apartment in Manhattan NY, the following data is required: List/Information on neighborhoods form Manhattan with their Geodata (latitud and longitud. List/Information about the subway metro stations in Manhattan with geodata. Listed apartments for rent in Manhattan area with descriptions (how many beds, price, location, address) Venues and ammenities in the Manhattan neighborhoods (e.g. top 10) 2.3 sources and manipulation The list of Manhattan neighborhoods is worked out during LAb exercise during the course. A csv file was created which will be read in order to create a dataframe and its mapping. The csv file 'mh_neigh_data.csv' has the following below data structure. The file will be directly read to the Jupiter Notebook for convenience and space savings. The clustering of neighborhoods and mapping will be shown however. An algorythm was used to determine the geodata from Nominatim . The actual algorythm coding may be shown in 'markdown' mode becasues it takes time to run.

A list of places for rent was collected by web-browsing real estate companies in Manhattan : http://www.rentmanhattan.com/index.cfm?page=search&state=results https://www.nestpick.com/search?city=new-york&page=1&order=relevance&district=manhattan&gclid=CjwK
cPxjZYkURqQEswQK2jKQEpv_MvKcrIhRWRzNkc_r-fGi0lxoCA7cQAvD_BwE&type=apartment&display=list

https://www.realtor.com/apartments/Manhattan_NY A csv file was compiled with the rental place that indicated: areas of Manhattan, address, number of beds, area and monthly rental price. The csv file "nnnn.csv" had the following below structure. An algorythm was used to create all the geodata using Nominatim, as shown in section 3.0. The actual algorythm coding may be shown in 'markdown' mode becasues it takes time to run. With the use of geolocator = Nominatim() , it was possible to determine the latitude and longiude for the subway metro locations as well as for the geodata for each rental place listed. The loop algorythms used are shown in the execution of data in section 3.0 "Great_circle" function from geolocator was used to calculate distances between two points , as in the case to calculate average rent price for units around each subway station and at 1.6 km radius. Foursquare is used to find the avenues at Manhattan neighborhoods in general and a cluster is created to later be able to search for the venues depending of the location shown.

1.3.3 2.4 How the data will be used to solve the problem

The data will be used as follows: Use Foursquare and geopy data to map top 10 venues for all Manhattan neighborhoods and clustered in groups (as per Course LAB) Use foursquare and geopy data to map the location of subway metro stations , separately and on top of the above clustered map in order to be able to identify the venues and ammenities near each metro station, or explore each subway location separately Use Foursquare and geopy data to map the location of rental places, in some form, linked to the subway locations. create a map that depicts, for instance, the average rental price per square ft, around a radious of 1.0 mile (1.6 km) around each subway station - or a similar metrics. I will be able to quickly point to the popups to know the relative price per subway area. Addresses from rental locations will be converted to geodata(lat, long) using Geopy-distance and Nominatim. Data will be searched in open data sources if available, from real estate sites if open to reading, libraries or other government agencies such as Metro New York MTA, etc.

1.3.4 2.5 Mapping of Data

The following maps were created to facilitate the analysis and the choice of the palace to live. Manhattan map of Neighborhoods manhattan subway metro locations Manhattan map of places for rent Manhattan map of clustered venues and neighborhoods Combined maps of Manhattan rent places with subway locations Combined maps of Manhattan rent places with subway locations and venues clusters

1.4 3. Methodology section

This section represents the main component of the report where the data is gathered, prepared for analysis. The tools described are used here and the Notebook cells indicates the execution of steps.

The analysis and the stragegy: The strategy is based on mapping the above described data in section 2.0, in order to facilitate the choice of at least two candidate places for rent. The choice is made based on the demands imposed : location near a subway, rental price and similar venues to Singapore. This visual approach and maps with popups labels allow quick identification of location, price and feature, thus making the selection very easy.

1.5 METHODOLOY EXECUTION - Mapping Data

[3]:
```
# Shenton Way, District 01, Singapore
address = 'Mccallum Street, Singapore'
geolocator = Nominatim()
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude
print('The geograpical coordinate of Singapore home are {}, {}.'.format(latitude, longitude))
```

```
/home/jupyterlab/conda/envs/python/lib/python3.6/site-
packages/ipykernel_launcher.py:3: DeprecationWarning: Using Nominatim with the
default "geopy/1.20.0" `user_agent` is strongly discouraged, as it violates
Nominatim's ToS https://operations.osmfoundation.org/policies/nominatim/ and may
possibly cause 403 and 429 HTTP errors. Please specify a custom `user_agent`
with `Nominatim(user_agent="my-application")` or by overriding the default
`user_agent`: `geopy.geocoders.options.default_user_agent = "my-application"`.
In geopy 2.0 this will become an exception.
  This is separate from the ipykernel package so we can avoid doing imports
until
```

```
The geograpical coordinate of Singapore home are 1.2792423, 103.8481312.
```

[4]:
```
neighborhood_latitude=1.2792655
neighborhood_longitude=103.8480938
```

1.5.1 Dial FourSquare to find venues around current residence in Singapore

[5]:
```
# @hidden_cell
CLIENT_ID = 'DVCxxxxxxxxxxxxxxxxxxC0CFLF1T' # your Foursquare ID
CLIENT_SECRET = '5NWAGyyyyyyyyyyyyyyyyyyyyyyyLFWL1' # your Foursquare Secret
VERSION = '2xxxxxxxxxxxxxxxxxx5' # Foursquare API version

#print('Your credentails:')
#print('CLIENT_ID: ' + CLIENT_ID)
#print('CLIENT_SECRET:' + CLIENT_SECRET)
```

[6]:
```
LIMIT = 100 # limit of number of venues returned by Foursquare API
radius = 500 # define radius
# create URL
url = 'https://api.foursquare.com/v2/venues/explore?
↪&client_id={}&client_secret={}&v={}&ll={},{}&radius={}&limit={}'.format(
    CLIENT_ID,
    CLIENT_SECRET,
    VERSION,
    neighborhood_latitude,
    neighborhood_longitude,
    radius,
    LIMIT)
url # display URL
```

[6]:
```
'https://api.foursquare.com/v2/venues/explore?&client_id=DVCxxxxxxxxxxxxxxxxxxC
0CFLF1T&client_secret=5NWAGyyyyyyyyyyyyyyyyyyyyyyyyLFWL1&v=2xxxxxxxxxxxxxxxxx5&
ll=1.2792655,103.8480938&radius=500&limit=100'
```

[13]:
```
# results display is hidden for report simplification
results = requests.get(url).json()
#results
```

[14]:
```
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']
```

[12]:
```
venues = results['response']['groups'][0]['items']
SGnearby_venues = json_normalize(venues) # flatten JSON
# filter columns
filtered_columns = ['venue.name', 'venue.categories', 'venue.location.lat', 'venue.location.lng']
SGnearby_venues =SGnearby_venues.loc[:, filtered_columns]
# filter the category for each row
SGnearby_venues['venue.categories'] = SGnearby_venues.apply(get_category_type, axis=1)
# clean columns
SGnearby_venues.columns = [col.split(".")[-1] for col in SGnearby_venues.columns]

SGnearby_venues.shape
```

```
    ---------------------------------------------------------------------------

    KeyError                                  Traceback (most recent call last)

    <ipython-input-12-a945df6402f0> in <module>
  ----> 1 venues = results['response']['groups'][0]['items']
        2 SGnearby_venues = json_normalize(venues) # flatten JSON
        3 # filter columns
        4 filtered_columns = ['venue.name', 'venue.categories', 'venue.location.lat', 'venue.location.
↪lng']
        5 SGnearby_venues =SGnearby_venues.loc[:, filtered_columns]
```

```
KeyError: 'groups'
```

[10]:
```python
# Venues near current Singapore residence place
SGnearby_venues.head(10)
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-10-cacc63616b4e> in <module>
  1 # Venues near current Singapore residence place
----> 2 SGnearby_venues.head(10)


NameError: name 'SGnearby_venues' is not defined
```

[15]:
```python
latitude=1.2792655
longitude=103.8480938
# create map of Singapore place  using latitude and longitude values
map_sg = folium.Map(location=[latitude, longitude], zoom_start=18)
# add markers to map
for lat, lng, label in zip(SGnearby_venues['lat'], SGnearby_venues['lng'],␣
 ↪SGnearby_venues['name']):
    label = folium.Popup(label, parse_html=True)
    folium.RegularPolygonMarker(
        [lat, lng],
        number_of_sides=30,
        radius=7,
        popup=label,
        color='blue',
        fill_color='#0f0f0f',
        fill_opacity=0.6,
    ).add_to(map_sg)

map_sg
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-15-ad221d2a18f2> in <module>
  4 map_sg = folium.Map(location=[latitude, longitude], zoom_start=18)
  5 # add markers to map
```

```
----> 6 for lat, lng, label in zip(SGnearby_venues['lat'], SGnearby_venues['lng'],␣
↪SGnearby_venues['name']):
      7     label = folium.Popup(label, parse_html=True)
      8     folium.RegularPolygonMarker(


    NameError: name 'SGnearby_venues' is not defined
```

1.6 MANHATTAN NEIGHBORHOODS - DATA AND MAPPING

[16]:
```
# Read csv file with clustered neighborhoods with geodata
manhattan_data  = pd.read_csv('mh_neigh_data.csv')
manhattan_data.head()
```

```
    ---------------------------------------------------------------------------

    FileNotFoundError                         Traceback (most recent call last)

    <ipython-input-16-c44f552da870> in <module>
      1 # Read csv file with clustered neighborhoods with geodata
----> 2 manhattan_data  = pd.read_csv('mh_neigh_data.csv')
      3 manhattan_data.head()


    ~/conda/envs/python/lib/python3.6/site-packages/pandas/io/parsers.py in␣
↪parser_f(filepath_or_buffer, sep, delimiter, header, names, index_col, usecols, squeeze,␣
↪prefix, mangle_dupe_cols, dtype, engine, converters, true_values, false_values,␣
↪skipinitialspace, skiprows, skipfooter, nrows, na_values, keep_default_na, na_filter, verbose,␣
↪skip_blank_lines, parse_dates, infer_datetime_format, keep_date_col, date_parser,␣
↪dayfirst, cache_dates, iterator, chunksize, compression, thousands, decimal, lineterminator,␣
↪quotechar, quoting, doublequote, escapechar, comment, encoding, dialect, error_bad_lines,␣
↪warn_bad_lines, delim_whitespace, low_memory, memory_map, float_precision)
    683         )
    684
--> 685         return _read(filepath_or_buffer, kwds)
    686
    687     parser_f.__name__ = name


    ~/conda/envs/python/lib/python3.6/site-packages/pandas/io/parsers.py in␣
↪_read(filepath_or_buffer, kwds)
    455
    456     # Create the parser.
--> 457     parser = TextFileReader(fp_or_buf, **kwds)
    458
    459     if chunksize or iterator:


    ~/conda/envs/python/lib/python3.6/site-packages/pandas/io/parsers.py in __init__(self,␣
↪f, engine, **kwds)
    893             self.options["has_index_names"] = kwds["has_index_names"]
    894
--> 895         self._make_engine(self.engine)
    896
    897     def close(self):


    ~/conda/envs/python/lib/python3.6/site-packages/pandas/io/parsers.py in␣
↪_make_engine(self, engine)
   1133     def _make_engine(self, engine="c"):
   1134         if engine == "c":
-> 1135             self._engine = CParserWrapper(self.f, **self.options)
   1136         else:
   1137             if engine == "python":


    ~/conda/envs/python/lib/python3.6/site-packages/pandas/io/parsers.py in __init__(self,␣
↪src, **kwds)
   1904         kwds["usecols"] = self.usecols
   1905
-> 1906         self._reader = parsers.TextReader(src, **kwds)
   1907         self.unnamed_cols = self._reader.unnamed_cols
   1908


    pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader.__cinit__()


    pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader._setup_parser_source()


    FileNotFoundError: [Errno 2] File b'mh_neigh_data.csv' does not exist: b'mh_neigh_data.
↪csv'
```

[ ]:
```
manhattan_data.tail()
```

[ ]:
```
manhattan_merged = pd.read_csv('manhattan_merged.csv')
manhattan_merged.head()
```

[ ]:
```
# create map of Manhattan using latitude and longitude values from Nominatim
latitude= 40.7308619
longitude= -73.9871558
```

```
kclusters=5
map_clusters = folium.Map(location=[latitude, longitude], zoom_start=13)

# 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(manhattan_merged['Latitude'], manhattan_merged['Longitude'],␣
  ↪manhattan_merged['Neighborhood'], manhattan_merged['Cluster Labels']):
    label = folium.Popup(str(poi) + ' Cluster ' + str(cluster), parse_html=True)
    folium.CircleMarker(
        [lat, lon],
        radius=20,
        popup=label,
        color=rainbow[cluster-1],
        fill=True,
        fill_color=rainbow[cluster-1],
        fill_opacity=0.7).add_to(map_clusters)
  # add markers for rental places to map
for lat, lng, label in zip(manhattan_data['Latitude'], manhattan_data['Longitude'],␣
  ↪manhattan_data['Neighborhood']):
    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_clusters)



map_clusters
```

[ ]:
```
## kk is the cluster number to explore
kk = 2
manhattan_merged.loc[manhattan_merged['Cluster Labels'] == kk, manhattan_merged.
  ↪columns[[1] + list(range(5, manhattan_merged.shape[1]))]]
```

1.6.1 Map of Manhattan places for rent

```
[ ]: # csv files with rental places with basic data but still wihtout geodata ( latitude and longitude)
     # pd.read_csv(' le.csv', header=None, nrows=5)
     mh_rent=pd.read_csv('MH_flats_price.csv')
     mh_rent.head()
```

```
[ ]: mh_rent.tail()
```

```
[ ]: mh_rent=pd.read_csv('MH_rent_latlong.csv')
     mh_rent.head()
```

```
[ ]: mh_rent.tail()
```

1.6.2 Manhattan apartment rent price statistics

```
[ ]: import seaborn as sns
     sns.distplot(mh_rent['Rent_Price'],bins=15)
```

```
[ ]: import seaborn as sns
     sns.distplot(mh_rent['Price_per_ft2'],bins=15)
```

```
[ ]: sns.boxplot(x='Rooms', y= 'Rent_Price', data=mh_rent)
```

```
[ ]:  create map of Manhattan using latitude and longitude values from Nominatim
     latitude= 40.7308619
     longitude= -73.9871558

     map_manhattan_rent = folium.Map(location=[latitude, longitude], zoom_start=12.5)

     # add markers to map
     for lat, lng, label in zip(mh_rent['Lat'], mh_rent['Long'],'$ ' + mh_rent['Rent_Price'].
      ↪astype(str)+ ',  '+ mh_rent['Address']):
         label = folium.Popup(label, parse_html=True)
         folium.CircleMarker(
             [lat, lng],
             radius=6,
             popup=label,
             color='blue',
             fill=True,
             fill_color='#3186cc',
             fill_opacity=0.7,
             parse_html=False).add_to(map_manhattan_rent)


     map_manhattan_rent
```

1.6.3 Map of Manhattan showing the places for rent and the cluster of venues

[ ]:
```
# create map of Manhattan using latitude and longitude values from Nominatim
latitude= 40.7308619
longitude= -73.9871558

# create map with clusters
kclusters=5
map_clusters2 = folium.Map(location=[latitude, longitude], zoom_start=13)

# 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(manhattan_merged['Latitude'], manhattan_merged['Longitude'],␣
  ↪manhattan_merged['Neighborhood'], manhattan_merged['Cluster Labels']):
    label = folium.Popup(str(poi) + ' Cluster ' + str(cluster), parse_html=True)
    folium.CircleMarker(
        [lat, lon],
        radius=20,
        popup=label,
        color=rainbow[cluster-1],
        fill=True,
        fill_color=rainbow[cluster-1],
        fill_opacity=0.7).add_to(map_clusters2)
    # add markers to map for rental places
for lat, lng, label in zip(mh_rent['Lat'], mh_rent['Long'],'$ ' + mh_rent['Rent_Price'].
  ↪astype(str)+ mh_rent['Address']):
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius=6,
        popup=label,
        color='blue',
        fill=True,
        fill_color='#3186cc',
        fill_opacity=0.7,
        parse_html=False).add_to(map_clusters2)

    # Adds tool to the top right
from folium.plugins import MeasureControl
map_manhattan_rent.add_child(MeasureControl())
```

```
# FMeasurement ruler icon to establish distnces on map
from folium.plugins import FloatImage
url = ('https://media.licdn.com/mpr/mpr/shrinknp_100_100/
  ↪AAEAAQAAAAAAAAlgAAAAJGE3OTA4YTdlLTkzZjUtNDFjYy1iZThlLWQ5OTNkYzlhNzM4OQ.
  ↪jpg')
FloatImage(url, bottom=5, left=85).add_to(map_manhattan_rent)
map_clusters2
```

```
[ ]: ## kk is the cluster number to explore
kk = 3
manhattan_merged.loc[manhattan_merged['Cluster Labels'] == kk, manhattan_merged.
  ↪columns[[1] + list(range(5, manhattan_merged.shape[1]))]]
```

```
[ ]: # A csv file summarized the subway station and the addresses for next step to determine geodata
mh=pd.read_csv('NYC_subway_list.csv')
mh.head()
```

```
[ ]: #  Add columns 'lat'  and 'long' to mh dataframe - with random temporary numbers to get␣
  ↪started
sLength = len(mh['sub_station'])
lat = pd.Series(np.random.randn(sLength))
long=pd.Series(np.random.randn(sLength))
mh = mh.assign(lat=lat.values)
mh = mh.assign(long=long.values)
```

```
[ ]: mh=pd.read_csv('MH_subway.csv')
print(mh.shape)
mh.head()
```

```
[ ]: # removing duplicate rows and creating new set mhsub1
mhsub1=mh.drop_duplicates(subset=['lat','long'], keep="last").reset_index(drop=True)
mhsub1.shape
```

```
[ ]: mhsub1.tail()
```

1.6.4 MAP of Manhattan showing the location of subway stations

```
[ ]: # map subway stations
# create map of Manhattan using latitude and longitude values obtain previoulsy via␣
  ↪Moninatim geolocator
latitude=40.7308619
longitude=-73.9871558

map_mhsub1 = folium.Map(location=[latitude, longitude], zoom_start=12)

# add markers of subway locations to map
for lat, lng, label in zip(mhsub1['lat'], mhsub1['long'],  mhsub1['sub_station'].astype(str) ):
    label = folium.Popup(label, parse_html=True)
    folium.RegularPolygonMarker(
        [lat, lng],
        number_of_sides=6,
        radius=6,
        popup=label,
        color='red',
        fill_color='red',
        fill_opacity=2.5,
    ).add_to(map_mhsub1)
map_mhsub1
```

[ ]:
```
mh_rent.head()
```

[ ]:
```
# create map of Manhattan using latitude and longitude values from Nominatim
latitude= 40.7308619
longitude= -73.9871558

map_manhattan_rent = folium.Map(location=[latitude, longitude], zoom_start=13.3)

# add markers to map
for lat, lng, label in zip(mh_rent['Lat'], mh_rent['Long'],'$ ' + mh_rent['Rent_Price'].
  ↪astype(str)+ mh_rent['Address']):
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius=6,
        popup=label,
        color='blue',
        fill=True,
        fill_color='#3186cc',
        fill_opacity=0.7,
        parse_html=False).add_to(map_manhattan_rent)

    # add markers of subway locations to map
for lat, lng, label in zip(mhsub1['lat'], mhsub1['long'],  mhsub1['sub_station'].astype(str) ):
    label = folium.Popup(label, parse_html=True)
    folium.RegularPolygonMarker(
        [lat, lng],
        number_of_sides=6,
        radius=6,
        popup=label,
        color='red',
        fill_color='red',
        fill_opacity=2.5,
         ).add_to(map_manhattan_rent)

    # Adds tool to the top right
from folium.plugins import MeasureControl
map_manhattan_rent.add_child(MeasureControl())
```

```
# Measurement ruler icon tool to measure distances in map
from folium.plugins import FloatImage
url = ('https://media.licdn.com/mpr/mpr/shrinknp_100_100/
 ↪AAEAAQAAAAAAAAlgAAAAJGE3OTA4YTdlLTkzZjUtNDFjYy1iZThlLWQ5OTNkYzlhNzM4OQ.
 ↪jpg')
FloatImage(url, bottom=5, left=85).add_to(map_manhattan_rent)

map_manhattan_rent
```

1.7 4.0 Results

```
[ ]: # create map of Manhattan using latitude and longitude values from Nominatim
latitude= 40.7308619
longitude= -73.9871558

map_mh_one = folium.Map(location=[latitude, longitude], zoom_start=13.3)

# add markers to map
for lat, lng, label in zip(mh_rent['Lat'], mh_rent['Long'],'$ ' + mh_rent['Rent_Price'].
 ↪astype(str)+ ', '+mh_rent['Address']):
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius=6,
        popup=label,
        color='blue',
        fill=True,
        fill_color='#3186cc',
        fill_opacity=0.7,
        parse_html=False).add_to(map_mh_one)

    # add markers of subway locations to map
for lat, lng, label in zip(mhsub1['lat'], mhsub1['long'],  mhsub1['sub_station'].astype(str) ):
    label = folium.Popup(label, parse_html=True)
    folium.RegularPolygonMarker(
        [lat, lng],
        number_of_sides=6,
        radius=6,
        popup=label,
        color='red',
        fill_color='red',
        fill_opacity=2.5,
         ).add_to(map_mh_one)
```

```
# set color scheme for the clusters
kclusters=5
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(manhattan_merged['Latitude'], manhattan_merged['Longitude'],␣
 ↪manhattan_merged['Neighborhood'], manhattan_merged['Cluster Labels']):
    label = folium.Popup(str(poi) + ' Cluster ' + str(cluster), parse_html=True)
    folium.CircleMarker(
        [lat, lon],
        radius=15,
        popup=label,
        color=rainbow[cluster-1],
        fill=True,
        fill_color=rainbow[cluster-1],
        fill_opacity=0.7).add_to(map_mh_one)
     # Adds tool to the top right
from folium.plugins import MeasureControl
map_mh_one.add_child(MeasureControl())

# Measurement ruler icon tool to measure distances in map
from folium.plugins import FloatImage
url = ('https://media.licdn.com/mpr/mpr/shrinknp_100_100/
 ↪AAEAAQAAAAAAAAlgAAAAJGE3OTA4YTdlLTkzZjUtNDFjYy1iZThlLWQ5OTNkYzlhNzM4OQ.
 ↪jpg')
FloatImage(url, bottom=5, left=85).add_to(map_mh_one)

map_mh_one
```

1.7.1 Problem Resolution - Select the apartment for rentű

The above consolidate map was used to explore options. After examining, I have chosen two locations that meet the requirements which will assess to make a choice. Apartment 1: 305 East 63rd Street in the Sutton Place Neighborhood and near 'subway 59th Street' station, Cluster # 2 Monthly rent : 7500 Dollars

Apartment 2: 19 Dutch Street in the Financial District Neighborhood and near 'Fulton Street Subway' station, Cluster # 3 Monthly rent : 6935 Dollars

```
[ ]: ## kk is the cluster number to explore
kk = 2
manhattan_merged.loc[manhattan_merged['Cluster Labels'] == kk, manhattan_merged.
 ↪columns[[1] + list(range(5, manhattan_merged.shape[1]))]]
```

[ ]:
```
## kk is the cluster number to explore
kk = 3
manhattan_merged.loc[manhattan_merged['Cluster Labels'] == kk, manhattan_merged.
 ↪columns[[1] + list(range(5, manhattan_merged.shape[1]))]]
```

1.8 5.0 DISCUSSION

In general, I am positively impressed with the overall organization, content and lab works presented during the Coursera IBM Certification Course I feel this Capstone project presented me a great opportunity to practice and apply the Data Science tools and methodologies learned. I have created a good project that I can present as an example to show my potential. I feel I have acquired a good starting point to become a professional Data Scientist and I will continue exploring to creating examples of practical cases.ű

1.9 6.0 CONCLUSIONS

I feel rewarded with the efforts, time and money spent. I believe this course with all the topics covered is well worthy of appreciation. This project has shown me a practical application to resolve a real situation that has impacting personal and financial impact using Data Science tools. The mapping with Folium is a very powerful technique to consolidate information and make the analysis and decision thoroughly and with confidence. I would recommend for use in similar situations. One must keep abreast of new tools for DS that continue to appear for application in several business fields.

[ ]:
```

```