IBM Coursera Capstone Project

The Battle of Neighborhoods (Week 1)

Part 1 : Introduction and Data Sections

In [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
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 c
ompleted the Foursquare API lab
import folium # map rendering library

print('Libraries imported.')
```

```
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

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

  added / updated specs:
    - geopy


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    geographiclib-1.50         |             py_0          34 KB  conda-fo
rge
    geopy-1.22.0               |     pyh9f0ad1d_0          63 KB  conda-fo
rge
    ------------------------------------------------------------
                                          Total:          97 KB

The following NEW packages will be INSTALLED:

  geographiclib      conda-forge/noarch::geographiclib-1.50-py_0
  geopy              conda-forge/noarch::geopy-1.22.0-pyh9f0ad1d_0



Downloading and Extracting Packages
geopy-1.22.0         | 63 KB     | ##################################### |
100%
geographiclib-1.50   | 34 KB     | ##################################### |
100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
Collecting package metadata (current_repodata.json): done
Solving environment: failed with initial frozen solve. Retrying with flexi
ble solve.
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

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

  added / updated specs:
    - folium=0.5.0


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    altair-4.1.0               |             py_1         614 KB  conda-fo
rge
    branca-0.4.1               |             py_0          26 KB  conda-fo
rge
    brotlipy-0.7.0             |py36h8c4c3a4_1000         346 KB  conda-fo
rge
```

```
    chardet-3.0.4              |py36h9f0ad1d_1006         188 KB  conda-fo
rge
    cryptography-2.9.2         |   py36h45558ae_0         613 KB  conda-fo
rge
    folium-0.5.0               |             py_0          45 KB  conda-fo
rge
    pandas-1.0.3               |   py36h830a2c2_1        11.1 MB  conda-fo
rge
    pysocks-1.7.1              |   py36h9f0ad1d_1          27 KB  conda-fo
rge
    toolz-0.10.0               |             py_0          46 KB  conda-fo
rge
    vincent-0.4.4              |             py_1          28 KB  conda-fo
rge
    ------------------------------------------------------------
                                           Total:        13.0 MB

The following NEW packages will be INSTALLED:

  altair             conda-forge/noarch::altair-4.1.0-py_1
  attrs              conda-forge/noarch::attrs-19.3.0-py_0
  branca             conda-forge/noarch::branca-0.4.1-py_0
  brotlipy           conda-forge/linux-64::brotlipy-0.7.0-py36h8c4c3a4_100
0
  chardet            conda-forge/linux-64::chardet-3.0.4-py36h9f0ad1d_1006
  cryptography       conda-forge/linux-64::cryptography-2.9.2-py36h45558ae
_0
  entrypoints        conda-forge/linux-64::entrypoints-0.3-py36h9f0ad1d_10
01
  folium             conda-forge/noarch::folium-0.5.0-py_0
  idna               conda-forge/noarch::idna-2.9-py_1
  importlib_metadata conda-forge/noarch::importlib_metadata-1.6.0-0
  jinja2             conda-forge/noarch::jinja2-2.11.2-pyh9f0ad1d_0
  jsonschema         conda-forge/linux-64::jsonschema-3.2.0-py36h9f0ad1d_1
  markupsafe         conda-forge/linux-64::markupsafe-1.1.1-py36h8c4c3a4_1
  pandas             conda-forge/linux-64::pandas-1.0.3-py36h830a2c2_1
  pyopenssl          conda-forge/noarch::pyopenssl-19.1.0-py_1
  pyrsistent         conda-forge/linux-64::pyrsistent-0.16.0-py36h8c4c3a4_
0
  pysocks            conda-forge/linux-64::pysocks-1.7.1-py36h9f0ad1d_1
  pytz               conda-forge/noarch::pytz-2020.1-pyh9f0ad1d_0
  requests           conda-forge/noarch::requests-2.23.0-pyh8c360ce_2
  toolz              conda-forge/noarch::toolz-0.10.0-py_0
  urllib3            conda-forge/noarch::urllib3-1.25.9-py_0
  vincent            conda-forge/noarch::vincent-0.4.4-py_1


Downloading and Extracting Packages
pysocks-1.7.1        | 27 KB     | ##################################### |
100%
toolz-0.10.0         | 46 KB     | ##################################### |
100%
chardet-3.0.4        | 188 KB    | ##################################### |
100%
folium-0.5.0         | 45 KB     | ##################################### |
100%
branca-0.4.1         | 26 KB     | ##################################### |
100%
cryptography-2.9.2   | 613 KB    | ##################################### |
100%
```

```
brotlipy-0.7.0       | 346 KB    | ##################################### |
100%
pandas-1.0.3         | 11.1 MB   | ##################################### |
100%
altair-4.1.0         | 614 KB    | ##################################### |
100%
vincent-0.4.4        | 28 KB     | ##################################### |
100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
Libraries imported.
```

REPORT CONTENT

1. Introduction: **-** Description of the business problem and the interested audience of this project.
2. Data: Description of the data that will be used to solve the problem and the sources.
3. Methodology: Discussion and description of exploratory data analysis carried out, any inferential statistical testing performed, and if any machine learnings were used establishing the strategy and purposes.
4. Results: Discussion of the results.
5. Discussion: Elaboration and discussion on any observations noted and any recommendations suggested based on the results.
6. Conclusion:

1. Introduction

Scenario:

I am a data scientist residing in Islamabad, Pakistan. I am very excited and I want to use this opportunity to practice my learnings in Coursera in order to answer relevant questions. The key question is : To find a convenient and enjoyable place in Islamabad? Certainly, I can use available real estate apps and Google but the idea is to use and apply myself the learn tools during the course. In order to make a comparison and evaluation of the rental options in Manhattan NY, I must set some basis, therefore the apartment in Manhattan must meet the following demands:

- Apartment must be 2 or 3 bedrooms
- Desired location is near a metro station in the Manhattan area and within 1.0 mile (1.6 km) radius
- Price of rent not exceed $7,000 per month
- Top ammenities in the selected neighborhood.
- Desirable to have venues such as coffee shops, restaurants Asian Thai, gym and food shops.
- As a reference, I have included a map of venues in Islamabad.

Business Problem:

The challenge is to find a suitable apartment for rent in Manhattan NY that complies with the demands on location, price and venues. The data required to resolve this challenge is described in the following section 2, below.

Interested Audience

I believe this is a relevant challenge with valid questions for anyone moving to other large city in US, EU or Asia. The same methodology can be applied in accordance to demands as applicable. This case is also applicable for anyone interested in exploring starting or locating a new business in any city. Lastly, it can also serve as a good practical exercise to develop Data Science skills.

2. Data

The following data is required to answer the issues of the problem:

- List of Boroughs and neighborhoods of Manhattan with their geodata (latitud and longitude)
- List of Subway metro stations in Manhattan with their address location.
- List of apartments for rent in Manhattan area with their addresses and price.
- Preferably, a list of apartment for rent with additional information, such as price, address, area, # of beds, etc.
- Venues for each Manhattan neighborhood.
- Venues for subway metro stations, as needed.

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.
- 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.
- 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.

The procesing of these DATA will allow to answer the key questions to make a decision:

- what is the cost of rent (per square ft) around a mile radius from each subway metro station?
- what is the area of Manhattan with best rental pricing that meets criteria established?
- What is the distance from work place (Park Ave and 53 rd St) and the tentative future home?
- What are the venues of the two best places to live? How the prices compare?
- How venues distribute among Manhattan neighborhoods and around metro stations?
- Are there tradeoffs between size and price and location?
- Any other interesting statistical data findings of the real estate and overall data.

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 strategy is based on mapping the above described data in section 2.0, in order to facilitate the choice of a candidate places for accomodation. The choice is made based on the demands imposed : similar venues to Islamabad, Pakistan. This visual approach and maps with popups labels allow quick identification of location, thus making the selection very easy. The procesing of these DATA and its mapping will allow to answer the key questions to make a decision:

- What are the venues of the best place to live?
- How venues distribute among Manhattan neighborhoods ?

In [3]:

```
#Reference of venues in Islamabad for comparison to Manhattan place
# Shenton Way, District 01, Islamabad
address = 'Islamabad, Pakistan'
geolocator = Nominatim()
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude
print('The geograpical coordinate of Islamabad are {}, {}.'.format(latitude, longitude
))
```

```
/home/jupyterlab/conda/envs/python/lib/python3.6/site-packages/ipykernel_l
auncher.py:4: DeprecationWarning: Using Nominatim with the default "geopy/
1.22.0" `user_agent` is strongly discouraged, as it violates Nominatim's T
oS https://operations.osmfoundation.org/policies/nominatim/ and may possib
ly cause 403 and 429 HTTP errors. Please specify a custom `user_agent` wit
h `Nominatim(user_agent="my-application")` or by overriding the default `u
ser_agent`: `geopy.geocoders.options.default_user_agent = "my-applicatio
n"`. In geopy 2.0 this will become an exception.
  after removing the cwd from sys.path.

The geograpical coordinate of Islamabad are 33.6938118, 73.0651511.
```

In [4]:

```
neighborhood_latitude=33.6938118
neighborhood_longitude=73.0651511
```

In [5]:

```
# @hidden_cell
CLIENT_ID = 'B5XQRT4TUVOOMGECQMCDFM0RPW5TZH1SCN0SJG1ISBXGWC23' # your Foursquare ID
CLIENT_SECRET = 'NAARKXSPGQQPFACUACVIYUF0HGDDZLFDCAJYLBD4T2LIOEI5' # your Foursquare Se
cret
VERSION = '20200101' # Foursquare API version

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

```
Your credentails:
CLIENT_ID: B5XQRT4TUVOOMGECQMCDFM0RPW5TZH1SCN0SJG1ISBXGWC23
CLIENT_SECRET:NAARKXSPGQQPFACUACVIYUF0HGDDZLFDCAJYLBD4T2LIOEI5
```

In [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
```

Out[6]:

```
'https://api.foursquare.com/v2/venues/explore?&client_id=B5XQRT4TUVOOMGECQ
MCDFM0RPW5TZH1SCN0SJG1ISBXGWC23&client_secret=NAARKXSPGQQPFACUACVIYUF0HGDD
ZLFDCAJYLBD4T2LIOEI5&v=20200101&ll=33.6938118,73.0651511&radius=500&limit=
100'
```

In [7]:

```
results = requests.get(url).json()
#results
```

In [8]:

```
# 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 [9]:

```
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.loca
tion.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.head(10)
```

```
/home/jupyterlab/conda/envs/python/lib/python3.6/site-packages/ipykernel_l
auncher.py:3: FutureWarning: pandas.io.json.json_normalize is deprecated,
use pandas.json_normalize instead
  This is separate from the ipykernel package so we can avoid doing import
s until
```

Out[9]:

	name	categories	lat	lng
0	Pakistan Monument	History Museum	33.693070	73.068910
1	Munchies	Fast Food Restaurant	33.694545	73.067365
2	Prestige Garage Door	Home Service	33.692489	73.068245
3	Sonia ASAP Locksmith	Locksmith	33.690541	73.067665

In [10]:

```
# create map of Islamabad place  using latitude and longitude values
map_isb = 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_ven
ues['name']):
    label = folium.Popup(label, parse_html=True)
    folium.RegularPolygonMarker(
        [lat, lng],
        number_of_sides=4,
        radius=8,
        popup=label,
        color='blue',
        fill_color='#0f0f0f',
        fill_opacity=0.7,
    ).add_to(map_isb)

map_isb
```

Out[10]:

Make this Notebook Trusted to load map: File -> Trust Notebook

MANHATTAN NEIGHBORHOODS - DATA AND MAPPING

Cluster neighborhood data was produced with Foursquare during course lab work. A csv file was produced containing the neighborhoods around the 40 Boroughs. Now, the csv file is just read for convenience and consolidation of report.

In [11]:

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

Out[11]:

	Borough	Neighborhood	Latitude	Longitude	Cluster Labels
0	Manhattan	Marble Hill	40.876551	-73.910660	2
1	Manhattan	Chinatown	40.715618	-73.994279	2
2	Manhattan	Washington Heights	40.851903	-73.936900	4
3	Manhattan	Inwood	40.867684	-73.921210	3
4	Manhattan	Hamilton Heights	40.823604	-73.949688	0

In [12]:

```
manhattan_data.tail()
```

Out[12]:

	Borough	Neighborhood	Latitude	Longitude	Cluster Labels
35	Manhattan	Turtle Bay	40.752042	-73.967708	3
36	Manhattan	Tudor City	40.746917	-73.971219	3
37	Manhattan	Stuyvesant Town	40.731000	-73.974052	4
38	Manhattan	Flatiron	40.739673	-73.990947	3
39	Manhattan	Hudson Yards	40.756658	-74.000111	2

Manhattan Borough neighborhoods - data with top 10 clustered venues

In [13]:

```
manhattan_merged = pd.read_csv('manhattan_merged.csv')
manhattan_merged.head()
```

Out[13]:

	Borough	Neighborhood	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue
0	Manhattan	Marble Hill	40.876551	-73.910660	2	Coffee Shop	Discount Store	Yoga Studio
1	Manhattan	Chinatown	40.715618	-73.994279	2	Chinese Restaurant	Cocktail Bar	Dim Sum Restaurant
2	Manhattan	Washington Heights	40.851903	-73.936900	4	Café	Bakery	Mobile Phone Shop
3	Manhattan	Inwood	40.867684	-73.921210	3	Mexican Restaurant	Lounge	Pizza Place
4	Manhattan	Hamilton Heights	40.823604	-73.949688	0	Mexican Restaurant	Coffee Shop	Café

Map of Manhattan neighborhoods with top 10 clustered venues

Popus allow to identify each neighborhood and the cluster of venues around it in order to proceed to examine in more detail in the next cell

In [14]:

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

#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['Longi
tude'], 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'], man
hattan_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
```

Out[14]:

Make this Notebook Trusted to load map: File -> Trust Notebook

Examine a paticular Cluster - print venues

After examining several cluster data , I concluded that cluster # 2 resembles closer the Islamabad place, therefore providing guidance as to where to look for the future apartment

In [15]:

```
## 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]))]]
```

Out[15]:

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7 C
0	Marble Hill	Coffee Shop	Discount Store	Yoga Studio	Steakhouse	Supplement Shop	Tennis Stadium	
1	Chinatown	Chinese Restaurant	Cocktail Bar	Dim Sum Restaurant	American Restaurant	Vietnamese Restaurant	Salon / Barbershop	
6	Central Harlem	African Restaurant	Seafood Restaurant	French Restaurant	American Restaurant	Cosmetics Shop	Chinese Restaurant	
9	Yorkville	Coffee Shop	Gym	Bar	Italian Restaurant	Sushi Restaurant	Pizza Place	Re
14	Clinton	Theater	Italian Restaurant	Coffee Shop	American Restaurant	Gym / Fitness Center	Hotel	Wi
23	Soho	Clothing Store	Boutique	Women's Store	Shoe Store	Men's Store	Furniture / Home Store	Re
26	Morningside Heights	Coffee Shop	American Restaurant	Park	Bookstore	Pizza Place	Sandwich Place	
34	Sutton Place	Gym / Fitness Center	Italian Restaurant	Furniture / Home Store	Indian Restaurant	Dessert Shop	American Restaurant	
39	Hudson Yards	Coffee Shop	Italian Restaurant	Hotel	Theater	American Restaurant	Café	

Map of Manhattan places for rent

Several Manhattan real estate webs were webscrapped to collect rental data, as mentioned in section 2.0 . The result was summarized in a csv file for direct reading, in order to consolidate the process. The initial data for 144 apartment did not have the latitude and longitude data (NaN) but the information was established in the following cell using an algorythm and Nominatim.

In [16]:

```
# 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()
```

Out[16]:

	Address	Area	Price_per_ft2	Rooms	Area-ft2	Rent_Price	Lat	Long
0	West 105th Street	Upper West Side	2.94	5.0	3400	10000	NaN	NaN
1	East 97th Street	Upper East Side	3.57	3.0	2100	7500	NaN	NaN
2	West 105th Street	Upper West Side	1.89	4.0	2800	5300	NaN	NaN
3	CARMINE ST.	West Village	3.03	2.0	1650	5000	NaN	NaN
4	171 W 23RD ST.	Chelsea	3.45	2.0	1450	5000	NaN	NaN

In [17]:

```
mh_rent.tail()
```

Out[17]:

	Address	Area	Price_per_ft2	Rooms	Area-ft2	Rent_Price	Lat	Long
139	200 East 72nd Street	Rental in Lenox Hill	5.15	3.0	1700	8750	NaN	NaN
140	50 Murray Street	No fee rental in Tribeca	7.11	2.0	1223	8700	NaN	NaN
141	300 East 56th Street	No fee rental in Midtown East	3.87	3.0	2100	8118	NaN	NaN
142	1930 Broadway	No fee rental in Central Park West	5.06	2.0	1600	8095	NaN	NaN
143	33 West 9th Street	Rental in Greenwich Village	6.67	2.0	1500	10000	NaN	NaN

Obtain geodata (lat,long) for each rental place in Manhattan with Nominatim

Data was stored in a csv file for simplifaction report purposes and saving code processing time in future.

In [18]:

```
## This section may be 'markedown' for the report because its execution takes few minut
es .
## Terefore, the csv previusly made may be just read directly.

for n in range(len(mh_rent)):
    address= mh_rent['Address'][n]
    address=(mh_rent['Address'][n]+ '  , '+' Manhattan NY ')
    geolocator = Nominatim()
    location = geolocator.geocode(address)
    latitude = location.latitude
    longitude = location.longitude
    mh_rent['Lat'][n]=latitude
    mh_rent['Long'][n]=longitude
    #print(n,latitude,longitude)
    time.sleep(2)

print('Geodata completed')
# save dataframe to csv file
mh_rent.to_csv('MH_rent_latlong.csv',index=False)
mh_rent.shape
```

```
/home/jupyterlab/conda/envs/python/lib/python3.6/site-packages/ipykernel_l
auncher.py:7: DeprecationWarning: Using Nominatim with the default "geopy/
1.22.0" `user_agent` is strongly discouraged, as it violates Nominatim's T
oS https://operations.osmfoundation.org/policies/nominatim/ and may possib
ly cause 403 and 429 HTTP errors. Please specify a custom `user_agent` wit
h `Nominatim(user_agent="my-application")` or by overriding the default `u
ser_agent`: `geopy.geocoders.options.default_user_agent = "my-applicatio
n"`. In geopy 2.0 this will become an exception.
  import sys
/home/jupyterlab/conda/envs/python/lib/python3.6/site-packages/ipykernel_l
auncher.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  # This is added back by InteractiveShellApp.init_path()
/home/jupyterlab/conda/envs/python/lib/python3.6/site-packages/ipykernel_l
auncher.py:12: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  if sys.path[0] == '':

Geodata completed
```

Out[18]:

```
(144, 8)
```

In [19]:

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

Out[19]:

	Address	Area	Price_per_ft2	Rooms	Area-ft2	Rent_Price	Lat	Long
0	West 105th Street	Upper West Side	2.94	5.0	3400	10000	40.799771	-73.966213
1	East 97th Street	Upper East Side	3.57	3.0	2100	7500	40.788517	-73.955118
2	West 105th Street	Upper West Side	1.89	4.0	2800	5300	40.799771	-73.966213
3	CARMINE ST.	West Village	3.03	2.0	1650	5000	40.730337	-74.002476
4	171 W 23RD ST.	Chelsea	3.45	2.0	1450	5000	40.744118	-73.995299

In [20]:

```
mh_rent.tail()
```

Out[20]:

	Address	Area	Price_per_ft2	Rooms	Area-ft2	Rent_Price	Lat	Long
139	200 East 72nd Street	Rental in Lenox Hill	5.15	3.0	1700	8750	40.769465	-73.960339
140	50 Murray Street	No fee rental in Tribeca	7.11	2.0	1223	8700	40.714051	-74.009608
141	300 East 56th Street	No fee rental in Midtown East	3.87	3.0	2100	8118	40.758216	-73.965190
142	1930 Broadway	No fee rental in Central Park West	5.06	2.0	1600	8095	40.772433	-73.981705
143	33 West 9th Street	Rental in Greenwich Village	6.67	2.0	1500	10000	40.733691	-73.997323

Manhattan apartment rent price statistics

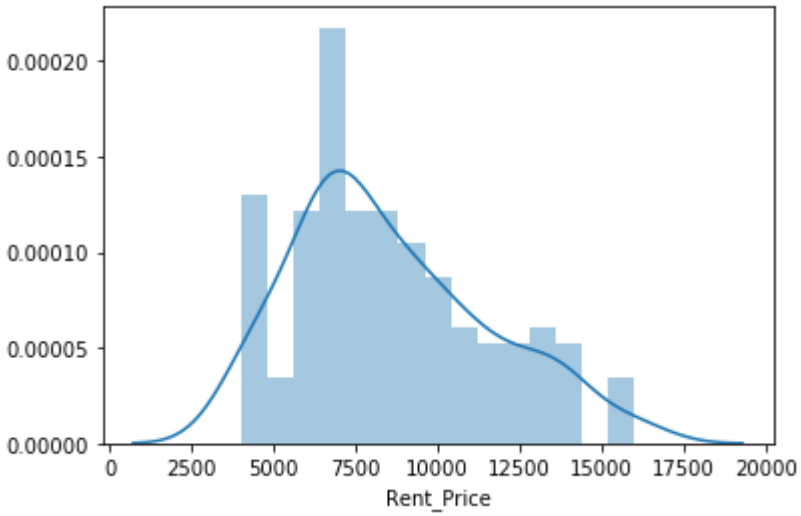
A US 7000 Dollar per month rent is actually around the mean value - similar to Islamabad!

In [21]:

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

Out[21]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f05d17f10f0>
```

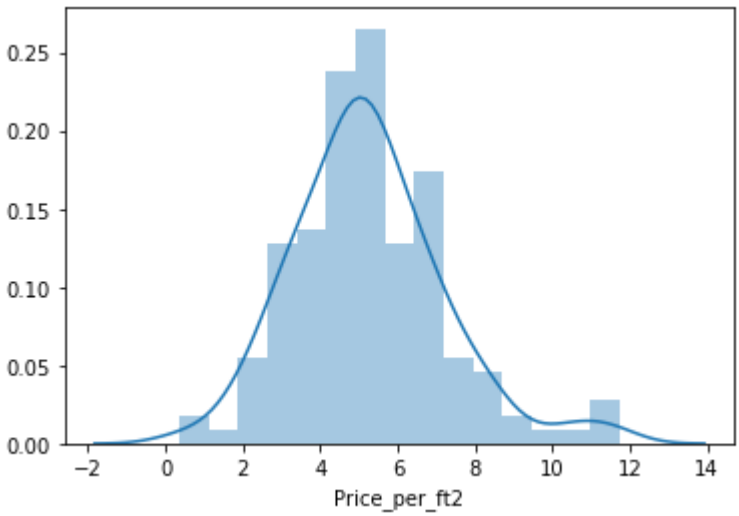


In [22]:

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

Out[22]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f05d16ece48>
```

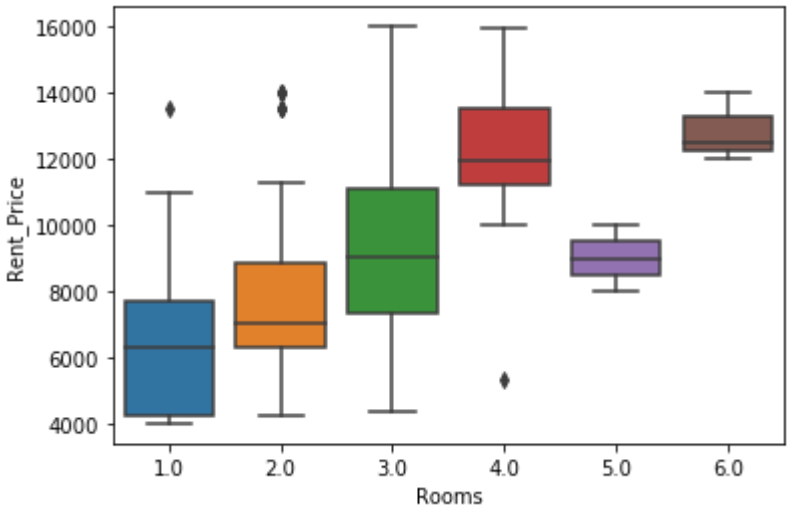


In [23]:

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

Out[23]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f05d1671710>
```



Map of Manhattan apartments for rent

The popups will indicate the address and the monthly price for rent thus making it convenient to select the target apartment with the price condition estimated (max US7000)

In [24]:

```
# 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
```

Out[24]:

Make this Notebook Trusted to load map: File -> Trust Notebook

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

Now, one can point to a rental place for price and address location information while knowing the cluster venues around it. This is an insightful way to explore rental possibilities.

In [25]:

```
# 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['Longi
tude'], 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/AAEAAQAAAAAAAAlgAAAAJGE3OTA4YT
dlLTkzZjUtNDFjYy1iZThlLWQ5OTNkYzlhNzM4OQ.jpg')
FloatImage(url, bottom=5, left=85).add_to(map_manhattan_rent)

map_clusters2
```

Out[25]:

Make this Notebook Trusted to load map: File -> Trust Notebook

Now one can explore a particular rental place and its venues in detail.

In the map above, examination of apartments with rental place below 7000/month is straightforward while knowing the venues around it. We could find an apartment with at the right price and in a location with desirable venues. The next step is to see if it is located near a subway metro station, in next cells work.

In [26]:

```
## 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]))]]
```

Out[26]:

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7 C
3	Inwood	Mexican Restaurant	Lounge	Pizza Place	Café	Wine Bar	Bakery	A Re
5	Manhattanville	Deli / Bodega	Italian Restaurant	Seafood Restaurant	Mexican Restaurant	Sushi Restaurant	Beer Garden	
10	Lenox Hill	Sushi Restaurant	Italian Restaurant	Coffee Shop	Gym / Fitness Center	Pizza Place	Burger Joint	
12	Upper West Side	Italian Restaurant	Bar	Bakery	Vegetarian / Vegan Restaurant	Indian Restaurant	Coffee Shop	Cc
16	Murray Hill	Sandwich Place	Hotel	Japanese Restaurant	Gym / Fitness Center	Coffee Shop	Salon / Barbershop	
17	Chelsea	Coffee Shop	Italian Restaurant	Ice Cream Shop	Bakery	Nightclub	Theater	Ar
18	Greenwich Village	Italian Restaurant	Sushi Restaurant	French Restaurant	Clothing Store	Chinese Restaurant	Café	Re
27	Gramercy	Italian Restaurant	Restaurant	Thrift / Vintage Store	Cocktail Bar	Bagel Shop	Coffee Shop	
29	Financial District	Coffee Shop	Hotel	Gym	Wine Shop	Steakhouse	Bar	Re
31	Noho	Italian Restaurant	French Restaurant	Cocktail Bar	Gift Shop	Bookstore	Grocery Store	I Re
32	Civic Center	Gym / Fitness Center	Bakery	Italian Restaurant	Cocktail Bar	French Restaurant	Sandwich Place	
35	Turtle Bay	Italian Restaurant	Coffee Shop	Steakhouse	Wine Bar	Sushi Restaurant	Hotel	
36	Tudor City	Café	Park	Pizza Place	Mexican Restaurant	Greek Restaurant	Sushi Restaurant	
38	Flatiron	Italian Restaurant	American Restaurant	Gym	Gym / Fitness Center	Yoga Studio	Vegetarian / Vegan Restaurant	

Mapping Manhattan Subway locations

Manhattan subway metro locations (address) was obtained from web scraping sites such as Wikipedia, Google and NY Metro Transit. For simplification, a csv file was produced from the 'numbers' (Apple excel) so that the reading of this file is the starting point here.

The geodata will be obtained via Nominatim using the algorithm below.

In [27]:

```
# 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()
```

Out[27]:

	sub_station	sub_address
0	Dyckman Street Subway Station	170 Nagle Ave, New York, NY 10034, USA
1	57 Street Subway Station	New York, NY 10106, USA
2	Broad St	New York, NY 10005, USA
3	175 Street Station	807 W 177th St, New York, NY 10033, USA
4	5 Av and 53 St	New York, NY 10022, USA

Add columns labeled 'lat' and 'long' to be filled with geodata.

In [28]:

```
#  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)
```

In [30]:

```
## Algorithm to find latitude and longitud for each subway metro station and add them t
o dataframe

for n in range(len(mh)):
    address= mh['sub_address'][n]
    geolocator = Nominatim()
    location = geolocator.geocode(address)
    latitude = location.latitude
    longitude = location.longitude
    mh['lat'][n]=latitude
    mh['long'][n]=longitude
    print(n,latitude,longitude)
    time.sleep(2)

print('Geodata completed')
# save dataframe to csv file
mh.to_csv('MH_subway.csv',index=False)
mh.shape
```

```
/home/jupyterlab/conda/envs/python/lib/python3.6/site-packages/ipykernel_l
auncher.py:5: DeprecationWarning: Using Nominatim with the default "geopy/
1.22.0" `user_agent` is strongly discouraged, as it violates Nominatim's T
oS https://operations.osmfoundation.org/policies/nominatim/ and may possib
ly cause 403 and 429 HTTP errors. Please specify a custom `user_agent` wit
h `Nominatim(user_agent="my-application")` or by overriding the default `u
ser_agent`: `geopy.geocoders.options.default_user_agent = "my-applicatio
n"`. In geopy 2.0 this will become an exception.
  """
/home/jupyterlab/conda/envs/python/lib/python3.6/site-packages/ipykernel_l
auncher.py:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  if __name__ == '__main__':
/home/jupyterlab/conda/envs/python/lib/python3.6/site-packages/ipykernel_l
auncher.py:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  # Remove the CWD from sys.path while we load stuff.
```

```
0 40.8618571 -73.9245089
1 40.7587979 -73.9623427
2 40.7127281 -74.0060152
3 40.8479915 -73.939785
4 40.7587979 -73.9623427
5 40.7256842 -73.9977263
6 40.7587979 -73.9623427
7 40.7587979 -73.9623427
8 40.7202174 -73.99372
9 40.754337125 -74.00291575
10 40.7587979 -73.9623427
11 40.7587979 -73.9623427
12 40.7127281 -74.0060152
13 40.7587979 -73.9623427
14 40.7587979 -73.9623427
15 40.7587979 -73.9623427
16 40.7127281 -74.0060152
17 40.7127281 -74.0060152
18 40.7587979 -73.9623427
19 40.7127281 -74.0060152
20 40.7127281 -74.0060152
21 40.7587979 -73.9623427
22 40.7587979 -73.9623427
23 40.7127281 -74.0060152
24 40.7587979 -73.9623427
25 40.7127281 -74.0060152
26 40.7244125 -73.998317
27 40.8618571 -73.9245089
28 40.7127281 -74.0060152
29 40.7587979 -73.9623427
30 40.7587979 -73.9623427
31 40.7587979 -73.9623427
32 40.7127281 -74.0060152
33 40.7127281 -74.0060152
34 40.7401651 -73.9863046
35 40.7587979 -73.9623427
36 40.7127281 -74.0060152
37 40.7587979 -73.9623427
38 40.7587979 -73.9623427
39 40.7697446 -73.51495832938596
40 40.7587979 -73.9623427
41 40.7684096 -73.981987
42 40.7587979 -73.9623427
43 40.7587979 -73.9623427
44 40.76429 -73.97282
45 40.7571476 -73.9720784
46 40.7587979 -73.9623427
47 40.7587979 -73.9623427
48 40.7729692 -73.9582529
49 40.7286179 -74.0053112
50 40.7587979 -73.9623427
51 40.7119994 -74.0079594
52 40.7127281 -74.0060152
53 40.7127281 -74.0060152
54 40.7587979 -73.9623427
55 40.7587979 -73.9623427
56 40.7127281 -74.0060152
57 40.7127281 -74.0060152
58 40.7587979 -73.9623427
59 40.8499781 -73.9386594
60 40.710358 -74.00782
```

```
61 40.7587979 -73.9623427
62 40.7587979 -73.9623427
63 40.7587979 -73.9623427
64 40.7587979 -73.9623427
65 40.858113 -73.932983
66 40.7627945 -73.9675637
67 40.7127281 -74.0060152
68 40.7587979 -73.9623427
69 40.7587979 -73.9623427
70 40.7127281 -74.0060152
71 40.7587979 -73.9623427
72 40.7587979 -73.9623427
73 40.7587979 -73.9623427
74 40.7127281 -74.0060152
75 40.7598088 -73.9992819
Geodata completed
```

Out[30]:

```
(76, 4)
```

In [31]:

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

```
(76, 4)
```

Out[31]:

	sub_station	sub_address	lat	long
0	Dyckman Street Subway Station	170 Nagle Ave, New York, NY 10034, USA	40.861857	-73.924509
1	57 Street Subway Station	New York, NY 10106, USA	40.758798	-73.962343
2	Broad St	New York, NY 10005, USA	40.712728	-74.006015
3	175 Street Station	807 W 177th St, New York, NY 10033, USA	40.847991	-73.939785
4	5 Av and 53 St	New York, NY 10022, USA	40.758798	-73.962343

In [32]:

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

Out[32]:

```
(21, 4)
```

In [34]:

```
mhsub1.tail()
```

Out[34]:

	sub_station	sub_address	lat	long
16	190 Street Subway Station	Bennett Ave, New York, NY 10040, USA	40.858113	-73.932983
17	59 St-Lexington Av Station	E 60th St, New York, NY 10065, USA	40.762794	-73.967564
18	23 Street Station	New York, NY 10010, United States	40.758798	-73.962343
19	14 Street / 8 Av	New York, NY 10014, United States	40.712728	-74.006015
20	MTA New York City	525 11th Ave, New York, NY 10018, USA	40.759809	-73.999282

MAP of Manhattan showing the location of subway stations

In [35]:

```
# map subway stations
# create map of Manhattan using latitude and longitude values obtain previoulsy via Mon
inatim 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
```

Out[35]:

Make this Notebook Trusted to load map: File -> Trust Notebook

Map of Manhattan showing places for rent and the subway locations nearby.

Now, we can visualize the desirable rental places and their nearest subway station. Popups display rental address and monthly rental price and the subway station name.

Notice that the icon in the top-right corner is a "ruler" that allows to measure the distance from a rental place to an specific subway station.

In [36]:

```
mh_rent.head()
```

Out[36]:

	Address	Area	Price_per_ft2	Rooms	Area-ft2	Rent_Price	Lat	Long
0	West 105th Street	Upper West Side	2.94	5.0	3400	10000	40.799771	-73.966213
1	East 97th Street	Upper East Side	3.57	3.0	2100	7500	40.788517	-73.955118
2	West 105th Street	Upper West Side	1.89	4.0	2800	5300	40.799771	-73.966213
3	CARMINE ST.	West Village	3.03	2.0	1650	5000	40.730337	-74.002476
4	171 W 23RD ST.	Chelsea	3.45	2.0	1450	5000	40.744118	-73.995299

In [37]:

```
# 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/AAEAAQAAAAAAAAlgAAAAJGE3OTA4YT
dlLTkzZjUtNDFjYy1iZThlLWQ5OTNkYzlhNzM4OQ.jpg')
FloatImage(url, bottom=5, left=85).add_to(map_manhattan_rent)

map_manhattan_rent
```

Out[37]:

Make this Notebook Trusted to load map: File -> Trust Notebook

4. Results¶

A CONSOLIDATED MAP

Let's consolidate all the required information to make the apartment selection in one map.

Map of Manhattan with rental places, subway locations and cluster of venues.

Red dots are Subway stations, Blue dots are apartments available for rent, Bubbles are the clusters of venues

In [38]:

```
# 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['Longi
tude'], 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
```

Out[38]:

Make this Notebook Trusted to load map: File -> Trust Notebook

Problem Resolution

The above consolidated 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

Venues for Apartment 1 - Cluster 2

In [39]:

```
## 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]))]]
```

Out[39]:

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7 C
0	Marble Hill	Coffee Shop	Discount Store	Yoga Studio	Steakhouse	Supplement Shop	Tennis Stadium	
1	Chinatown	Chinese Restaurant	Cocktail Bar	Dim Sum Restaurant	American Restaurant	Vietnamese Restaurant	Salon / Barbershop	
6	Central Harlem	African Restaurant	Seafood Restaurant	French Restaurant	American Restaurant	Cosmetics Shop	Chinese Restaurant	
9	Yorkville	Coffee Shop	Gym	Bar	Italian Restaurant	Sushi Restaurant	Pizza Place	Re
14	Clinton	Theater	Italian Restaurant	Coffee Shop	American Restaurant	Gym / Fitness Center	Hotel	Wi
23	Soho	Clothing Store	Boutique	Women's Store	Shoe Store	Men's Store	Furniture / Home Store	Re
26	Morningside Heights	Coffee Shop	American Restaurant	Park	Bookstore	Pizza Place	Sandwich Place	
34	Sutton Place	Gym / Fitness Center	Italian Restaurant	Furniture / Home Store	Indian Restaurant	Dessert Shop	American Restaurant	
39	Hudson Yards	Coffee Shop	Italian Restaurant	Hotel	Theater	American Restaurant	Café	

In [40]:

```
## 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]))]]
```

Out[40]:

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7 C
3	Inwood	Mexican Restaurant	Lounge	Pizza Place	Café	Wine Bar	Bakery	A Re
5	Manhattanville	Deli / Bodega	Italian Restaurant	Seafood Restaurant	Mexican Restaurant	Sushi Restaurant	Beer Garden	
10	Lenox Hill	Sushi Restaurant	Italian Restaurant	Coffee Shop	Gym / Fitness Center	Pizza Place	Burger Joint	
12	Upper West Side	Italian Restaurant	Bar	Bakery	Vegetarian / Vegan Restaurant	Indian Restaurant	Coffee Shop	Cc
16	Murray Hill	Sandwich Place	Hotel	Japanese Restaurant	Gym / Fitness Center	Coffee Shop	Salon / Barbershop	
17	Chelsea	Coffee Shop	Italian Restaurant	Ice Cream Shop	Bakery	Nightclub	Theater	Ar
18	Greenwich Village	Italian Restaurant	Sushi Restaurant	French Restaurant	Clothing Store	Chinese Restaurant	Café	Re
27	Gramercy	Italian Restaurant	Restaurant	Thrift / Vintage Store	Cocktail Bar	Bagel Shop	Coffee Shop	
29	Financial District	Coffee Shop	Hotel	Gym	Wine Shop	Steakhouse	Bar	Re
31	Noho	Italian Restaurant	French Restaurant	Cocktail Bar	Gift Shop	Bookstore	Grocery Store	I Re
32	Civic Center	Gym / Fitness Center	Bakery	Italian Restaurant	Cocktail Bar	French Restaurant	Sandwich Place	
35	Turtle Bay	Italian Restaurant	Coffee Shop	Steakhouse	Wine Bar	Sushi Restaurant	Hotel	
36	Tudor City	Café	Park	Pizza Place	Mexican Restaurant	Greek Restaurant	Sushi Restaurant	
38	Flatiron	Italian Restaurant	American Restaurant	Gym	Gym / Fitness Center	Yoga Studio	Vegetarian / Vegan Restaurant	

Apartment Selection

Using the "one map" above, I was able to explore all possibilities since the popups provide the information needed for a good decision.

Apartment 1 rent cost is US7500 slightly above the US7000 budget. Apt 1 is located 400 meters from subway station at 59th Street and work place (Park Ave and 53rd) is another 600 meters way. I can walk to work place and use subway for other places around. Venues for this apt are as of Cluster 2 and it is located in a fine district in the East side of Manhattan.

Apartment 2 rent cost is US6935, just under the US7000 budget. Apt 2 is located 60 meters from subway station at Fulton Street, but I will have to ride the subway daily to work , possibly 40-60 min ride. Venues for this apt are as of Cluster 3. Based on current Islamabad venues, I feel that Cluster 2 type of venues is a closer resemblance to my current place. That means that APARTMENT 1 is a better choice since the extra monthly rent is worth the conveniences it provides.

5. 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.

6. 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 Data Science that continue to appear for application in several business fields.

The End

In [ ]: