Battle of Neighborhoods Coursera Project Report

Content

1. Introduction Section

1.1 Scenario

How can I find a convenient and enjoyable place similar to one in Singapore? 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 should not exceed 7,000 dollars per month

- top ammenities in the selected neighborhood shall be similar to current residence
- desirable to have venues such as coffee shops, restaurants Asian Thai, wine stores, gym and food shops

1.2 Problem to be solved

The challenge is to find a suitable apartment for rent in Manhattan NY that complies with the demands on location, price and venues.

1.3 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 Section

2.1 Data of place to be compared with

I've picked 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

2.2 Data required to solve the problem

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

- List of Boroughs and neighborhoods of Manhattan with their geodata (latitud and longitud)
- 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 (than can be clustered)
- Venues for subway metro stations, as needed

2.3 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 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 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.

2.4 Mapping 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

3. Methodology

3.1 Process steps and strategy to resolve the problem

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.

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 (assume: 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?

3.2 Data Science Methods, machine learning, mapping tools and exploratory data analysis

In [2]:
```
import numpy as np
import time
import pandas as pd
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)

import json
import requests
from pandas.io.json import json_normalize

from geopy.geocoders import Nominatim
from geopy.exc import GeocoderTimedOut
import folium

print('Libraries imported.')
```

```
Libraries imported.
```

In [3]:
```
address = 'Mccallum Street, Singapore'

geolocator = Nominatim()
location = geolocator.geocode(address, timeout=10)
latitude = location.latitude
longitude = location.longitude
print('The geograpical coordinate of the place in Singapore are {}, {}.'.format(l
```

```
F:\Anaconda\lib\site-packages\ipykernel_launcher.py:3: DeprecationWarning: Usin
g Nominatim with the default "geopy/1.20.0" `user_agent` is strongly discourage
d, as it violates Nominatim's ToS https://operations.osmfoundation.org/policie
s/nominatim/ (https://operations.osmfoundation.org/policies/nominatim/) and may
possibly cause 403 and 429 HTTP errors. Please specify a custom `user_agent` wi
th `Nominatim(user_agent="my-application")` or by overriding the default `user_
agent`: `geopy.geocoders.options.default_user_agent = "my-application"`. In geo
py 2.0 this will become an exception.
  This is separate from the ipykernel package so we can avoid doing imports unt
il
```

```
The geograpical coordinate of the place in Singapore are 1.2792423, 103.848131
2.
```

In [4]:
```
neighborhood_latitude=1.2792423
neighborhood_longitude=103.8481312
```

Dial FourSquare to find venues around chosen neighborhood in Singapore

In [5]:
```
CLIENT_ID="LUETHARMZN0ATS5LKT1YNTB2C5Y2MS42IUKEYIJ5JGN1NNJU"
CLIENT_SECRET="4H4F054UDSZJ2EATK5DEMCMRPR3RNRCTXCZBCCCIELED3EZB"
VERSION = '20180604'
LIMIT = 100
```

In [6]:
```
# define radius
radius = 500

# create URL
url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={
    CLIENT_ID,
    CLIENT_SECRET,
    VERSION,
    neighborhood_latitude,
    neighborhood_longitude,
    radius,
    LIMIT)

# display URL
url
```

Out[6]:
```
'https://api.foursquare.com/v2/venues/explore?&client_id=LUETHARMZN0ATS5LKT1YNT
B2C5Y2MS42IUKEYIJ5JGN1NNJU&client_secret=4H4F054UDSZJ2EATK5DEMCMRPR3RNRCTXCZBCC
CIELED3EZB&v=20180604&ll=1.2792423,103.8481312&radius=500&limit=100'
```

In [7]:
```
results = requests.get(url).json()
```

Function that extracts the category of the venue - borrow from the Foursquare lab

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', 'venu
SGnearby_venues =SGnearby_venues.loc[:, filtered_columns]

# filter the category for each row
SGnearby_venues['venue.categories'] = SGnearby_venues.apply(get_category_type, ax

# clean columns
SGnearby_venues.columns = [col.split(".")[-1] for col in SGnearby_venues.columns]

# venues near chosen neighborhood in Singapore
SGnearby_venues.head(10)
```

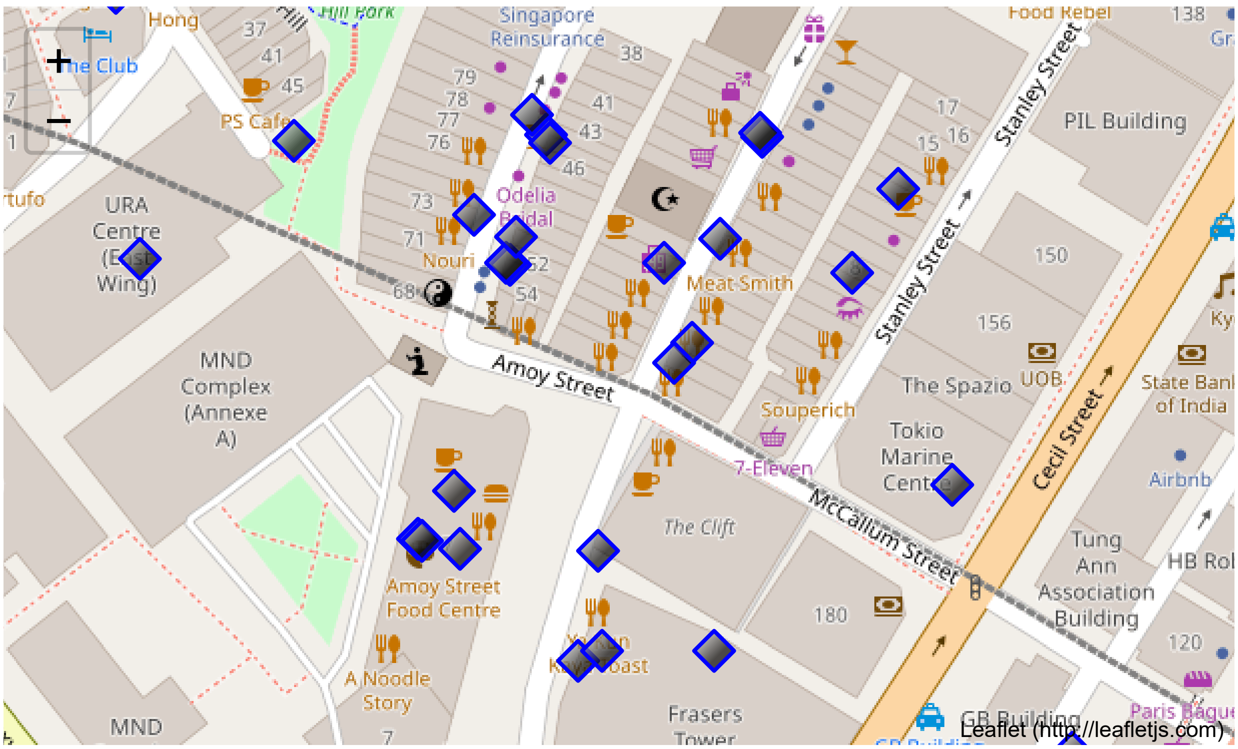
Out[9]:

	name	categories	lat	lng
0	Napoleon Food & Wine Bar	Wine Bar	1.279925	103.847333
1	Pepper Bowl	Asian Restaurant	1.279371	103.846710
2	Native	Cocktail Bar	1.280135	103.846844
3	Park Bench Deli	Deli / Bodega	1.279872	103.847287
4	Freehouse	Beer Garden	1.281254	103.848513
5	Sofitel So Singapore	Hotel	1.280124	103.849867
6	Coffee Break	Coffee Shop	1.279529	103.846695
7	PS.Cafe	Café	1.280468	103.846264
8	Dumpling Darlings	Dumpling Restaurant	1.280483	103.846942
9	Nouri	Modern European Restaurant	1.280267	103.846750

Map of Singapore with venues near residence place for reference

In [10]:
```
# create map of Singapore place  using latitude and longitude values
map_sg = folium.Map(location=[latitude, longitude], zoom_start=20)

# add markers to map
for lat, lng, label in zip(SGnearby_venues['lat'], SGnearby_venues['lng'], SGnear
    label = folium.Popup(label, parse_html=True)
    folium.RegularPolygonMarker(
        [lat, lng],
        number_of_sides=4,
        radius=10,
        popup=label,
        color='blue',
        fill_color='#0f0f0f',
        fill_opacity=0.7,
    ).add_to(map_sg)

map_sg
```

Out[10]:



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('C:\Users\MUJ\Documents\Jupyter Notebooks\my_neighb
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('C:\Users\MUJ\Documents\Jupyter Notebooks\merged.c
manhattan_merged.head()
```

Out[13]:

	Borough	Neighborhood	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Co
0	Manhattan	Marble Hill	40.876551	-73.910660	2	Coffee Shop	Discount Store	Yoga Studio	Steak
1	Manhattan	Chinatown	40.715618	-73.994279	2	Chinese Restaurant	Cocktail Bar	Dim Sum Restaurant	Am Rest
2	Manhattan	Washington Heights	40.851903	-73.936900	4	Café	Bakery	Mobile Phone Shop	Pizza
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	B

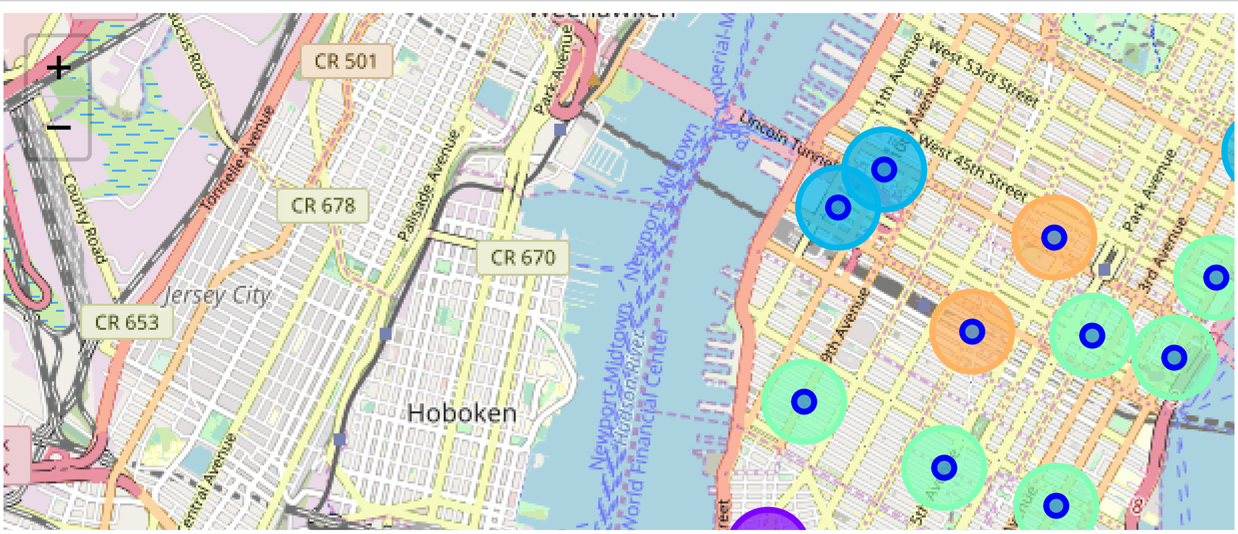
popus allow to identify each neighborhood and the cluster of venues around it in order to

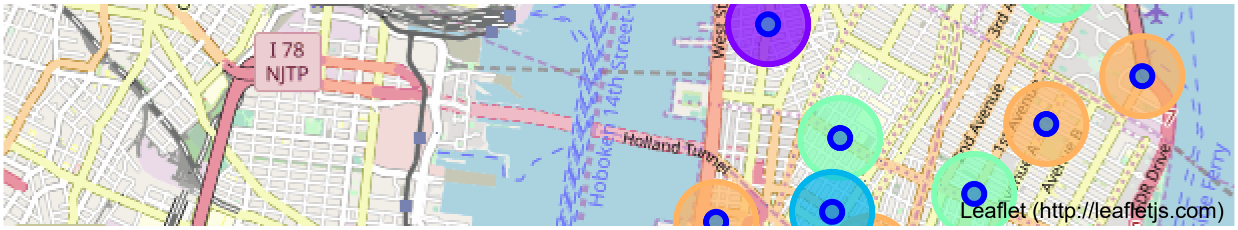
proceed to examine in more detail in the next cell

In [14]:
```
import matplotlib.cm as cm
import matplotlib.colors as colors
```

In [15]:
```
# 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[
    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'
    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[15]:





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

In [16]:
```
## kk is the cluster number to explore
kk = 2
manhattan_merged.loc[manhattan_merged['Cluster Labels'] == kk, manhattan_merged.c
```

Out[16]:

	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	7th Mos Commo Venu
0	Marble Hill	Coffee Shop	Discount Store	Yoga Studio	Steakhouse	Supplement Shop	Tennis Stadium	Sho Stor
1	Chinatown	Chinese Restaurant	Cocktail Bar	Dim Sum Restaurant	American Restaurant	Vietnamese Restaurant	Salon / Barbershop	Noodl Hous
6	Central Harlem	African Restaurant	Seafood Restaurant	French Restaurant	American Restaurant	Cosmetics Shop	Chinese Restaurant	Ever Spac
9	Yorkville	Coffee Shop	Gym	Bar	Italian Restaurant	Sushi Restaurant	Pizza Place	Mexica Restaurar
14	Clinton	Theater	Italian Restaurant	Coffee Shop	American Restaurant	Gym / Fitness Center	Hotel	Wine Sho
23	Soho	Clothing Store	Boutique	Women's Store	Shoe Store	Men's Store	Furniture / Home Store	Italia Restaurar
26	Morningside Heights	Coffee Shop	American Restaurant	Park	Bookstore	Pizza Place	Sandwich Place	Burge Joir
34	Sutton Place	Gym / Fitness Center	Italian Restaurant	Furniture / Home Store	Indian Restaurant	Dessert Shop	American Restaurant	Baker
39	Hudson Yards	Coffee Shop	Italian Restaurant	Hotel	Theater	American Restaurant	Café	Gym Fitnes Cente

Several Manhattan real estate webs were webscrapped to collect rental data, as mentioned in section 2.0 . The resut was summarized in a csv file for direct reading, in order to consolidate the proces

In [17]:
```
# csv files with rental places with basic data but still wihtout geodata ( latitu
# pd.read_csv(' le.csv', header=None, nrows=5)
mh_rent=pd.read_csv('C:\Users\MUJ\Documents\Jupyter Notebooks\manhattan_flats_pri
mh_rent.head()
```

Out[17]:

	Address	Area	Price_per_ft2	Rooms	Area-ft2	Rent_Price	Lat	Long
0	West 105th Street	Upper West Side	2.94	5	3400	10000	NaN	NaN
1	East 97th Street	Upper East Side	3.57	3	2100	7500	NaN	NaN
2	West 105th Street	Upper West Side	1.89	4	2800	5300	NaN	NaN
3	CARMINE ST.	West Village	3.03	2	1650	5000	NaN	NaN
4	171 W 23RD ST.	Chelsea	3.45	2	1450	5000	NaN	NaN

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

In [ ]:
```
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('manhattan_rent.csv',index=False)
```

In [19]:
```
mh_rent=pd.read_csv('C:\Users\MUJ\Documents\Jupyter Notebooks\manhattan_rent.csv'
mh_rent.head()
```

Out[19]:

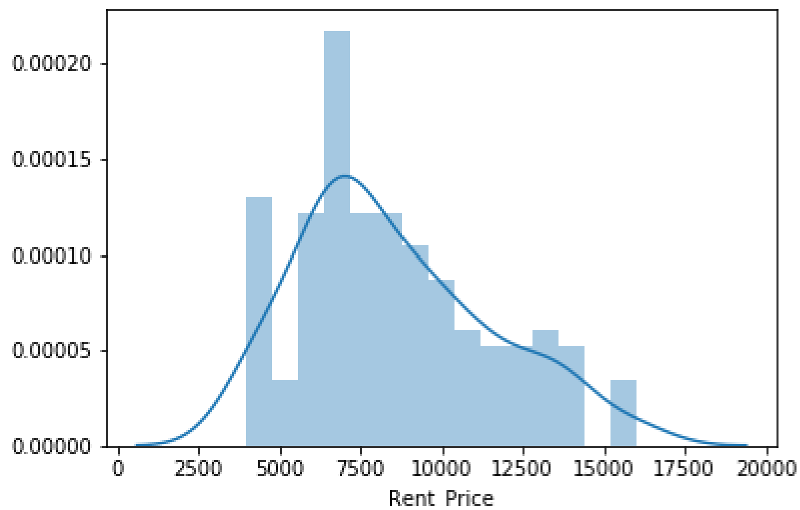
	Address	Area	Price_per_ft2	Rooms	Area-ft2	Rent_Price	Lat	Long
0	West 105th Street	Upper West Side	2.94	5	3400	10000	40.799771	-73.966213
1	East 97th Street	Upper East Side	3.57	3	2100	7500	40.788585	-73.955277
2	West 105th Street	Upper West Side	1.89	4	2800	5300	40.799771	-73.966213
3	CARMINE ST.	West Village	3.03	2	1650	5000	40.730523	-74.001873
4	171 W 23RD ST.	Chelsea	3.45	2	1450	5000	40.744118	-73.995299

Manhattan apartment rent price statistics

In [20]:
```
import seaborn as sns
import matplotlib as plt
%matplotlib inline
```

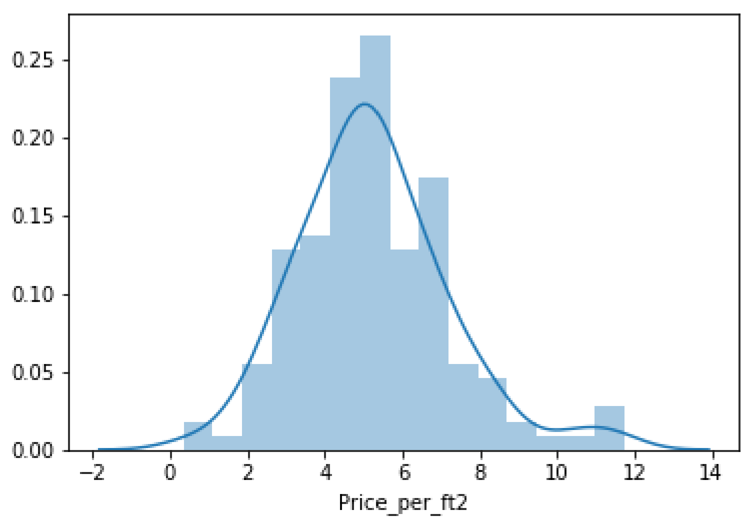
In [21]:
```
sns.distplot(mh_rent['Rent_Price'],bins=15)
```

Out[21]: <matplotlib.axes._subplots.AxesSubplot at 0xf6badd8>



In [37]:
```
import seaborn as sns
sns.distplot(mh_rent['Price_per_ft2'],bins=15)
```

Out[37]: <matplotlib.axes._subplots.AxesSubplot at 0xec9f908>

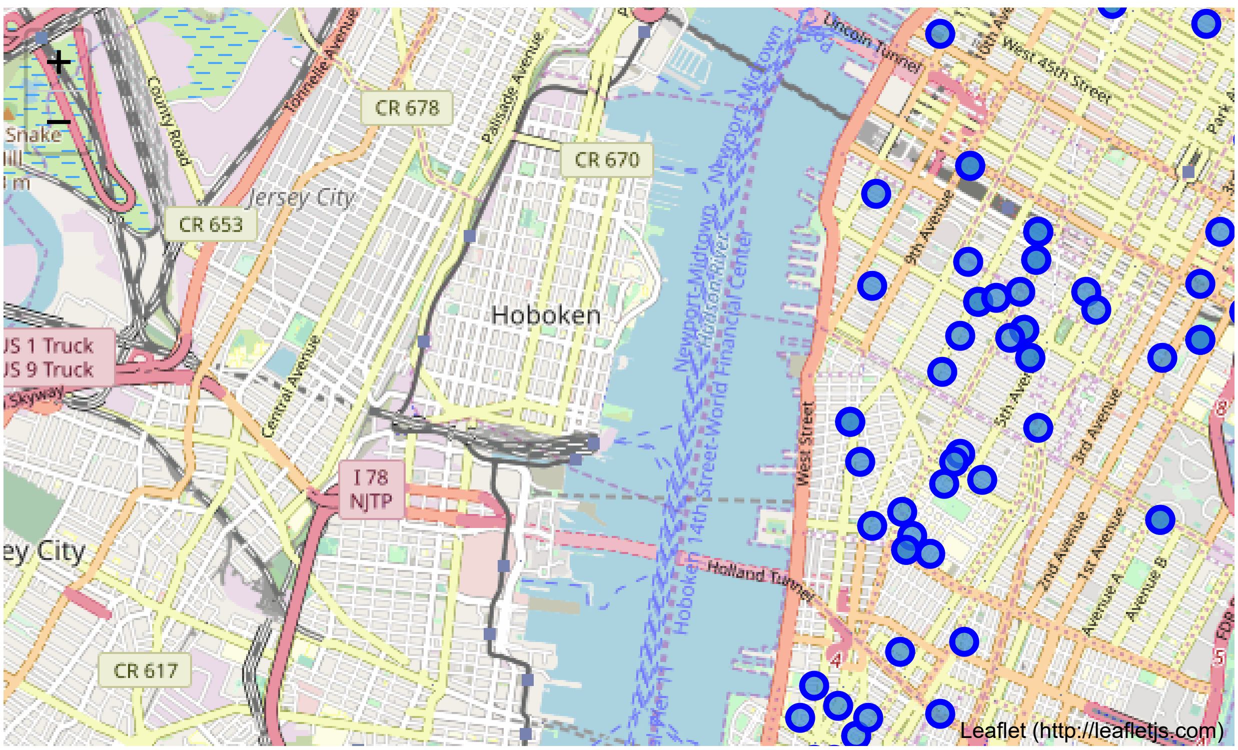


A US 7000 Dollar per month rent is actually around the mean value - similar to Singapore

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

In [22]:
```
# 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_P
    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[22]:

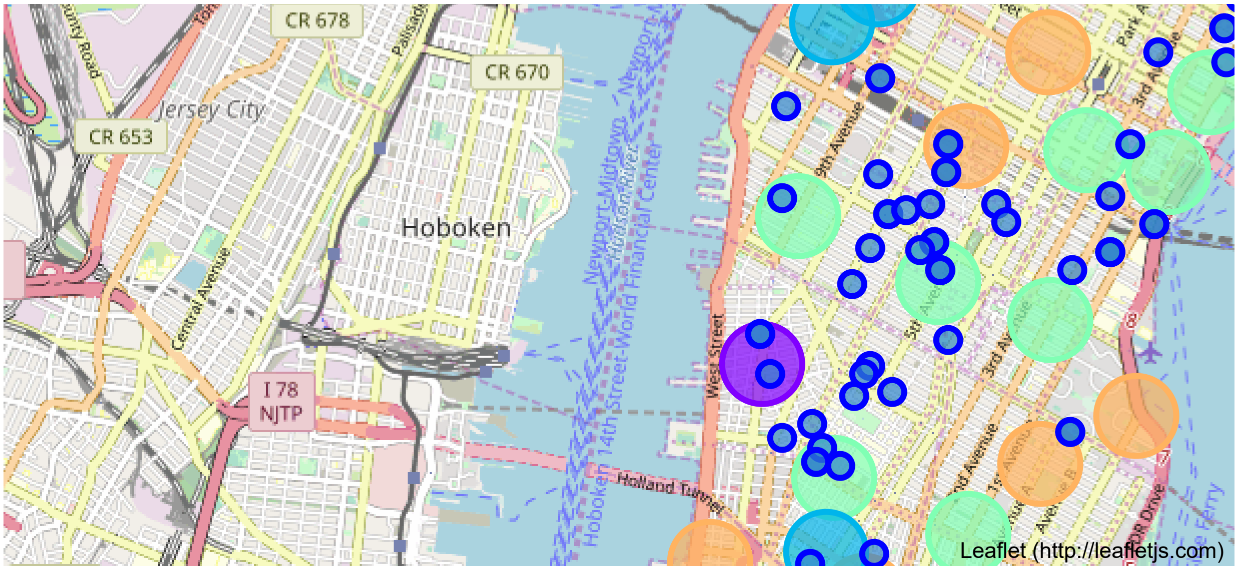


Map of Manhattan showing the places for rent and the cluster of venues

Now, one can point to a rental place for price and address location information while knowing the cluster venues around it.

In [23]:
```
# 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[
    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_P
    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/AAEAAQAAAAAAAAlgAAAAJGE3
FloatImage(url, bottom=5, left=85).add_to(map_manhattan_rent)

map_clusters2
```

Out[23]:



In the map above, examination of appartments with rental place below 7000/month is straightforwad while knowing the venues around it.

We could find an appartment 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 [40]:
```
## kk is the cluster number to explore
kk = 3
manhattan_merged.loc[manhattan_merged['Cluster Labels'] == kk, manhattan_merged.c
```

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	7th Mos Commoı Venu
3	Inwood	Mexican Restaurant	Lounge	Pizza Place	Café	Wine Bar	Bakery	America Restauran
5	Manhattanville	Deli / Bodega	Italian Restaurant	Seafood Restaurant	Mexican Restaurant	Sushi Restaurant	Beer Garden	Coffe Sho
10	Lenox Hill	Sushi Restaurant	Italian Restaurant	Coffee Shop	Gym / Fitness Center	Pizza Place	Burger Joint	Deli Bodeg
12	Upper West Side	Italian Restaurant	Bar	Bakery	Vegetarian / Vegan Restaurant	Indian Restaurant	Coffee Shop	Cosmetic Sho
16	Murray Hill	Sandwich Place	Hotel	Japanese Restaurant	Gym / Fitness Center	Coffee Shop	Salon / Barbershop	Burge Join
17	Chelsea	Coffee Shop	Italian Restaurant	Ice Cream Shop	Bakery	Nightclub	Theater	Art Galler
18	Greenwich Village	Italian Restaurant	Sushi Restaurant	French Restaurant	Clothing Store	Chinese Restaurant	Café	India Restauran
27	Gramercy	Italian Restaurant	Restaurant	Thrift / Vintage Store	Cocktail Bar	Bagel Shop	Coffee Shop	Pizz Plac
29	Financial District	Coffee Shop	Hotel	Gym	Wine Shop	Steakhouse	Bar	Italia Restauran
31	Noho	Italian Restaurant	French Restaurant	Cocktail Bar	Gift Shop	Bookstore	Grocery Store	Mexica Restauran
32	Civic Center	Gym / Fitness Center	Bakery	Italian Restaurant	Cocktail Bar	French Restaurant	Sandwich Place	Coffe Sho
35	Turtle Bay	Italian Restaurant	Coffee Shop	Steakhouse	Wine Bar	Sushi Restaurant	Hotel	Noodl Hous
36	Tudor City	Café	Park	Pizza Place	Mexican Restaurant	Greek Restaurant	Sushi Restaurant	Hote
38	Flatiron	Italian Restaurant	American Restaurant	Gym	Gym / Fitness Center	Yoga Studio	Vegetarian / Vegan Restaurant	Baker

Mapping Manhattan Subway locations

In [24]:
```
# A csv file summarized the subway station and the addresses for next step to det
mh=pd.read_csv('C:\Users\MUJ\Documents\Jupyter Notebooks\NYC_subway_list.csv')
mh.head()
```

Out[24]:

	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 colums labeled 'lat' and 'long' to be filled with geodata

In [25]:
```
#  Add columns 'lat'  and 'long' to mh dataframe - with random temporary numbers
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 [ ]:
```
## Algorythm to find latitude and longitud for each subway metro station and add

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)
```

In [26]:
```
mh=pd.read_csv('C:\Users\MUJ\Documents\Jupyter Notebooks\MH_subway.csv')
print(mh.shape)
mh.head()
```

(76, 4)

Out[26]:

	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	40.764250	-73.954525
2	Broad St	New York, NY 10005, USA	40.730862	-73.987156
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	40.764250	-73.954525

In [27]:
```
# removing duplicate rows and creating new set mhsub1
mhsub1=mh.drop_duplicates(subset=['lat','long'], keep="last").reset_index(drop=Tr
mhsub1.shape
```

Out[27]: (22, 4)

In [28]:
```
mhsub1.tail()
```

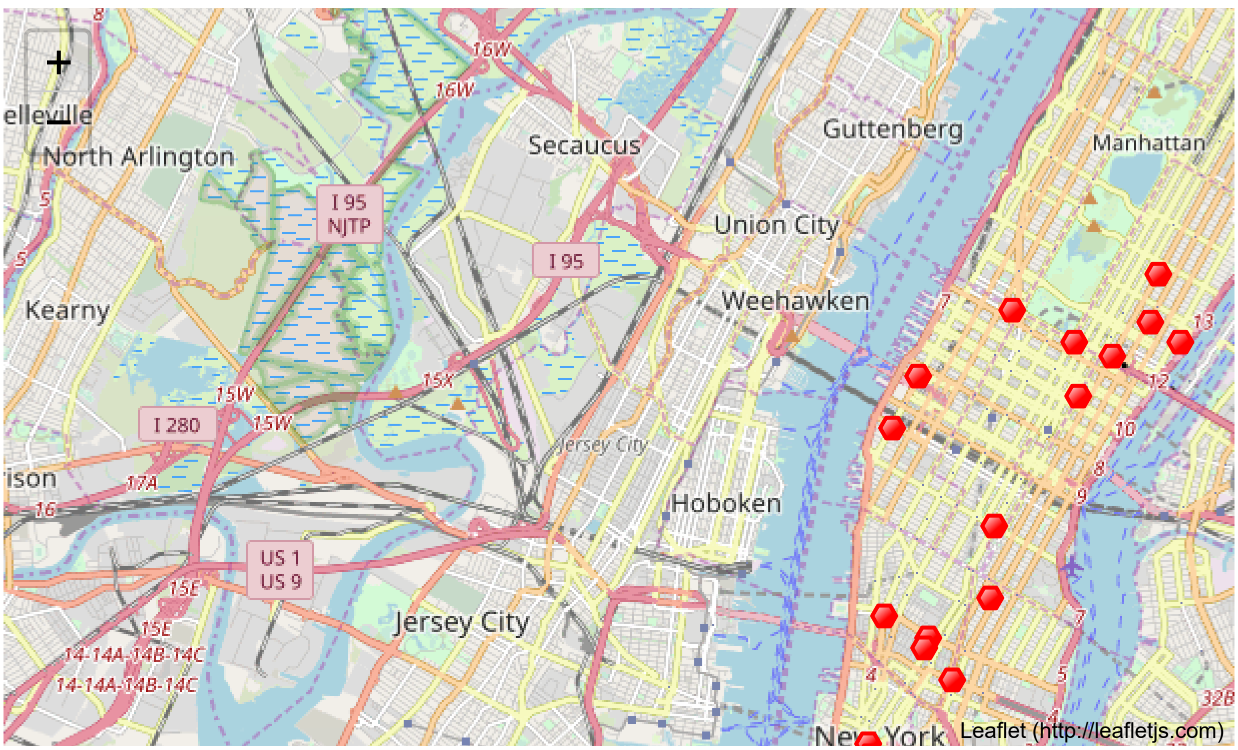
Out[28]:

	sub_station	sub_address	lat	long
17	190 Street Subway Station	Bennett Ave, New York, NY 10040, USA	40.858113	-73.932983
18	59 St-Lexington Av Station	E 60th St, New York, NY 10065, USA	40.762259	-73.966271
19	57 Street Station	New York, NY 10019, United States	40.764250	-73.954525
20	14 Street / 8 Av	New York, NY 10014, United States	40.730862	-73.987156
21	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 [29]:
```
# map subway stations
# create map of Manhattan using latitude and longitude values obtain previoulsy v
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'].
    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[29]:



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

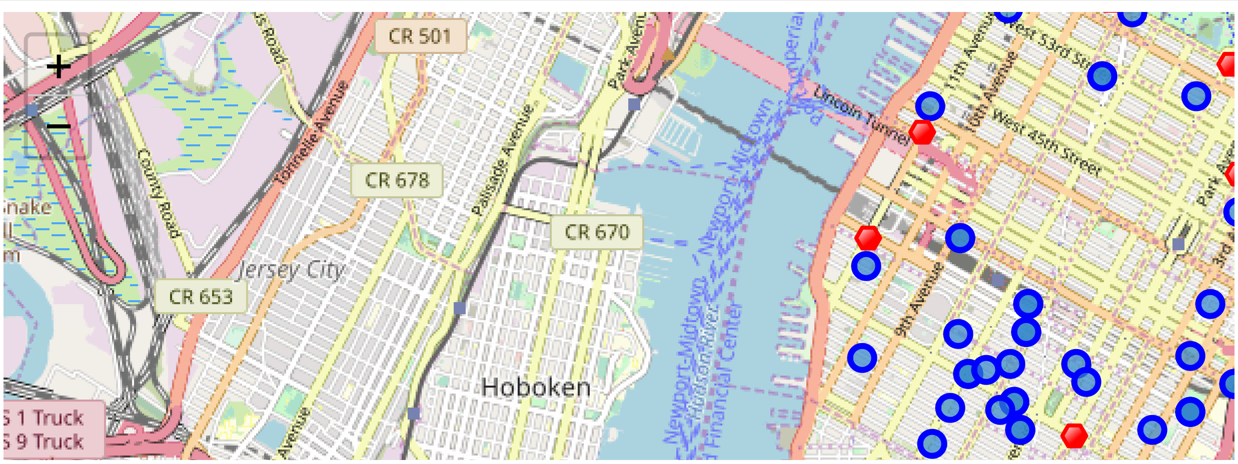
In [30]:
```
mh_rent.head()
```

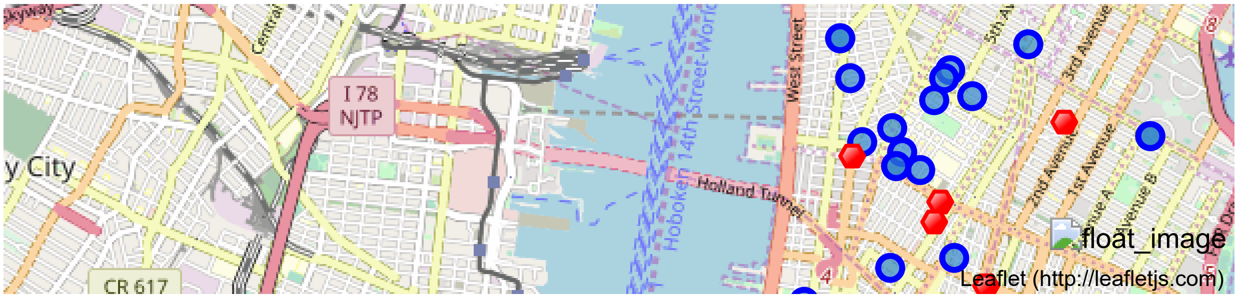
Out[30]:

	Address	Area	Price_per_ft2	Rooms	Area-ft2	Rent_Price	Lat	Long
0	West 105th Street	Upper West Side	2.94	5	3400	10000	40.799771	-73.966213
1	East 97th Street	Upper East Side	3.57	3	2100	7500	40.788585	-73.955277
2	West 105th Street	Upper West Side	1.89	4	2800	5300	40.799771	-73.966213
3	CARMINE ST.	West Village	3.03	2	1650	5000	40.730523	-74.001873
4	171 W 23RD ST.	Chelsea	3.45	2	1450	5000	40.744118	-73.995299

In [31]:
```
# 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_P
    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'].
    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/AAEAAQAAAAAAAAlgAAAAJGE3
FloatImage(url, bottom=5, left=85).add_to(map_manhattan_rent)

map_manhattan_rent
```

Out[31]:





4. Results

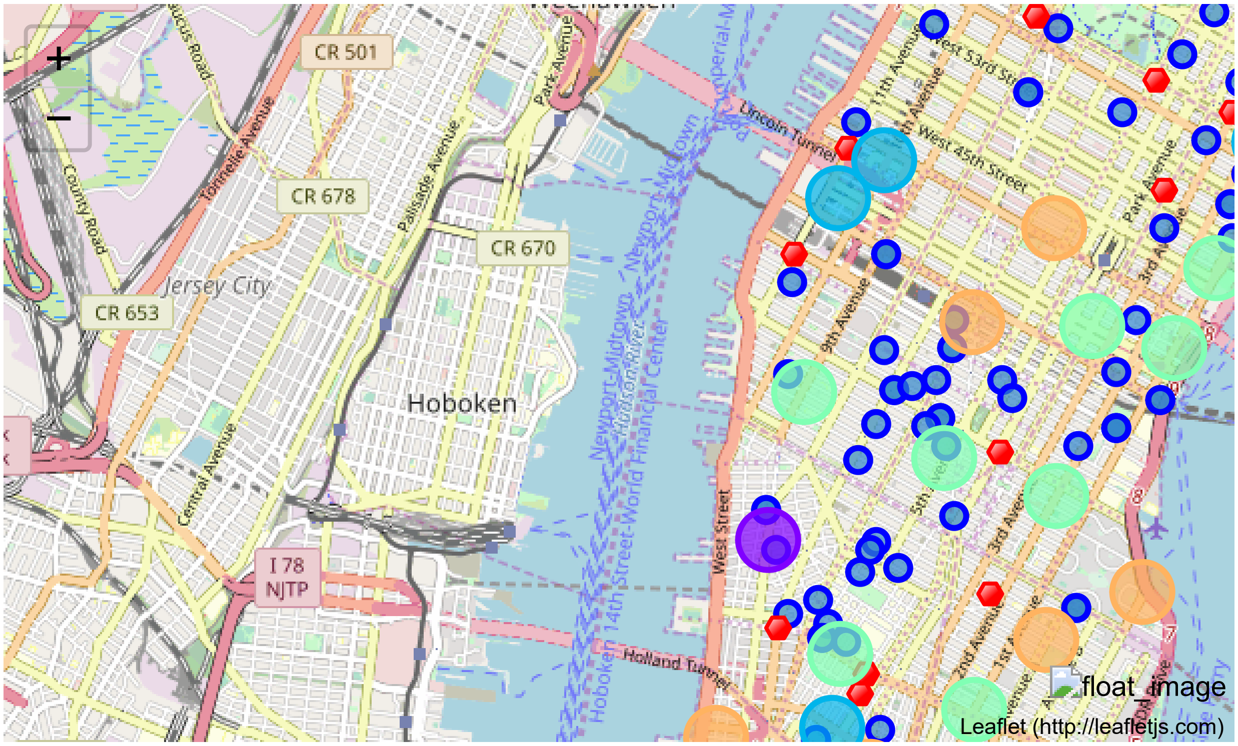
Let's consolidate all the required inforamtion to make the apartment selection in one map for Manhattan with rental places, subway locations and cluster of venues

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

In [32]:
```
# 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_P
    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'].
    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[
    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/AAEAAQAAAAAAAAlgAAAAJGE3(
FloatImage(url, bottom=5, left=85).add_to(map_mh_one)

map_mh_one
```

Out[32]:



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 [33]:
```
## kk is the cluster number to explore
kk = 2
manhattan_merged.loc[manhattan_merged['Cluster Labels'] == kk, manhattan_merged.c
```

Out[33]:

	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	7th Mos Commo Venu
0	Marble Hill	Coffee Shop	Discount Store	Yoga Studio	Steakhouse	Supplement Shop	Tennis Stadium	Sho Stor
1	Chinatown	Chinese Restaurant	Cocktail Bar	Dim Sum Restaurant	American Restaurant	Vietnamese Restaurant	Salon / Barbershop	Noodl Hous
6	Central Harlem	African Restaurant	Seafood Restaurant	French Restaurant	American Restaurant	Cosmetics Shop	Chinese Restaurant	Ever Spac
9	Yorkville	Coffee Shop	Gym	Bar	Italian Restaurant	Sushi Restaurant	Pizza Place	Mexica Restaurar
14	Clinton	Theater	Italian Restaurant	Coffee Shop	American Restaurant	Gym / Fitness Center	Hotel	Wine Sho
23	Soho	Clothing Store	Boutique	Women's Store	Shoe Store	Men's Store	Furniture / Home Store	Italia Restaurar
26	Morningside Heights	Coffee Shop	American Restaurant	Park	Bookstore	Pizza Place	Sandwich Place	Burge Joir
34	Sutton Place	Gym / Fitness Center	Italian Restaurant	Furniture / Home Store	Indian Restaurant	Dessert Shop	American Restaurant	Baker
39	Hudson Yards	Coffee Shop	Italian Restaurant	Hotel	Theater	American Restaurant	Café	Gym Fitnes Cente

Venues for Apartment 2 - Cluster 3

In [34]:
```
## kk is the cluster number to explore
kk = 3
manhattan_merged.loc[manhattan_merged['Cluster Labels'] == kk, manhattan_merged.c
```

Out[34]:

	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	7th Mos Commo Venu
3	Inwood	Mexican Restaurant	Lounge	Pizza Place	Café	Wine Bar	Bakery	America Restauran
5	Manhattanville	Deli / Bodega	Italian Restaurant	Seafood Restaurant	Mexican Restaurant	Sushi Restaurant	Beer Garden	Coffe Sho
10	Lenox Hill	Sushi Restaurant	Italian Restaurant	Coffee Shop	Gym / Fitness Center	Pizza Place	Burger Joint	Deli Bodeg
12	Upper West Side	Italian Restaurant	Bar	Bakery	Vegetarian / Vegan Restaurant	Indian Restaurant	Coffee Shop	Cosmetic Sho
16	Murray Hill	Sandwich Place	Hotel	Japanese Restaurant	Gym / Fitness Center	Coffee Shop	Salon / Barbershop	Burge Join
17	Chelsea	Coffee Shop	Italian Restaurant	Ice Cream Shop	Bakery	Nightclub	Theater	Art Galler
18	Greenwich Village	Italian Restaurant	Sushi Restaurant	French Restaurant	Clothing Store	Chinese Restaurant	Café	India Restauran
27	Gramercy	Italian Restaurant	Restaurant	Thrift / Vintage Store	Cocktail Bar	Bagel Shop	Coffee Shop	Pizz Plac
29	Financial District	Coffee Shop	Hotel	Gym	Wine Shop	Steakhouse	Bar	Italia Restauran
31	Noho	Italian Restaurant	French Restaurant	Cocktail Bar	Gift Shop	Bookstore	Grocery Store	Mexica Restauran
32	Civic Center	Gym / Fitness Center	Bakery	Italian Restaurant	Cocktail Bar	French Restaurant	Sandwich Place	Coffe Sho
35	Turtle Bay	Italian Restaurant	Coffee Shop	Steakhouse	Wine Bar	Sushi Restaurant	Hotel	Noodl Hous
36	Tudor City	Café	Park	Pizza Place	Mexican Restaurant	Greek Restaurant	Sushi Restaurant	Hote
38	Flatiron	Italian Restaurant	American Restaurant	Gym	Gym / Fitness Center	Yoga Studio	Vegetarian / Vegan Restaurant	Baker

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. Apartment 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 aroung. 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. Apartment 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 Singapore 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

I am impressed with the overall organization, content and lab works presented during the Coursera IBM Certification Course.
I feel this Capstone project was a great opportunity to practice and apply the Data Science tools and methodologies learned.
I feel I have acquired skills to become a professional Data Scientist and will continue exploring.

6.CONCLUSION

This project has given me insight on how to solve a real life problem.
The mapping with Folium is a very powerful technique to consolidate information, analyze and make a decision confidently.