**Coursera Capstone Project**

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

**Upload Libraries Required**

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 # 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.')

Fetching package metadata .............

Solving package specifications: .

Package plan for installation in environment /opt/conda/envs/DSX-Python35:

The following NEW packages will be INSTALLED:

geographiclib: 1.49-py\_0 conda-forge

geopy: 1.18.1-py\_0 conda-forge

geographiclib- 100% |################################| Time: 0:00:00 23.05 MB/s

geopy-1.18.1-p 100% |################################| Time: 0:00:00 32.12 MB/s

Fetching package metadata .............

Solving package specifications: .

Package plan for installation in environment /opt/conda/envs/DSX-Python35:

The following NEW packages will be INSTALLED:

altair: 2.2.2-py35\_1 conda-forge

branca: 0.3.1-py\_0 conda-forge

folium: 0.5.0-py\_0 conda-forge

vincent: 0.4.4-py\_1 conda-forge

altair-2.2.2-p 100% |################################| Time: 0:00:00 54.50 MB/s

branca-0.3.1-p 100% |################################| Time: 0:00:00 28.22 MB/s

vincent-0.4.4- 100% |################################| Time: 0:00:00 30.15 MB/s

folium-0.5.0-p 100% |################################| Time: 0:00:00 37.70 MB/s

Libraries imported.

**Coursera Capstone - REPORT**

**Content**

1. Introduction Section :

1.1 Discussion of the "backgroung situation" leading to the problem at hand:

1.2 Problem to be resolved

1.3 Audience for this project.

1. Data Section:

2.1 Data of Current Situation (current residence place)

2.2 Data required to resolve the problem

2.3 Data sources and data manipulation

1. Methodology section :

3.1 Process steps and strategy to resolve the problem

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

1. Results section

Discussion of the results and how they help to take a decision.

1. Discussion section

Elaboration and discussion on any observations and/or recommendations for improvement.

1. Conclusion section

Desicison taken and Report Conclusion.

**1. Introduction Section :**

**Discussion of the business problem and the audience who would be interested in this project.**

**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 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 Interested Audience**

I believe this is a relevant project for a person or entity considering moving to a major city in Europe, US or Asia, since the approach and methodologies used here are applicable in all cases. The use of FourSquare data and mapping techniques combined with data analysis will help resolve the key questions arisen. Lastly, this project is a good practical case toward the development of Data Science skills.

**2. Data Section:**

**Description of the data and its sources that will be used to solve the problem**

**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

**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.

mh\_neigh\_data.tail():

Borough Neighborhood Latitude Longitude

35 Manhattan Turtle Bay 40.752042 -73.967708

36 Manhattan Tudor City 40.746917 -73.971219

37 Manhattan Stuyvesant Town 40.731000 -73.974052

38 Manhattan Flatiron 40.739673 -73.990947

39 Manhattan Hudson Yards 40.756658 -74.000111

A list of Manhattan subway metro stops was complied in Numbers (Apple excel) and it was complemeted with wikipedia data ( <https://en.wikipedia.org/wiki/List_of_New_York_City_Subway_stations_in_Manhattan>) and information from NY Transit authority and Google maps (<https://www.google.com/maps/search/manhattan+subway+metro+stations/@40.7837297,-74.1033043,11z/data=!3m1!4b1>) for a final consolidated list of subway stops names and their address. The geolocation was obtained via an algorythm using Nominatim. Details will be shown in the execution of methodolody in section 3.0. The subway csv file is ''MH\_subway.csv'" and the data structure is: mhsub.tail(): 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

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=CjwKCAiAjNjgBRAgEiwAGLlf2hkP3A-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.

**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.

**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

**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.

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

* what is the cost of available rental places that meet the demands?
* what is the cost of rent 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 rental 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.

**METHODOLOY EXECUTION - Mapping Data**

**Singapore Map - Current residence and venues in neighborhood**

for comparison to future Manhattan renting place

In [13]:

# 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))

/opt/conda/envs/DSX-Python35/lib/python3.5/site-packages/ipykernel/\_\_main\_\_.py:3: DeprecationWarning: Using Nominatim with the default "geopy/1.18.1" `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.

app.launch\_new\_instance()

The geograpical coordinate of Singapore home are 1.2792655, 103.8480938.

In [14]:

neighborhood\_latitude=1.2792655

neighborhood\_longitude=103.8480938

**Dial FourSquare to find venues around current residence in Singapore**

In [15]:

# The code was removed by Watson Studio for sharing.

In [16]:

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[16]:

'https://api.foursquare.com/v2/venues/explore?&client\_id=DVCTZDPDYXTS0BRJFPLMHM323APGXNWZI5PLRQ1VC0CFLF1T&client\_secret=5NWAGXRLXIXAV0L3DNYY1EPIHNMAAAIZFDFELYSYXL5LFWL1&v=20180605&ll=1.2792655,103.8480938&radius=500&limit=100'

In [17]:

# results display is hidden for report simplification

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

#results

**function that extracts the category of the venue - borrow from the Foursquare lab.**

In [18]:

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 [19]:

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

Out[19]:

(100, 4)

In [20]:

# Venues near current Singapore residence place

SGnearby\_venues.head(10)

Out[20]:

|  | **name** | **categories** | **lat** | **lng** |
| --- | --- | --- | --- | --- |
| **0** | Napoleon Food & Wine Bar | Wine Bar | 1.279925 | 103.847333 |
| **1** | Park Bench Deli | Deli / Bodega | 1.279872 | 103.847287 |
| **2** | Native | Cocktail Bar | 1.280135 | 103.846844 |
| **3** | Muchachos | Burrito Place | 1.279175 | 103.847082 |
| **4** | Matt's | The Chocolate Shop | Dessert Shop | 1.280462 | 103.846950 |
| **5** | Freehouse | Beer Garden | 1.281254 | 103.848513 |
| **6** | PS.Cafe | Café | 1.280468 | 103.846264 |
| **7** | 왕대박 Wang Dae Bak Korean BBQ Restaurant | Korean Restaurant | 1.281345 | 103.847551 |
| **8** | Ancient Therapy | Massage Studio | 1.280413 | 103.847481 |
| **9** | Oven & Fried Chicken | Korean Restaurant | 1.280479 | 103.847522 |

**Map of Singapore residence place with venues in Neighborhood - for reference**

In [21]:

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

Out[21]:

**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 [22]:

# 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-22-6a2b7dc1163d> 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()

/opt/conda/envs/DSX-Python35/lib/python3.5/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, nrows, na\_values, keep\_default\_na, na\_filter, verbose, skip\_blank\_lines, parse\_dates, infer\_datetime\_format, keep\_date\_col, date\_parser, dayfirst, iterator, chunksize, compression, thousands, decimal, lineterminator, quotechar, quoting, escapechar, comment, encoding, dialect, tupleize\_cols, error\_bad\_lines, warn\_bad\_lines, skipfooter, skip\_footer, doublequote, delim\_whitespace, as\_recarray, compact\_ints, use\_unsigned, low\_memory, buffer\_lines, memory\_map, float\_precision)

703 skip\_blank\_lines=skip\_blank\_lines)

704

--> 705 return \_read(filepath\_or\_buffer, kwds)

706

707 parser\_f.\_\_name\_\_ = name

/opt/conda/envs/DSX-Python35/lib/python3.5/site-packages/pandas/io/parsers.py in \_read(filepath\_or\_buffer, kwds)

443

444 # Create the parser.

--> 445 parser = TextFileReader(filepath\_or\_buffer, \*\*kwds)

446

447 if chunksize or iterator:

/opt/conda/envs/DSX-Python35/lib/python3.5/site-packages/pandas/io/parsers.py in \_\_init\_\_(self, f, engine, \*\*kwds)

812 self.options['has\_index\_names'] = kwds['has\_index\_names']

813

--> 814 self.\_make\_engine(self.engine)

815

816 def close(self):

/opt/conda/envs/DSX-Python35/lib/python3.5/site-packages/pandas/io/parsers.py in \_make\_engine(self, engine)

1043 def \_make\_engine(self, engine='c'):

1044 if engine == 'c':

-> 1045 self.\_engine = CParserWrapper(self.f, \*\*self.options)

1046 else:

1047 if engine == 'python':

/opt/conda/envs/DSX-Python35/lib/python3.5/site-packages/pandas/io/parsers.py in \_\_init\_\_(self, src, \*\*kwds)

1682 kwds['allow\_leading\_cols'] = self.index\_col is not False

1683

-> 1684 self.\_reader = parsers.TextReader(src, \*\*kwds)

1685

1686 # XXX

pandas/\_libs/parsers.pyx in pandas.\_libs.parsers.TextReader.\_\_cinit\_\_()

pandas/\_libs/parsers.pyx in pandas.\_libs.parsers.TextReader.\_setup\_parser\_source()

FileNotFoundError: File b'mh\_neigh\_data.csv' does not exist

In [ ]:

manhattan\_data.tail()

**Manhattan Borough neighborhoods - data with top 10 clustered venues**

In [23]:

manhattan\_merged = pd.read\_csv('manhattan\_merged.csv')

manhattan\_merged.head()

---------------------------------------------------------------------------

FileNotFoundError Traceback (most recent call last)

<ipython-input-23-e28a63508e70> in <module>()

----> 1 manhattan\_merged = pd.read\_csv('manhattan\_merged.csv')

2 manhattan\_merged.head()

/opt/conda/envs/DSX-Python35/lib/python3.5/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, nrows, na\_values, keep\_default\_na, na\_filter, verbose, skip\_blank\_lines, parse\_dates, infer\_datetime\_format, keep\_date\_col, date\_parser, dayfirst, iterator, chunksize, compression, thousands, decimal, lineterminator, quotechar, quoting, escapechar, comment, encoding, dialect, tupleize\_cols, error\_bad\_lines, warn\_bad\_lines, skipfooter, skip\_footer, doublequote, delim\_whitespace, as\_recarray, compact\_ints, use\_unsigned, low\_memory, buffer\_lines, memory\_map, float\_precision)

703 skip\_blank\_lines=skip\_blank\_lines)

704

--> 705 return \_read(filepath\_or\_buffer, kwds)

706

707 parser\_f.\_\_name\_\_ = name

/opt/conda/envs/DSX-Python35/lib/python3.5/site-packages/pandas/io/parsers.py in \_read(filepath\_or\_buffer, kwds)

443

444 # Create the parser.

--> 445 parser = TextFileReader(filepath\_or\_buffer, \*\*kwds)

446

447 if chunksize or iterator:

/opt/conda/envs/DSX-Python35/lib/python3.5/site-packages/pandas/io/parsers.py in \_\_init\_\_(self, f, engine, \*\*kwds)

812 self.options['has\_index\_names'] = kwds['has\_index\_names']

813

--> 814 self.\_make\_engine(self.engine)

815

816 def close(self):

/opt/conda/envs/DSX-Python35/lib/python3.5/site-packages/pandas/io/parsers.py in \_make\_engine(self, engine)

1043 def \_make\_engine(self, engine='c'):

1044 if engine == 'c':

-> 1045 self.\_engine = CParserWrapper(self.f, \*\*self.options)

1046 else:

1047 if engine == 'python':

/opt/conda/envs/DSX-Python35/lib/python3.5/site-packages/pandas/io/parsers.py in \_\_init\_\_(self, src, \*\*kwds)

1682 kwds['allow\_leading\_cols'] = self.index\_col is not False

1683

-> 1684 self.\_reader = parsers.TextReader(src, \*\*kwds)

1685

1686 # XXX

pandas/\_libs/parsers.pyx in pandas.\_libs.parsers.TextReader.\_\_cinit\_\_()

pandas/\_libs/parsers.pyx in pandas.\_libs.parsers.TextReader.\_setup\_parser\_source()

FileNotFoundError: File b'manhattan\_merged.csv' does not exist

**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 [ ]:

# 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

**Examine a paticular Cluster - print venues**

**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 .**

**Assign a value to 'kk' to explore a given cluster.**

In [ ]:

## 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]))]]

**Map of Manhattan places for rent**

**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.**

**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 [ ]:

# 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()

In [ ]:

mh\_rent.tail()

**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 [ ]:

## This section may be 'markedown' for the report because its execution takes few minutes .

## 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

In [ ]:

mh\_rent=pd.read\_csv('MH\_rent\_latlong.csv')

mh\_rent.head()

In [ ]:

mh\_rent.tail()

**Manhattan apartment rent price statistics**

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

In [ ]:

import seaborn as sns

sns.distplot(mh\_rent['Rent\_Price'],bins=15)

In [ ]:

import seaborn as sns

sns.distplot(mh\_rent['Price\_per\_ft2'],bins=15)

In [ ]:

sns.boxplot(x='Rooms', y= 'Rent\_Price', data=mh\_rent)

**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 appartment with the price condition estipulated (max US7000 )**

In [ ]:

# 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

**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 possibilites**

In [ ]:

# 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

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

**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 [ ]:

## 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]))]]

**Mapping Manhattan Subway locations**

**Manhattan subway metro locations ( address) was obtained from webscrapping 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 obtain via Nominatim using the algorythm below.**

In [ ]:

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

In [ ]:

# 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 [ ]:

## Algorythm to find latitude and longitud for each subway metro station and add them to 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

In [ ]:

mh=pd.read\_csv('MH\_subway.csv')

print(mh.shape)

mh.head()

In [ ]:

# removing duplicate rows and creating new set mhsub1

mhsub1=mh.drop\_duplicates(subset=['lat','long'], keep="last").reset\_index(drop=True)

mhsub1.shape

In [ ]:

mhsub1.tail()

**MAP of Manhattan showing the location of subway stations**

In [ ]:

# 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

**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 [ ]:

mh\_rent.head()

In [ ]:

# 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

**4.0 Results**

**ONE CONSOLIDATE MAP**

**Let's consolidate all the required inforamtion 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 [ ]:

# 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

**Problem Resolution**

**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

**Venues for Apartment 1 - Cluster 2**

In [ ]:

## 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]))]]

In [ ]:

## Venues for Apartment 2 - Cluster 3

In [ ]:

## 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]))]]

**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 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. 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 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.**

**5.0 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.**

**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.**

In [ ]:

**End of Project and Course/ Thanks to Coursera Team and Students!. Happy 2019!!**