IBM CAPSTONE PROJECT REPORT
THE BATTLE OF THE NEIGHBORHOODS (WEEK 1 & WEEK 2)

Background and Introduction

Problem Description

In today's society there are many ways to shop for just about everything. The ways include online shopping, in person brick and mortar, and there are services where businesses provide personnel who will go and shop for the specific items their customers want. Often times it is difficult for a personal shopper or an individual to buy things if they do not know what they specifically want. Then if you factor in an enormous amount of choices and the challenge only escalates. Even with today's computerized recommender systems that can make recommendations based upon a profile, these systems have their weaknesses as well. Simply put, the problem is that there are too many choices, and the overwhelming number of choices makes it very difficult for people and smart machines to decide and decide well. According to this NYTimes article https://www.nytimes.com/2010/02/27/your-money/27shortcuts.html "Research also shows that an excess of choices often leads us to be less, not more, satisfied once we actually decide. There's often that nagging feeling we could have done better."

A city like New York is an enormous city and many times when visitors to the city are looking for a hotel in NYC the choices can be overwhelming. So not only is the problem of choosing a hotel extremely difficult but finding the right area to stay in adds to the complexity. Often times many tourists that visit NYC don't know anything about the city, but they can often, and without hesitation tell you what it is that they like to do. They can very easily share the types of experiences they like to have, the venues they visit the most, etc. I believe that when a customer tells you what they like to do, therein lies the solution to part of the problem of having too many choices. If you can tell me what you like to do, I can help you determine where you would best like to stay and give you options that target the types of venues you most frequent when traveling. I feel that this solution leaves the buyer with less remorse in the long run.

What I am proposing is to use publicly available location data and Foursquare data to code a solution that pulls venue data from Foursquare and builds a dataframe that gives you the top ten venues by neighborhood. From there we can pull the foursquare data that has all of the hotels for that area you want to stay based on your tastes in venues. This would simplify the arduous task of picking a hotel from the 669 hotels that are currently available in NYC.

The target audience for this type of solution is the traveler who is easily overwhelmed by the number of lodging options in NYC or any large city destination for that matter. This solution gives a travel a more focused solution that starts with what types of venues they are attracted to and offer the areas, and hotels with the highest concentrations of those venues. From there the traveler can select the area they are most attracted to and then the solution would be to give them a list of nearby hotels. The ultimate goal is to simplify the process of choosing a hotel, lower the travelers stress and make the trip more enjoyable.

Methodology

1. Collect the data
2. Inspect/Review/Understand the data
3. Prepare the data
4. Code and visualize the solution

1. Data Collection

The environment that I will work from for the data analysis and coding of this solution will be a Jupyter Notebook. While utilizing Python as the scripting language, I will leverage the many different Python libraries that handle JSON data, Dataframes, and mapping objects. The Dataframes will mostly contain location data that is extracted through the Foursquare API and publicly available location data. The publicly available New York City location data I will begin with comes from the following link. **https://geo.nyu.edu/catalog/nyu_2451_34572** Foursquare is a repository of location data that it obtains through interacting with its partners like Snap, Twitter, Google etc. The users of those mobile applications are sending location data back to Foursquare and Foursquare becomes an astounding repository of personal location data as well as other location data for businesses, geographic locations, etc. The Foursquare database is rich with comprehensive location data and it will provide the remainder of the data that we need for this evaluation.

The primary method for retrieving this data will be through the Foursquare RESTful API.
An example query statement through the Foursquare API is below:

https://api.foursquare.com/v2/venues/search?client_id=CLIENT_ID&client_secret=CLIENT_SECRET&ll=LATITUDE,LONGITUDE&v=VERSION&query=QUERY&radius=RADIUS&limit=LIMIT

This single line of code above allows us to pass our credentials and parameters for the query in one contiguous line of code. With a successful response we can code a GET request to extract the JSON data from the results of the query. The results are in. JSON format and will appear as such.

```
{'meta': {'code': 200, 'requestId': '5f1e0504acc3b6486dd91655'},
 'response': {'venues': [{'id': '56d8c0f8498edb854f926e6a',
    'name': 'The Beekman, A Thompson Hotel',
    'location': {'address': '123 Nassau St',
     'crossStreet': 'Beekman St',
     'lat': 40.7111725,
     'lng': -74.0067017,
     'labeledLatLngs': [{'label': 'display',
       'lat': 40.7111725,
       'lng': -74.0067017}],
     'distance': 182,
     'postalCode': '10038',
     'cc': 'US',
     'city': 'New York',
```

```
 'state': 'NY',
 'country': 'United States',
 'formattedAddress': ['123 Nassau St (Beekman St)',
  'New York, NY 10038',
  'United States']},
'categories': [{'id': '4bf58dd8d48988d1fa931735',
  'name': 'Hotel',
  'pluralName': 'Hotels',
  'shortName': 'Hotel',
```

2. Data Inspection / Data Review / Understanding the Data

The data extracted will be in numerous lines of JSON formatted data and we can transform the data into Pandas dataframes which make for reviewing and understanding the data much easier. Working with JSON formatted data can be a bit of a challenge for those that are less familiar but Python helps simplify this. We first assign a variable called neighborhood_info that pulls the features dictionary from the broader dataset.

Data review and inspection process

```
In [18]:
1 #printing the first item to inspect its features
2 neighborhoods_info[0]

Out[18]: {'type': 'Feature',
 'id': 'nyu_2451_34572.1',
 'geometry': {'type': 'Point',
  'coordinates': [-73.84720052054902, 40.89470517661]},
 'geometry_name': 'geom',
 'properties': {'name': 'Wakefield',
  'stacked': 1,
  'annoline1': 'Wakefield',
  'annoline2': None,
  'annoline3': None,
  'annoangle': 0.0,
  'borough': 'Bronx',
  'bbox': [-73.84720052054902,
   40.89470517661,
   -73.84720052054902,
   40.89470517661]}}
```

The first item in the neighborhoods_info variable is a subset of the data that contains the information for the Wakefield neighborhood in the Bronx. Next we will create a Pandas dataframe that will have columns for borough, neighborhood, latitude, and longitude. We will use the Pandas library DataFrame method for this.

Calling this dataframe neighborhoods, we can display the neighborhoods dataframe with the .head() method, and see what the populated dataframe looks like.

```
1 neighborhoods.head()
```

	Borough	Neighborhood	Latitude	Longitude
0	Bronx	Wakefield	40.894705	-73.847201
1	Bronx	Co-op City	40.874294	-73.829939
2	Bronx	Eastchester	40.887556	-73.827806
3	Bronx	Fieldston	40.895437	-73.905643
4	Bronx	Riverdale	40.890834	-73.912585

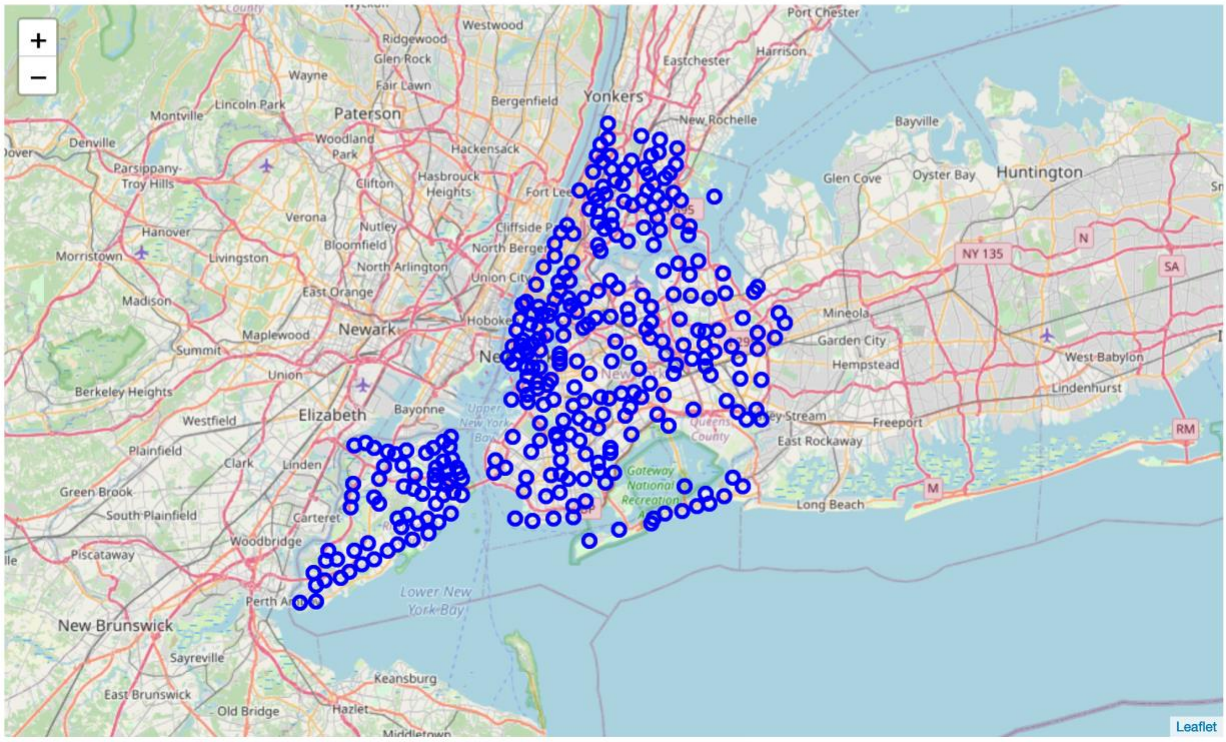
Now that we have an established, structured dataframe with the required data such as borough name, neighborhood name, latitude and longitude we need to be certain that the dataframe is comprehensive enough. Since we loaded a dataset that was described as containing the information that we needed, we still need to continue investigating that this in fact true. This is where the Folium library mapping functions is very useful.

Folium makes it easy to visualize data imported into your developer environment on an interactive map. The map below is the visualization that we will use to confirm that our data is in fact good enough to continue the analysis with the data we initially imported.

```
# create map of Manhattan using latitude and longitude values
map = folium.Map(location=[latitude1, longitude1], zoom_start=11.5)

# add markers to map
for lat, lng, label in zip(neighborhoods['Latitude'], neighborhoods['Longitude'], neighborhoods['Neighborhood']):
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius=5,
        popup=label,
        color='blue',
        fill=False,
        #fill_color='#3186cc',
        fill_opacity=0.7,
        parse_html=False).add_to(map)

map
```



Instantiating a map object with Latitude and Longitude from our Nominatim object plus an iteration over our neighborhoods dataframe allows us to plot all of the neighborhoods in NYC

across the 5 boroughs. The initial data set is validated as adequate enough to continue the process.

3. Prepare the Data for Further Analysis

The original data set and neighborhood dataframe gives us a great point of reference for a starting point. The unique thing about New York City is that it is a very diverse city. Each borough has its own identity, and each neighborhood has its own unique identity as well. It's true that no two neighborhoods are exactly alike. Therefore, we want to prepare the data for further analysis where we will look to determine what the best venues are based on the borough that a traveler would want to stay in.

For this example, we are going to select Brooklyn as the borough that we are choosing to stay in. My initial reason for selecting Brooklyn is because Brooklyn is very diverse. A borough like Brooklyn is going to have a very diverse set of venues as well. So, our next steps will be to gather, aggregate and rank those venues by neighborhood before we go to the final steps. The step below is where we define a new dataframe named brooklyn that is a subset of our original neighborhoods dataframe.

We need to prepare the data for modeling our solution

Since we are targeting NYC, we will choose Brooklyn as our test location and look to evaluate the venues in the different neighborhoods of Manhattan. Simply put all the worlds best rappers came from Brooklyn so we want to analyze the venues in Brooklyn.

```
#creating a dataframe that is specific to Brooklyn
brooklyn = neighborhoods[neighborhoods['Borough'] == 'Brooklyn'].reset_index(drop=True)
brooklyn.head()
```

After coding a function that has a Foursquare API call we can then continue the data preparation process and generate a dataframe that has all of the brooklyn venues that were returned from the Foursquare API call.

```
#assessing the size and shape of the new Brooklyn Venues dataframe
print(brooklyn_venues.shape)
brooklyn_venues.head()
```

```
(4812, 7)
```

Our brooklyn_venues data frame has 4,812 rows and 7 columns of data.

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Bay Ridge	40.625801	-74.030621	Pilo Arts Day Spa and Salon	40.624748	-74.030591	Spa
1	Bay Ridge	40.625801	-74.030621	Bagel Boy	40.627896	-74.029335	Bagel Shop
2	Bay Ridge	40.625801	-74.030621	Cocoa Grinder	40.623967	-74.030863	Juice Bar
3	Bay Ridge	40.625801	-74.030621	Pegasus Cafe	40.623168	-74.031186	Breakfast Spot
4	Bay Ridge	40.625801	-74.030621	Leo's Casa Calamari	40.624200	-74.030931	Pizza Place

This is ample data to continue the process. After several more coding steps we will create a structured table that presents each neighborhood along with the associated most common venues. By reviewing this table, we can start to choose which neighborhoods fit our tastes. Some neighborhoods will have a variety of venues that appeal to a traveler and some neighborhoods will be less appealing.

	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 Most Common Venue	8th Most Common Venue	9th Most Common Venue	10th Most Common Venue
0	Bath Beach	Pharmacy	Bubble Tea Shop	Chinese Restaurant	Fast Food Restaurant	Italian Restaurant	German Restaurant	Cantonese Restaurant	Surf Spot	Sushi Restaurant	Donut Shop
1	Bay Ridge	Pizza Place	Italian Restaurant	Grocery Store	Greek Restaurant	Spa	Breakfast Spot	Sports Bar	Taco Place	Caucasian Restaurant	Bookstore
2	Bedford Stuyvesant	Coffee Shop	Deli / Bodega	Pizza Place	Bar	Café	Fried Chicken Joint	Playground	New American Restaurant	Cocktail Bar	Park
3	Bensonhurst	Sushi Restaurant	Donut Shop	Chinese Restaurant	Ice Cream Shop	Italian Restaurant	Butcher	Supermarket	Spa	Flower Shop	Shabu-Shabu Restaurant
4	Bergen Beach	Harbor / Marina	Athletics & Sports	Hockey Field	Playground	Baseball Field	Women's Store	Ethiopian Restaurant	Event Service	Event Space	Factory
5	Boerum Hill	Furniture / Home Store	Yoga Studio	Spa	Coffee Shop	Bar	Japanese Restaurant	Burrito Place	Seafood Restaurant	Middle Eastern Restaurant	Sandwich Place
6	Borough Park	Bank	Pharmacy	Café	Pizza Place	Fast Food Restaurant	Hotel	Bakery	Restaurant	Coffee Shop	Chinese Restaurant
7	Brighton Beach	Restaurant	Sushi Restaurant	Eastern European Restaurant	Gourmet Shop	Russian Restaurant	Supplement Shop	Korean Restaurant	Supermarket	Beach	Food & Drink Shop
8	Broadway Junction	Donut Shop	Diner	Fried Chicken Joint	Bakery	Pizza Place	Burger Joint	Ice Cream Shop	Gas Station	Discount Store	Seafood Restaurant
9	Brooklyn Heights	Yoga Studio	Pet Store	Coffee Shop	Playground	Deli / Bodega	Pizza Place	Cosmetics Shop	Diner	Japanese Restaurant	Scenic Lookout
10	Brownsville	Moving Target	Fried Chicken Joint	Chinese Restaurant	Pizza Place	Restaurant	Playground	Spanish Restaurant	Burger Joint	Park	Performing Arts Venue
11	Bushwick	Bar	Mexican Restaurant	Coffee Shop	Thrift / Vintage Store	Italian Restaurant	Café	Mediterranean Restaurant	French Restaurant	Nightclub	Sandwich Place
12	Canarsie	Caribbean Restaurant	Gym	Home Service	Deli / Bodega	Food	Chinese Restaurant	Bus Line	Asian Restaurant	Event Space	Factory
13	Carroll Gardens	Coffee Shop	Italian Restaurant	Spa	Bar	Cosmetics Shop	Farmers Market	Beer Garden	Latin American Restaurant	Gaming Cafe	French Restaurant
14	City Line	Donut Shop	Mobile Phone Shop	Grocery Store	Shoe Store	Fried Chicken Joint	Pharmacy	Clothing Store	Fast Food Restaurant	Bus Stop	Sandwich Place
15	Clinton Hill	Thai Restaurant	Yoga Studio	Wine Shop	Italian Restaurant	Restaurant	Pet Store	Sculpture Garden	Sandwich Place	Chinese Restaurant	Pub
16	Cobble Hill	Italian Restaurant	Yoga Studio	Cocktail Bar	Bar	Men's Store	French Restaurant	Bookstore	Boutique	Spanish Restaurant	Spa
17	Coney Island	Baseball Stadium	Park	Monument / Landmark	Food Court	Beach	Caribbean Restaurant	Skating Rink	Music Venue	Theme Park Ride / Attraction	Pharmacy
18	Crown Heights	Pizza Place	Museum	Café	Playground	Bagel Shop	Supermarket	Fried Chicken Joint	Candy Store	Liquor Store	Burger Joint
19	Cypress Hills	Latin American Restaurant	Metro Station	Pizza Place	Fried Chicken Joint	Ice Cream Shop	Fast Food Restaurant	Spanish Restaurant	Food	Supermarket	Mexican Restaurant
20	Ditmas Park	Donut Shop	Pizza Place	Caribbean Restaurant	Farmers Market	Ramen Restaurant	Mobile Phone Shop	Mexican Restaurant	Latin American Restaurant	Japanese Restaurant	Health Food Store

This is very helpful because we can see that if you are a coffee junkie some neighborhoods have lots of coffee shops or if Latin American Restaurants are a must then you might consider staying in Cypress hill.

4. Code and visualize the solution

In the final process we first need to choose a neighborhood in the borough that we think suits our tastes the most. For this exercise I am selecting Crown Heights, Brooklyn, NY. Known for its pizza places, museums and cafes, Crown Heights Brooklyn is where I want to be.

This section is where we select the neighborhood with the venues that match our taste and then look to retrieve the list of hotels that are in that area.

```
#
venue_neighborhood = 'Crown Heights, Brooklyn, NY'

geolocator2 = Nominatim(user_agent = 'brooklyn_agent')
location2 = geolocator2.geocode(venue_neighborhood)
latitude2 = location2.latitude
longitude2 = location2.longitude
print(latitude2, longitude2)
```

```
40.6688226 -73.9311116
```

One more Nominatim instantiation to get new latitudes and longitudes for Crown Heights and we can again use Foursquare to retrieve a data set that is filled with hotels in an around Crown Heights. The parameters utilized a 10,000 m radius, and limited the search to 10 hotels.

```
dataframe_filtered.name
```

```
0                              Hotel RL Brooklyn
1                  Best Western Plus Arena Hotel
2                            The High Line Hotel
3                               Soho Grand Hotel
4                                   Hotel Indigo
5                            Hotel 50 Bowery NYC
6                    The Best Exotic Utica Hotel
7                              Metro Court Hotel
8                                  Hotel Delmano
9    Trump International Hotel & Tower® New York
Name: name, dtype: object
```

Now that we have a dataframe that is cleaned and prepped to include nearby hotels it's much easier to have a starting reference point when choosing a place to stay. For those of us that are familiar with Brooklyn and NYC, we can see that some of the hotels in this list not in Brooklyn. That is due to the length of our 10km search radius. These parameters can be adjusted and can be selectable to suit the needs of the user.

The section below details the final results. We created another interactive map that will combine our desired destination based on its venue parameters and the hotels that are nearby.

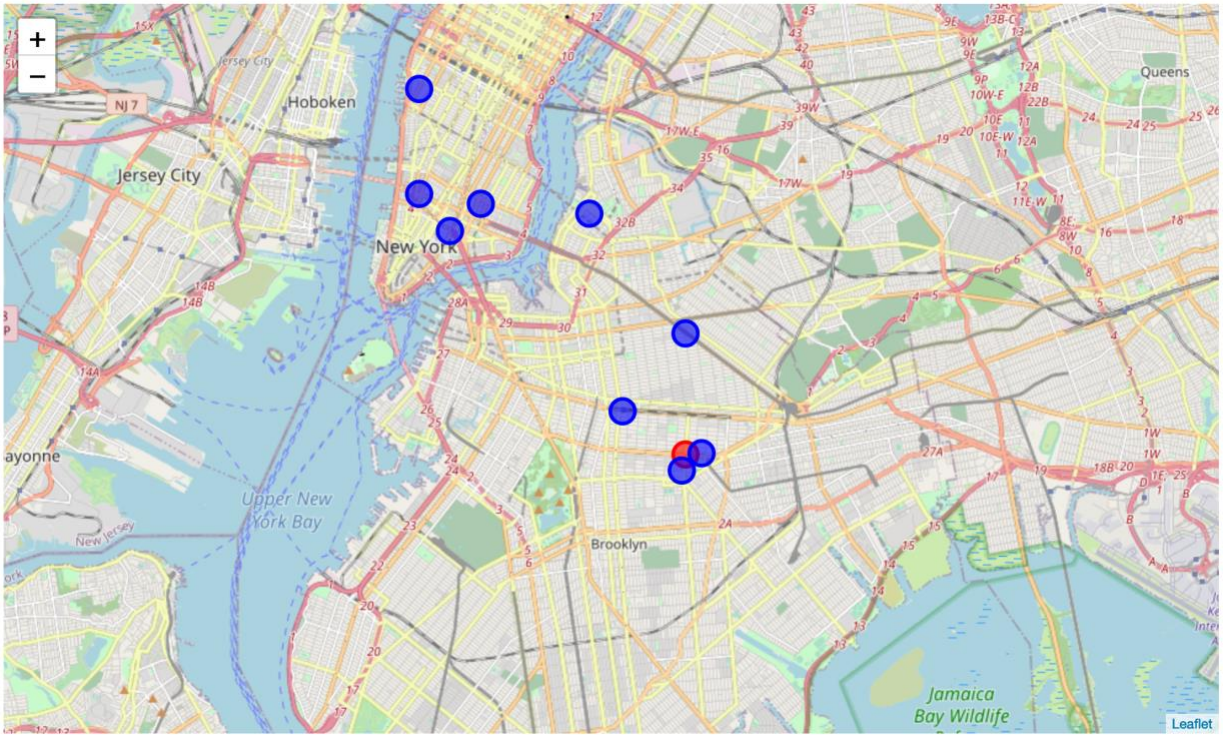
```python
venues_map = folium.Map(location=[latitude2, longitude2], zoom_start=13) # generate map centred around the desired

# add a red circle marker to represent the centroid of our desired destination
folium.features.CircleMarker(
    [latitude2, longitude2],
    radius=10,
    color='red',
    popup='Desired Destination',
    fill = True,
    fill_color = 'red',
    fill_opacity = 0.6
).add_to(venues_map)

# add the hotels as blue circle markers
for lat, lng, label in zip(dataframe_filtered.lat, dataframe_filtered.lng, dataframe_filtered.categories):
    folium.features.CircleMarker(
        [lat, lng],
        radius=10,
        color='blue',
        popup=label,
        fill = True,
        fill_color='blue',
        fill_opacity=0.6
    ).add_to(venues_map)

# display map
venues_map
```

5. Results and discussion



We set out to discover a path for making travel plans a little simpler. We started with an understanding that there are far too many choices available to us as shoppers and travelers to comfortably make decisions that don't require a lot of research, or result in a decision that produces the fear of missing out. We also set out to make this solution something we could craft using Python and the principles of Data Science.

We were able to use publicly available data to create dataframes that were comprehensive with respect to location data in a very complex city like New York City. We were able to manipulate that data using publicly available tools like Python and its associated libraries. Foursquare was an extremely valuable tool because we able to add to our dataset with comprehensive data that was publicly available.

In the image above the red circle depicts the neighborhood where we would like to stay in Brooklyn. The blue circles represent hotels that are nearest to our desired venue according our search through the Foursquare API. It would certainly be useful to check this map against maps available through popular travel internet sites and google as their resources are far more developed and robust. The unique feature that this solution offers is that it aggregates venues per neighborhood, and ranks them. This gives us a powerful tool because it allows our target user to pick an area based on venue statistics, and then a hotel afterwards.

Our initial data science strategy has proven that we can be successful in attempting to solve our initial problem and bring a solution to our target audience. This notebook can use further development to enhance some of the visualizations and make the maps more interactive but that is an opportunity for further development.