CAPSTONE PROJECT

Table of Contents

CAPSTONE PROJECT ***1***

Introduction ***2***

Process ***2***

Foursquare Process ***2***

Challenges with DC Geography ***3***

Mapping and Plotting Data ***4***

Commuting in DC ***5***

Final Outcome ***5***

Introduction

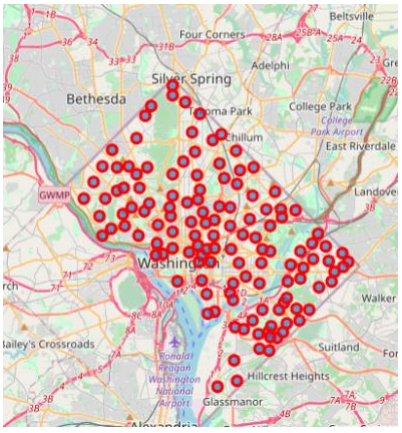
My Battle of the Neighborhood will take place in DC. I am going to identify neighborhoods that would be ideal for a co-working space. I am going to use FourSquare to organize neighborhood amenities, data from WMATA (DC Area Mass Transit), DC Government KML files, and a location file with 13 distinct locations in DC (Named using the Greek Alphabet Alpha to Nu). In the end, I am going to rank order the locations based on access to Metro, Recreation, Restaurants, Services, Medical, and Education. Given life in DC, I am weighting Metro as the most valuable, followed by restaurants, than education. The remaining categories will receive no weighting. So this isn't just a battle of the neighborhoods, but a battle for 13 unique locations in DC (representing many different geographic groupings). LET THE BATTLE BEGIN!

Process

I am using a mix of local files, KML and Source information from the DC Government, Mass Transit API, and QGIS Software to identify the best location for a Co-working spot in DC. Overall, there are four steps in the process

1. Load local data
2. Load Foursquare data
3. Load WMATA data
4. Merge and clean the three data sources and rank the 13 unique DC locations

Foursquare Process



First, I needed to give Foursquare a geographic point of reference. I used the center of DC (The US Capitol, well close to that) and mapped my DC neighborhood file.

This allowed, me to get JSON data centered around the right latitude and longitude

```
In [12]: url='https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={},{}&
             CLIENT_ID,
             CLIENT_SECRET,
             VERSION,
             dclat,
             dclong,
             radius,
             Limit)
         url

Out[12]: 'https://api.foursquare.com/v2/venues/explore?&client_id=
         OUFGTV1V1PFQ&client_secret=
         85565773,-76.98034771&radius=500&limit=100'

In [13]:
         results = requests.get(url).json()
         results

Out[13]: {'meta': {'code': 200, 'requestId': '5e9b2d43c546f3001b2ba030'},
          'response': {'headerLocation': 'Anacostia',
           'headerFullLocation': 'Anacostia, Washington',
           'headerLocationGranularity': 'neighborhood',
           'totalResults': 4,
           'suggestedBounds': {'ne': {'lat': 38.8601577345, 'lng': -76.97457984594331},
            'sw': {'lat': 38.851157725499995, 'lng': -76.98611557405671}},
           'groups': [{'type': 'Recommended Places',
             'name': 'recommended',
             'items': [{'reasons': {'count': 0,
                'items': [{'summary': 'This spot is popular',
                  'type': 'general',
                  'reasonName': 'globalInteractionReason'}]},
               'venue': {'id': '4b4cd6d6f964a52015c126e3',
```

After pulling the JSON data into a dataframe, I could start to use the data to figure out which neighborhood had the best features.

This resulted in a DF that provided me totals by business in 6 groups (Determined by me)

- Food
- Services
- Commute
- Medical
- Education
- Recreation

The picture below, shows the new DF that is much more manageable to use than the one to the right.

Out[24]:

	Neighborhood	Afghan Restaurant	American Restaurant	Antique Shop	Arcade	Arepa Restaurant	Art Gallery	Art Museum	Arts & Crafts Store	Arts & Entertainment	...	Wareh S
0	16th Street Heights	0	0	0	0	0	0	0	0	0	...	
1	Adams Morgan	1	0	0	0	0	1	0	0	0	...	
2	American University Park	0	0	0	0	0	0	0	0	0	...	
3	Arboretum	0	0	0	0	0	0	0	0	0	...	
4	Barnaby Woods	0	0	0	0	0	0	0	0	0	...	
...	...	...	...	...	...	...	...	...	...	...	...	
125	West End	0	3	0	0	0	0	0	0	0	...	
126	Woodland	0	0	0	0	0	1	0	0	0	...	
127	Woodland-Normanstone	0	0	0	0	0	0	0	0	0	...	
128	Woodley Park	0	0	0	0	0	0	0	0	0	...	
129	Woodridge	0	0	0	0	0	0	0	0	0	...	

130 rows × 300 columns

In [47]:
```
df_index_4=pd.merge(df_index_3, dfMed, on='Neighborhood')
df_index_4
```

Out[47]:

	Neighborhood	Resturant Total	Recreation Total	Services Total	Commuting Total	Education Total	Medical Total
0	16th Street Heights	8	4	1	1	0	0
1	Adams Morgan	33	8	9	0	0	0
2	American University Park	1	0	0	0	0	0
3	Arboretum	6	5	4	0	0	0
4	Barnaby Woods	1	2	0	0	0	0
...	...	...	...	...	...	...	...
125	West End	23	10	11	0	0	0
126	Woodland	1	4	0	0	0	0
127	Woodland-Normanstone	1	4	0	0	0	0
128	Woodley Park	7	8	5	1	0	1
129	Woodridge	2	0	4	0	0	0

130 rows × 7 columns

However, as I later discovered, the commuting DF is not very good, as it severely undercounts the options available. Later, I use the WMATA API to pull in better commuting data.

Challenges with DC Geography

My location file is attached to clusters, not Neighborhoods because the nature of DC's geography. DC is comprised of 4 sections (Northwest, Northeast, Southwest, Southeast) and 8 Wards(Ward1-Ward8). In addition, DC has many unique and changing Neighborhoods. The problem -- wards are not confined to sections and neighborhoods are not confined to wards. So a Neighborhood could be in multiple wards and sections. DC government provides many KML files to help people with the interesting geographic organization in the city, the problem, the KML file for Neighborhoods was not polygons, rather points. So I had no way of testing which neighborhood my location was in, BUT I could use the KML file for Cluster. Cluster is another way DC organizes its geography. I wanted to us python to test if my point is inside of a cluster, but ran into some issues. To save time, I used QGIS and was able to create two files dfNeihborhood and dfLocation, which mapped DC's neighborhoods to clusters and my locations to Clusters. Now I have a common point to connect the two.

Fun fact- the sections in DC are important for an address. As in some cases there are four different places with the same address, but different sections. Even more fun facts-- the 4 sections meet in the geographic center of the US Capitol. So the closer

each address is to the US Capitol, the closer it is to the duplicate address. Always check NW, NE, SW, SE to make sure you know!

Mapping and Plotting Data

The next step of the process is to identify, which neighborhood the 13 selected locations are in. A simple Google search could identify the results, but that is not Python. I went through months of working learning GIS and python and I am running into issues. To speed the process, I used QGIS, which allows me to check to see if a lat/long exist inside a polygon. Using QGIS, I was able to check to see if my location was in a specific neighborhood. As I continue to learn python, I am going to focus mostly on GIS and python task as that is most relevant to my work.

QGIS allowed me to create two files

- Dfneighborhood.csv
  - Adds the cluster assigned by the DC government to a neighborhood
- dfLocation.csv
  - Adds the cluster assigned by the DC government to my 13 locations

As I said, the DC geography is interesting. A lot of conflicting information, but it seems the most likely reason DC is set up the way it is for two reasons:

1) Originally, DC was part of Maryland and Virginia, with a small section reserved for the Federal Government
2) The section set aside for the Federal Government, was not designed for anyone to live there, just work

Leading to the Civil War, Virginia "took" its part of DC back and after the Civil War, Maryland gave up it's part. So modern DC is a collection of the original Federal and Maryland land. Because of this oddity, the lack the formal structure and rights other parts of the United States have (i.e., no Senator represents DC).

This was a very challenging aspect of the project. Ultimately, I ended up with a DF that allowed me to tally up the total amount of business associated with each of the 14 locations using the neighborhood file (dfneighborhood.csv) to assign foursquare data to each location (dfLocation.csv).

```
In [52]: #combine locations into single row
df_Final=df_index_7.groupby('Location').sum()
#Dropping Commuting total because the number is off-- instead I am going to add WMATA data inst
df_Final.drop(['Commuting Total'], axis=1, inplace=True)
df_Final
```

Out[52]:

Location	Resturant Total	Recreation Total	Services Total	Education Total	Medical Total
Alpha	7	3	2	0	0
Beta	28	17	12	0	1
Delta	26	15	18	0	1
Epsilon	26	5	2	0	0
Eta	32	8	10	0	1
Gamma	62	25	20	0	0
Iota	272	89	68	3	4
Kappa	38	15	31	0	1
Mu	25	9	16	0	2
Theta	10	5	8	0	0
Zeta	6	7	12	0	0

Commuting in DC

Nothing is more important to a DC area resident than commuting. Traffic is brutal and constant. Looking at the Foursquare data, the commuting totals were very small and I knew there was a vast network of metro stops. Fortunately for me. WMATA, which runs the trains in DC area, has a great API and is very open with the data. I connected my project to WMATA. While the resulting data was in JSON, the project already had code to transform JSON into a DF, which I resued to complete the task.

```
#Just looking at the data, I can see that the Commuting data points are off.
#There are a bunch of metro stations and even more buslines

#We are going to get a list of Metro stations from WMATA's free JSON connection

import http.client, urllib.request, urllib.parse, urllib.error, base64

headers = {
    # Request headers
    'api_key': [redacted]
}

params = urllib.parse.urlencode({
    # Request parameters
    'LineCode': '',
})

try:
    conn = http.client.HTTPSConnection('api.wmata.com')
    conn.request("GET", "/Rail.svc/json/jStations?%s" % params, "{body}", headers)
    response = conn.getresponse()
    json_data = response.read()
    print(json_data)
    conn.close()
except Exception as e:
    print("[Errno {0}] {1}".format(e.errno, e.strerror))
```

Final Outcome

I completed my project with df_Final_3. A DF that contains all my locations, with Foursquare data and WMATA data. I ranked each using weighting to ensure that commuting was the most important column and ranked and totaled my columns. Resulting in location IOTA as the winner for best location for a co-working space.

	Location	Street	City	State	Zip	Latitude	Longitude	Cluster Name	Resturant Total	Recreation Total	Services Total	Education Total
0	Beta	320 21st St NE	Washington	DC	20002	38.894234	-76.975460	Cluster 25	28	17	12	0
1	Gamma	1503 East Capitol Street, SE	Washington	DC	20003	38.889428	-76.983260	Cluster 26	62	25	20	0
2	Epsilon	711 N Street, NW	Washington	DC	20001	38.907529	-77.022662	Cluster 7	26	5	2	0
3	Zeta	2019 Rhode Island Ave, NE	Washington	DC	20018	38.928684	-76.974973	Cluster 22	6	7	12	0
4	Eta	510 Webster St., NW	Washington	DC	20011	38.943775	-77.020883	Cluster 18	32	8	10	0
5	Iota	1135 New Jersey Ave, NW	Washington	DC	20001	38.904889	-77.014154	Cluster 8	272	89	68	3
6	Kappa	4404 Wisconsin Ave, NW	Washington	DC	20016	38.946962	-77.079989	Cluster 11	38	15	31	0
7	Mu	6008 Georgia Ave, NW	Washington	DC	20011	38.963195	-77.028373	Cluster 17	25	9	16	0

In [73]:
```
df_Final3=df_Final3.set_index('FINAL')
df_Final3=df_Final3.sort_index()

print(df_Final3)
```

```
      Location                        Street        City State    Zip  \
FINAL
45.5      Zeta     2019 Rhode Island Ave, NE  Washington    DC  20018
52.0   Epsilon             711 N Street, NW  Washington    DC  20001
59.0        Mu          6008 Georgia Ave, NW  Washington    DC  20011
68.0       Eta           510 Webster St., NW  Washington    DC  20011
97.5      Beta                320 21st St NE  Washington    DC  20002
110.0    Kappa        4404 Wisconsin Ave, NW  Washington    DC  20016
128.0    Gamma  1503 East Capitol Street, SE  Washington    DC  20003
160.0     Iota       1135 New Jersey Ave, NW  Washington    DC  20001

       Latitude  Longitude Cluster Name  Resturant Total  Recreation Total  \
FINAL
45.5   38.928684 -76.974973   Cluster 22                6                 7
52.0   38.907529 -77.022662    Cluster 7               26                 5
59.0   38.963195 -77.028373   Cluster 17               25                 9
68.0   38.943775 -77.020883   Cluster 18               32                 8
97.5   38.894234 -76.975460   Cluster 25               28                17
110.0  38.946962 -77.079989   Cluster 11               38                15
128.0  38.889428 -76.983260   Cluster 26               62                25
160.0  38.904889 -77.014154    Cluster 8              272                89
```