Coursera Capstone

IBM Applied Data Science Capstone

Locating good Residential Building having better Indian restaurant option in the neighborhood in Washington DC

By: Sarika Rajeev

June 2019

Introduction/Business Problem

The objective of this capstone project is to come up with a real problem and select a location for analysis. We need to use location data from Foursquare.com. We require to solve the problem using data science methodology and machine learning techniques.

In our problem statement, we have a group of software engineers from India who are planning to live in Washington DC for three months. Their hiring company is paying all expenditure. So, they would need to find flats/apartments having a good rating. They are going to work outside India for the first time. So, it's desirable that they are located nearby an Indian restaurant (having a great number of likes) preferred by others.

The target audience for this project includes:

1. Any professional/outsourcing employees who are going to relocate for short-term in Washington DC.
2. Real-estate agents who are interested in locating the attractive neighborhood in Washington DC.
3. Anyone who has an interest in location-based Data Science Project.

The code in python with data analysis is available on my GitHub page:

https://github.com/sarikarajeev33/capstone-final-project/blob/master/capstone_final_project.ipynb

Data Section

1. For the data collection part, we have used Wikipedia to find out Washington DC neighborhood names and link for the cities:

   https://en.wikipedia.org/wiki/Neighborhoods_in_Washington,_D.C.

   We have collected the data and after observation, we have neglected/dropped the unwanted data. With the help of link of the cities, we have found the Coordinates (longitude and latitude) of the cities in the neighborhood of Washington DC.

```
url = "https://en.wikipedia.org/wiki/Neighborhoods_in_Washington,_D.C."
page = requests.get(url)
print(page.text[:500])
```

```
<!DOCTYPE html>
<html class="client-nojs" lang="en" dir="ltr">
<head>
<meta charset="UTF-8"/>
<title>Neighborhoods in Washington, D.C. - Wikipedia</title>
<script>document.documentElement.className=document.documentElement.className.replace(/(^|\s)client-nojs(\s|$)/,"$1client-js
$2");RLCONF={"wgCanonicalNamespace":"","wgCanonicalSpecialPageName":!1,"wgNamespaceNumber":0,"wgPageName":"Neighborhoods_in_W
ashington,_D.C.","wgTitle":"Neighborhoods in Washington, D.C.","wgCurRevisionId":899568075,"wgRe
```

```
webpage = html.fromstring(page.content)
lst = webpage.xpath('//li/a/@href')
print(lst[0:15])
#lst
```

```
['#List_of_neighborhoods_by_ward', '#Ward_1', '#Ward_2', '#Ward_3', '#Ward_4', '#Ward_5', '#Ward_6', '#Ward_7', '#Ward_8', '#
References', '#External_links', '/wiki/Adams_Morgan', '/wiki/Columbia_Heights_(Washington,_D.C.)', '/wiki/Howard_University',
'/wiki/Kalorama,_Washington,_D.C.']
```

```
lst=lst[11:lst.index('/wiki/Woodland,_Washington,_D.C.')+1]
lst[:10]
```

Fig. 1. Using Wikipedia for neighborhoods of Washington DC

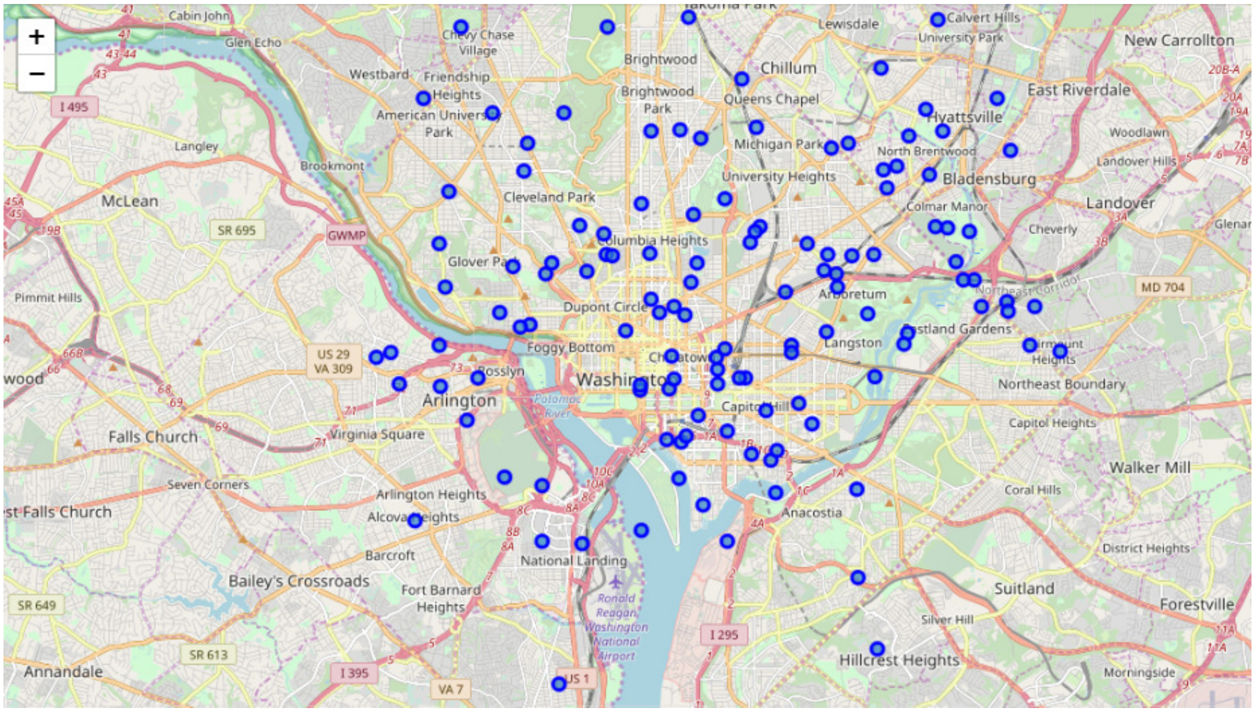

Fig. 2. Map of adding a marker (neighborhoods) on the map of Washington DC

2. To find the **Residential Building (Apartment / Condo)** with good rating we have utilized FourSquare API, namely the `explore the venues` request with a corresponding category-Id of ` 4d954b06a243a5684965b473.

```
# create a function to repeat the same process to all the neighborhoods in Toronto
def getNearbyhomes(names, latt, long, radius=500, LIMIT=100):
    from pandas.io.json import json_normalize
    venues_list=[]
    categoryId = '4d954b06a243a5684965b473'
    for name, lat, lng in zip(names, latt, long):
        print(name)

        # create the API request URL
        url1 = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={},{}&radius={}&limit={}&categoryId={}'.format(
            CLIENT_ID,
            CLIENT_SECRET,
            VERSION,
            lat,
            lng,
            radius,
            LIMIT,
            categoryId)

        # make the GET request
        results = requests.get(url1).json()["response"]['groups'][0]['items']
        venues_list.append([(
            name,
            lat,
            lng,
            v['venue']['name'],
            v['venue']['location']['lat'],
            v['venue']['location']['lng'],
            v['venue']['id']) for v in results])
    nearby_venues = pd.DataFrame([item for venue_list in venues_list for item in venue_list])
    nearby_venues.columns = ['Neighborhood',
                  'Neighborhood Latitude',
                  'Neighborhood Longitude',
                  'Venue',
                  'Venue Latitude',
                  'Venue Longitude',
                  'Venue Category']

    return(nearby_venues)
```

```
Washington_homes = getNearbyhomes(names=districts_cords["Neighborhood"],
                                  latt=districts_cords["Latitude"],
                                  long=districts_cords["Longitude"]
                                 )
```

Fig. 3. Screenshot of utilizing the FourSquare API for finding homes

3. To find the **Indian Restaurant** with more number of likes (preferred by others), we have used FourSquare API, namely the `explore the venues` request with a corresponding category-Id of ` 4bf58dd8d48988d10f941735.

```
# create a function to repeat the same process to all the neighborhoods in Toronto
def getNearbyres(names, latt, long, radius=1800, LIMIT=200):
    from pandas.io.json import json_normalize
    venues_list=[]
    categoryId2 = '4bf58dd8d48988d10f941735'
    for name, lat, lng in zip(names, latt, long):
        print(name)

        # create the API request URL
        url2='https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={},{}&radius={}&limit={}&categoryId={}'.format(
             CLIENT_ID,
             CLIENT_SECRET,
             VERSION,
             lat,
             lng,
             radius,
             LIMIT,
             categoryId2)

        # make the GET request
        results = requests.get(url2).json()["response"]['groups'][0]['items']
        venues_list.append([(
            name,
            lat,
            lng,
            v['venue']['name'],
            v['venue']['location']['lat'],
            v['venue']['location']['lng'],
            v['venue']['id']) for v in results])
    nearby_venues = pd.DataFrame([item for venue_list in venues_list for item in venue_list])
    nearby_venues.columns = ['Neighborhood',
                  'Neighborhood Latitude',
                  'Neighborhood Longitude',
                  'Venue',
                  'Venue Latitude',
                  'Venue Longitude',
                  'Venue Category']

    return(nearby_venues)
```

```
Washington_res = getNearbyres(names=districts_cords["Neighborhood"],
                                   latt=districts_cords["Latitude"],
                                   long=districts_cords["Longitude"]
                                  )
```

Fig. 4. Screenshot of utilizing the FourSquare API for finding nearby restaurants

Methodology

After the data is collected we need to understand data using approaches such as univariate statistics, and histogram. We have evaluated the mean and standard deviation of the collected data to find that whether data collected is useful or there is redundancy of data for example: same neighborhoods could have multiple option of homes and restaurants. We have used “groupby” command to get rid of redundancy of data. So, after we understand the data we need to prepare the data using the models such as feature engineering. We have used google map for data visualization of neighborhoods of Washington dc.

```
# Let's visualize our total likes based on a histogram

import matplotlib.pyplot as plt
Washington_residential_group['Resturent_Likes'].hist(bins=4)
plt.show()
```

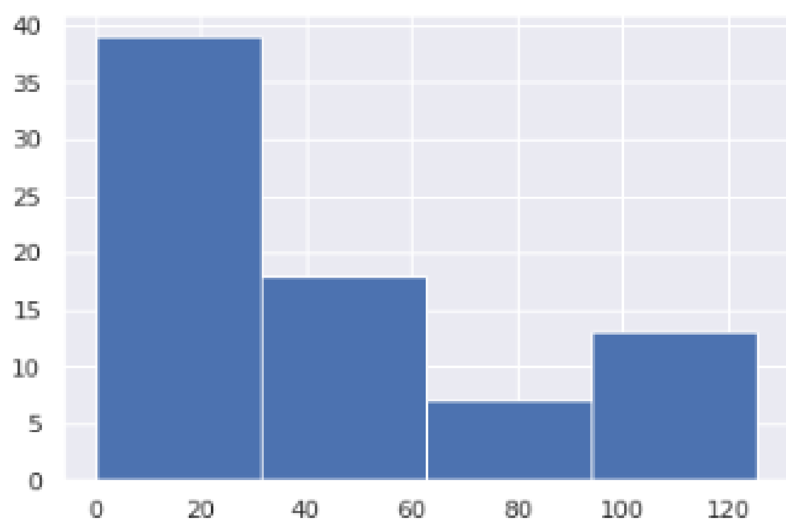

Fig. 5. Data visualization using histogram for no. of likes of restaurants in the neighborhoods of Washington DC

```
# let's visualize our total likes of resturents based on a histogram

import matplotlib.pyplot as plt
Washington_homes_group['Popularirt_of_home'].hist(bins=4)
plt.show()
```

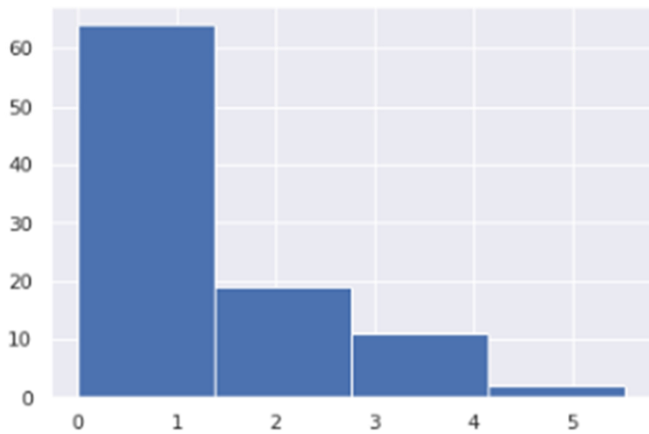

Fig. 6. Data visualization using histogram for popularity of homes in the neighborhoods of Washington DC

We have utilized the API of foursquare.com to obtain the number of likes of restaurant and homes. After data is prepared we can model the data using machine learning models. We have chosen the K-means clustering approach. We have divided the data set into four clusters using k-means clustering model. We have also created a cluster map to show the four cluster of neighborhoods.

Result

I have used the K-means clustering algorithm, to solve the problem of favorable accommodation of software professionals. I could generate four clusters of neighborhoods of Washington DC. These are as follows:

Cluster 1:

- Somewhat Popular homes
- The mostly poor or average restaurant

```
merge.loc[merge['label']==0]
```

	Neighborhood	Neighborhood Latitude_x	Neighborhood Longitude_x	Popularirt_of_home	Neighborhood Latitude_y	Neighborhood Longitude_y	Resturent_Likes	total likes_resturent	Popular_home	label
0	Adams Morgan	38.922610	-77.042661	0.935484	38.922610	-77.042661	61.210526	avg avg	Somewhat Popular	0
1	Berkley	38.912056	-77.088250	1.000000	38.912056	-77.088250	8.000000	poor	Somewhat Popular	0
5	Brightwood Park	38.957100	-77.024900	1.000000	38.957100	-77.024900	1.000000	poor	Somewhat Popular	0
9	Colony Hill	38.912800	-77.087200	2.000000	38.912800	-77.087200	8.000000	poor	Somewhat Popular	0
12	Dupont Circle	38.909620	-77.043410	1.360000	38.909620	-77.043410	50.580645	avg avg	Somewhat Popular	0
19	Glover Park	38.922500	-77.074722	1.000000	38.922500	-77.074722	4.500000	poor	Somewhat Popular	0
23	Kalorama	38.918400	-77.048000	1.157895	38.918400	-77.048000	58.850000	avg avg	Somewhat Popular	0
25	Langdon	38.922800	-76.973900	1.000000	38.922800	-76.973900	7.000000	poor	Somewhat Popular	0
27	Logan Circle	38.909720	-77.030280	1.125000	38.909720	-77.030280	81.194444	great	Somewhat Popular	0
30	Mount Pleasant	38.928694	-77.037333	1.176471	38.928694	-77.037333	32.733333	avg avg	Somewhat Popular	0
31	Mount Vernon Square	38.902528	-77.023583	1.800000	38.902528	-77.023583	99.965517	great	Somewhat Popular	0
37	Observatory Circle	38.921000	-77.067000	1.000000	38.921000	-77.067000	7.818182	poor	Somewhat Popular	0
38	Park View	38.934800	-77.021200	1.500000	38.934800	-77.021200	6.333333	poor	Somewhat Popular	0
39	Penn Quarter	38.897200	-77.024000	1.700000	38.897200	-77.024000	95.615385	great	Somewhat Popular	0
44	Shepherd Park	38.984400	-77.033000	2.000000	38.984400	-77.033000	11.750000	below avg	Somewhat Popular	0
45	Southwest Federal Center	38.885861	-77.015194	1.000000	38.885861	-77.015194	86.857143	great	Somewhat Popular	0
47	Stronghold	38.924900	-77.008400	1.750000	38.924900	-77.008400	73.000000	great	Somewhat Popular	0
55	Wakefield	38.949692	-77.071558	1.750000	38.949692	-77.071558	12.200000	below avg	Somewhat Popular	0

Fig. 7. Screenshot of Cluster 1.

Cluster 2:

- . Most Popular Home
- . Great and above avg. restaurants

```
merge.loc[merge['label']==1]
```

	Neighborhood	Neighborhood Latitude_x	Neighborhood Longitude_x	Popularirt_of_home	Neighborhood Latitude_y	Neighborhood Longitude_y	Resturent_Likes	total likes_resturent	Popular_home	label
3	Brentwood	38.918700	-76.990200	4.800000	38.918700	-76.990200	9.000000	below avg	Most Popular	1
7	Carver Langston	38.900900	-76.977300	3.600000	38.900900	-76.977300	59.000000	avg avg	Most Popular	1
8	Chinatown	38.899800	-77.021700	2.190476	38.899800	-77.021700	96.714286	great	Most Popular	1
10	Columbia Heights	38.925000	-77.030000	2.444444	38.925000	-77.030000	67.437500	avg avg	Most Popular	1
11	Downtown	38.902500	-77.032861	2.857143	38.902500	-77.032861	98.823529	great	Most Popular	1
16	Foggy Bottom	38.900889	-77.050056	2.791667	38.900889	-77.050056	55.038462	avg avg	Most Popular	1
22	Judiciary Square	38.895280	-77.018472	2.285714	38.895280	-77.018472	85.869565	great	Most Popular	1
29	McLean Gardens	38.937200	-77.075000	3.000000	38.937200	-77.075000	20.625000	below avg	Most Popular	1
32	Mount Vernon Triangle	38.902500	-77.017780	2.500000	38.902500	-77.017780	112.629630	great	Most Popular	1
33	Navy Yard	38.877003	-77.001628	3.133333	38.877003	-77.001628	9.142857	below avg	Most Popular	1
34	Near Northeast	38.901300	-77.003200	3.750000	38.901300	-77.003200	125.428571	great	Most Popular	1
35	NoMa	38.906500	-77.004917	5.538462	38.906500	-77.004917	32.666667	avg avg	Most Popular	1
41	Pleasant Hill	38.919800	-77.020000	2.444444	38.919800	-77.020000	121.500000	great	Most Popular	1
43	Shaw	38.911139	-77.021917	2.125000	38.911139	-77.021917	95.904762	great	Most Popular	1
46	Southwest Waterfront	38.881200	-77.016400	3.000000	38.881200	-77.016400	118.800000	great	Most Popular	1
48	Sursum Corda	38.905300	-77.011200	3.555556	38.905300	-77.011200	115.500000	great	Most Popular	1
49	Swampoodle	38.900194	-77.008194	3.625000	38.900194	-77.008194	106.052632	great	Most Popular	1
53	Truxton Circle	38.911056	-77.008972	2.750000	38.911056	-77.008972	57.800000	avg avg	Most Popular	1
54	U Street	38.917046	-77.032930	2.318182	38.917046	-77.032930	48.625000	avg avg	Most Popular	1
56	West End	38.907056	-77.049694	2.608696	38.907056	-77.049694	50.225806	avg avg	Most Popular	1
59	Woodley Park	38.928444	-77.055972	2.200000	38.928444	-77.055972	16.400000	below avg	Most Popular	1

Fig. 8. Screenshot of Cluster 2

.

Cluster 3:

- . Least Popular Home
- . Poor and below avg. restaurants

```
merge.loc[merge['label']==2]
```

	Neighborhood	Neighborhood Latitude_x	Neighborhood Longitude_x	Popularirt_of_home	Neighborhood Latitude_y	Neighborhood Longitude_y	Resturent_Likes	total likes_resturent	Popular_home	label
4	Brightwood	38.961200	-77.027500	0.750000	38.961200	-77.027500	1.000000	poor	least popular	2
6	Burleith	38.915300	-77.072800	0.333333	38.915300	-77.072800	7.250000	poor	least popular	2
13	Eckington	38.915300	-77.001700	0.666667	38.915300	-77.001700	45.500000	avg avg	least popular	2
14	Edgewood	38.922600	-77.000500	0.500000	38.922600	-77.000500	6.750000	poor	least popular	2
15	Fairlawn	38.870830	-76.978890	0.500000	38.870830	-76.978890	0.000000	poor	least popular	2
17	Friendship Heights	38.957000	-77.083778	0.800000	38.957000	-77.083778	17.000000	below avg	least popular	2
24	Kingman Park	38.895400	-76.977200	0.500000	38.895400	-76.977200	59.000000	avg avg	least popular	2
26	LeDroit Park	38.919052	-77.017035	0.500000	38.919052	-77.017035	120.500000	great	least popular	2
28	Massachusetts Heights	38.927200	-77.069200	0.800000	38.927200	-77.069200	19.571429	below avg	least popular	2
40	Petworth	38.942161	-77.025525	0.666667	38.942161	-77.025525	4.000000	poor	least popular	2
50	Takoma	38.975000	-77.020280	0.500000	38.975000	-77.020280	0.500000	poor	least popular	2

Fig. 9. Screenshot of Cluster 3

Cluster 4:

- . unpopular Home
- . Average restaurants

```
merge.loc[merge['label']==3]
```

	Neighborhood	Neighborhood Latitude_x	Neighborhood Longitude_x	Popularirt_of_home	Neighborhood Latitude_y	Neighborhood Longitude_y	Resturent_Likes	total likes_resturent	Popular_home	label
2	Bloomingdale	38.91640	-77.01140	0.083333	38.91640	-77.01140	77.000000	great	Unpopular	3
18	Georgetown	38.90944	-77.06500	0.000000	38.90944	-77.06500	40.470588	avg avg	Unpopular	3
20	Howard University	38.92222	-77.01944	0.000000	38.92222	-77.01944	61.571429	avg avg	Unpopular	3
21	Ivy City	38.90990	-76.99170	0.000000	38.90990	-76.99170	48.200000	avg avg	Unpopular	3
36	North Michigan Park	38.94540	-76.99600	0.000000	38.94540	-76.99600	7.000000	poor	Unpopular	3
42	Pleasant Plains	38.92880	-77.02330	0.142857	38.92880	-77.02330	53.285714	avg avg	Unpopular	3
51	Tenleytown	38.94600	-77.07900	0.000000	38.94600	-77.07900	10.142857	below avg	Unpopular	3
52	Trinidad	38.90570	-76.98440	0.000000	38.90570	-76.98440	48.200000	avg avg	Unpopular	3
57	Woodland	38.92060	-77.06080	0.000000	38.92060	-77.06080	14.857143	below avg	Unpopular	3
58	Woodland Normanstone	38.92070	-77.06080	0.000000	38.92070	-77.06080	14.857143	below avg	Unpopular	3

Fig. 10. Screenshot of Cluster 4

Discussion

For the problem of exploring favorable residence for a group of software professionals who are coming to Washington Dc for 3 months, I have utilized API of foursquare.com to obtain location data. I have also used Wikipedia of neighborhoods of Washington DC to find the neighborhoods of Washington DC. I have used likes as a proxy for the quality of the restaurant. The more likes imply, the better the restaurant is. I have used "likes" for the popularity of homes. Initially, I have tried to get the rating for good home using API, but from the free subscription of foursquare.com, I was only able to get a rating of few homes. Then I have switched to the "likes" criteria for evaluating the popularity of homes.

I have used the K-means approach to divide the data set into four clusters. I have got four clusters with different categories (options) of homes and restaurants. I have also used cluster-maps for data visualization. So, the clients will have a clear image of all four options of clusters and the tradeoff is evidently visible.

Conclusion

Although it is hard to find a balance between good housing and a fine restaurant. But we have provided our consumers/end users four clusters of neighborhoods of Washington DC to meet their desired criteria. This will make it easier to understand the trade-off for each offer.