Coursera Capstone

IBM Applied Data Science Capstone

“The Battle of Neighbourhoods”

Finding a Neighbourhood for an Indian family to settle down in Queens, New York

By: Abhinav Yadav
December 2020

1. Introduction:

The United States of America is the country with one of the largest immigrant population in the world. After the IT boom , this immigration in recent times has been driven by IT workers from Asia coming into the states. A large number of them belong to India which due to its high population and educational system is able to provide one of the cheapest most talented manpower for IT firms. Indians flock by the 1000's to America chasing the American Dream. This along with the America's policy of helping the best brains of the world settle into this country has led to a large number of highly skilled immigrant population. New York has one of the largest Indian population . Out of all the Boroughs - Queens with 6.2 % Percent of Indian Americans in Municipality Population seems to be one that has welcomed the India community with open arms and is a popular location for Indians to settle down. Indians culturally have a strong sense of family and generally prefer to be around theirs . This means they generally come to America with their spouses and Children . While to settle down the future life of their Children influences a decision to a large extent . People prefer living in neighbourhoods that have schools and parks . The presence of Doctor's Offices is also a big influencer.

In this project we will try to find an optimal location for an India family to settle. Specifically, this report will be targeted to people interested in settling down in Queens , New York, USA.

Since there are lots of Neighbourhoods in Queens we will try and find a location that has :
1- Schools
2- Parks
3- Doctor's Office

We will also try and include neighbourhoods with Indian Restaurants (Although this will be an additional clincher not an influencer)

We will use our data science powers to generate a few most promising neighbourhoods based on this criteria. Advantages of each area will then be clearly expressed so that best possible final location can be chosen by stakeholders.

2. Problem Statement:

The objective of this capstone project will be to find a suitable neighbourhood for an Indian family with kids to settle down .This would be done by leveraging Data Science and Machine Learning (k - means). The main Business question that will be answered in the Capstone Project will be : “Which neighbourhoods in Queens New York are best suitable for an Indian Family with kids to settle down in ?”

3. Stakeholders/Target Audience

The Capstone Project will be particularly useful for people looking to move into new towns, cities or even countries. The project can also be modified to go beyond the current scope as it can be used to scope out other businesses in the area be it gyms , schools etc . The project can be used in an advisory capability by property consultants, realtors etc who can use the project to give their customer an overview of the area and allow them to make a better informed decision. With basic knowledge, customers themselves can use this project to better understand the option available with them to make informed decisions.

4. Foursquare API

The project largely relies on utilising the Foursquare API, mainly the Places API to gather data related to locations. Foursquare is a location technology platform that allows the user to access its upto date database through an API to provide details of location/ venues the user might be interested in. The details include Name, Category, Location (latitude, longitude) , Ratings , reviews , menu etc as per the customer needs. We will be leveraging the API in this project to find out the venues that are present in the Queens Area to look for a suitable neighbourhood for settling down.

5. Data

Data used in this project is the New York dataset and was sourced from: https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DS0701EN-SkillsNetwork/labs/newyork_data.json

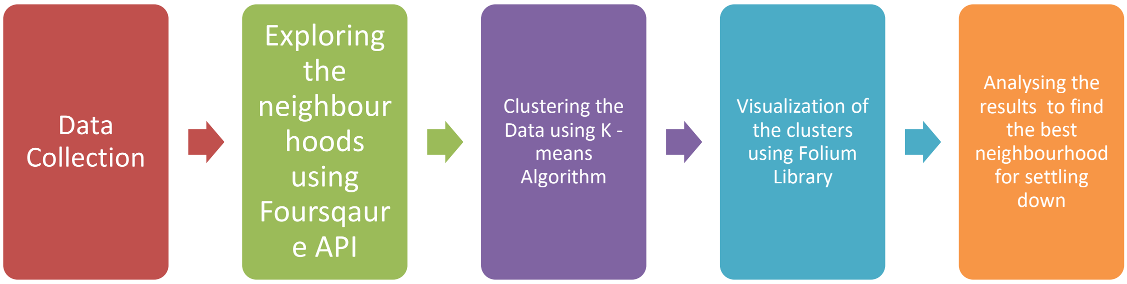
The coordinates of places if and when required can be sought by using geopy. Geopy is a Python client for several popular geocoding web services. Geopy makes it easy for Python developers to locate the coordinates of addresses, cities, countries, and landmarks across the globe using third-party geocoders and other data sources

Finally Foursquare API will be used for identifying and analysing areas of interests which basically involves using the API to gather the following details –

a. Number of venues in a particular area based on the radius provided by the user based on neighbourhood details.
b. Name of the venue
c. Category of venues (Schools , Parks etc)

Location of the Venue.

6. Methodology



Data Collection

Data used in this project is the New York dataset and was sourced from: https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DS0701EN-SkillsNetwork/labs/newyork_data.json

```
In [2]: !wget -q -O 'newyork_data.json' https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkill
        print('Data downloaded!')

Data downloaded!

In [3]: with open('newyork_data.json') as json_data:
            newyork_data = json.load(json_data)

In [4]: neighborhoods_data = newyork_data['features']
```

The data set obtained is converted into a pandas dataframe to allow better access and manipulation of data.

```
In [5]: # define the dataframe columns
        column_names = ['Borough', 'Neighborhood', 'Latitude', 'Longitude']

        # instantiate the dataframe
        neighborhoods = pd.DataFrame(columns=column_names)

        for data in neighborhoods_data:
            borough = neighborhood_name = data['properties']['borough']
            neighborhood_name = data['properties']['name']

            neighborhood_latlon = data['geometry']['coordinates']
            neighborhood_lat = neighborhood_latlon[1]
            neighborhood_lon = neighborhood_latlon[0]

            neighborhoods = neighborhoods.append({'Borough': borough,
                                                  'Neighborhood': neighborhood_name,
                                                  'Latitude': neighborhood_lat,
                                                  'Longitude': neighborhood_lon}, ignore_index=True)
```

In [6]:
```
neighborhoods.head()
```

Out[6]:

	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

In [7]:
```
print('The dataframe has {} boroughs and {} neighborhoods.'.format(
        len(neighborhoods['Borough'].unique()),
        neighborhoods.shape[0]
    )
)
```

```
The dataframe has 5 boroughs and 306 neighborhoods.
```

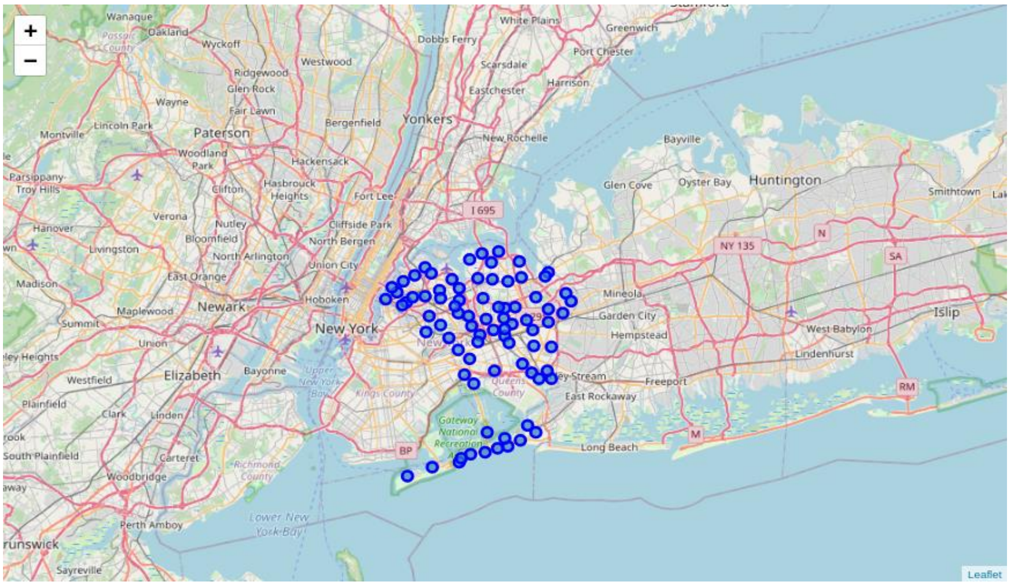
Exploring the Queens Borough

Since exploring the whole of New York will be unfeasible we limit our search for a place for a Neighbourhood to the Queens Area . We can visualize the Queens area using Folium Library

In [14]:
```
Queens_data = neighborhoods[neighborhoods['Borough'] == 'Queens'].reset_index(drop=True)
Queens_data.head()

address = 'Queens, NY'

geolocator = Nominatim(user_agent="ny_explorer")
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude
print('The geograpical coordinate of Queens, NY are {}, {}.'.format(latitude, longitude))
```

```
The geograpical coordinate of Queens, NY are 40.7498243, -73.7976337.
```

In [15]:
```
# create map of Manhattan using latitude and longitude values
map_Queens = folium.Map(location=[latitude, longitude], zoom_start=11)

# add markers to map
for lat, lng, label in zip(Queens_data['Latitude'], Queens_data['Longitude'], Queens_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_Queens)

map_Queens
```



Exploring the Neighbourhood using Foursquare API

We then Initialize the Foursquare API to allow us to use Foursquare services to access venues etc according to the neighbourhood.

```
In [10]: LIMIT = 500
         radius = 5000
         CLIENT_ID = 'FC2KSLQ5XDRHHBQV3WM0GZARVVJ21CK3J2IOZE3QKP50GC03'
         CLIENT_SECRET = '0S5B52OWK0ZWHQROXCDKLW0POY11L4JTESJICNOVUXTXDKQW'
         VERSION = '20201209'
```

Collecting the date regarding the venues in the area of within 1000 meters in the Queens Borough using the Foursquare API.

```
In [12]:
neighborhoods = neighborhoods[neighborhoods['Borough'] == 'Queens'].reset_index(drop=True)
Queens_venues = getNearbyVenues(names=neighborhoods['Neighborhood'], latitudes=neighborhoods['Latitude'], longitudes
Queens_venues.head()
```

Out[12]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Astoria	40.768509	-73.915654	A-One Laundry	40.766701	-73.916924	Laundry Service
1	Astoria	40.768509	-73.915654	Cypriana Liquor Store	40.767462	-73.914957	Liquor Store
2	Astoria	40.768509	-73.915654	Leli's Bakery and Pastry Shop	40.765132	-73.917696	Bakery
3	Astoria	40.768509	-73.915654	Burger King	40.769464	-73.916434	Fast Food Restaurant
4	Astoria	40.768509	-73.915654	NYPD - 114th Precinct	40.769508	-73.915360	Police Station

```
In [13]: Queens_venues.shape

Out[13]: (7954, 7)
```

Exploring the most common venues according to the neighbourhood

```
In [16]: Queens_common_venues = Queens_venues.groupby('Venue Category').count().sort_values('Neighborhood',ascending=False).r
         Queens_common_venues.head()
```

Out[16]:

	Venue Category	Count
0	Salon / Barbershop	325
1	Residential Building (Apartment / Condo)	272
2	Doctor's Office	245
3	Deli / Bodega	243
4	Building	239

We then start to look for our priorities in Queens

i) - Looking for schools in Queens

In [17]:
```
Queens_schools = Queens_venues[Queens_venues['Venue Category']=='School'].reset_index(drop=True)
Queens_schools
```

Out[17]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Jackson Heights	40.751981	-73.882821	Saint Joan Of Arc School	40.752067	-73.884623	School
1	Jackson Heights	40.751981	-73.882821	Ps 212Q	40.753397	-73.884463	School
2	Jackson Heights	40.751981	-73.882821	Children's Workshop	40.753057	-73.882103	School
3	Elmhurst	40.744049	-73.881656	P.S.7	40.743808	-73.882510	School
4	Corona	40.742382	-73.856825	P.S. 14 Q Fairview school	40.741677	-73.853932	School
5	Forest Hills	40.725264	-73.844475	PS 303Q	40.725825	-73.843885	School
6	Richmond Hill	40.697947	-73.831833	Ps 90	40.696197	-73.830170	School
7	Flushing	40.764454	-73.831773	E-Math	40.763031	-73.831048	School
8	Sunnyside	40.740176	-73.926916	P.S. 199Q (Maurice A. Fitzgerald Elementary Sc...	40.740169	-73.925599	School
9	Sunnyside	40.740176	-73.926916	P.S. 199	40.740951	-73.925761	School

In [18]:
```
Queens_schools['Neighborhood'].value_counts().head(10)
```

Out[18]:
```
Neponsit             4
Belle Harbor         4
Kew Gardens Hills    3
Fresh Meadows        3
Bayside              3
Little Neck          3
Jackson Heights      3
Bellaire             3
Douglaston           3
Edgemere             3
Name: Neighborhood, dtype: int64
```

ii) – Looking for Doctors Offices in Queens

In [19]:
```
Queens_doc = Queens_venues[Queens_venues['Venue Category']=="Doctor's Office"].reset_index(drop=True)
Queens_doc
```

Out[19]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Astoria	40.768509	-73.915654	Chirag V. Vasa, M.D.	40.767670	-73.917000	Doctor's Office
1	Woodside	40.746349	-73.901842	Pediatric Eye MD	40.747246	-73.902658	Doctor's Office
2	Jackson Heights	40.751981	-73.882821	Dr. Eduard Shnaydman, M.D.	40.751873	-73.884514	Doctor's Office
3	Jackson Heights	40.751981	-73.882821	84th Street Medical PC	40.750180	-73.882350	Doctor's Office
4	Jackson Heights	40.751981	-73.882821	Lens Lab Express	40.749629	-73.884094	Doctor's Office
5	Jackson Heights	40.751981	-73.882821	Sleep Diagnostics Of New York Inc.	40.750172	-73.884659	Doctor's Office
6	Jackson Heights	40.751981	-73.882821	Dr Elena King	40.750290	-73.884468	Doctor's Office
7	Jackson Heights	40.751981	-73.882821	37 Avenue Medical, P.C.	40.750038	-73.883690	Doctor's Office
8	Jackson Heights	40.751981	-73.882821	Dr. Chhabra's Office	40.752960	-73.883636	Doctor's Office
9	Jackson Heights	40.751981	-73.882821	Mt Sinai	40.750000	-73.884048	Doctor's Office

In [20]:
```
Queens_doc["Neighborhood"].value_counts().head(10)
```

Out[20]:
```
Rego Park          14
Lindenwood         13
Glendale            9
Holliswood          9
Bay Terrace         9
Pomonok             8
Bayside             8
Jackson Heights     8
Cambria Heights     8
Richmond Hill       7
Name: Neighborhood, dtype: int64
```

iii) - Looking for Parks in Queens

In [21]:
```
Queens_park = Queens_venues[Queens_venues['Venue Category']=="Park"].reset_index(drop=True)
Queens_park
```

Out[21]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Elmhurst	40.744049	-73.881656	Broadway Park (Elmhurst)	40.742425	-73.882481	Park
1	Corona	40.742382	-73.856825	William F. Moore Park ('Spaghetti Park')	40.743666	-73.855443	Park
2	East Elmhurst	40.764073	-73.867041	1%	40.766079	-73.866219	Park
3	Maspeth	40.725427	-73.896217	Peter Chahales Park	40.724714	-73.894511	Park
4	Glendale	40.702762	-73.870742	Forest Park - Dry Harbor Playground	40.702988	-73.867396	Park
5	Glendale	40.702762	-73.870742	Dry Harbour Park	40.702910	-73.868190	Park
6	Woodhaven	40.689887	-73.858110	51B 10-89	40.691453	-73.857525	Park
7	South Ozone Park	40.668550	-73.809865	Pals Oval Park	40.668634	-73.805878	Park
8	South Ozone Park	40.668550	-73.809865	Back Street Park	40.666542	-73.806407	Park
9	South Ozone Park	40.668550	-73.809865	Back Streets Park (Officer Edward Byrn Park)	40.667846	-73.806453	Park

In [22]:
```
Queens_park["Neighborhood"].value_counts().head(10)
```

Out[22]:
```
Malba                  4
Forest Hills Gardens   4
Rochdale               4
Brookville             3
Neponsit               3
Broad Channel          3
South Ozone Park       3
Edgemere               2
Hollis                 2
Glendale               2
Name: Neighborhood, dtype: int64
```

iv) - Looking for Indian Restaurants in Queens

In [42]:
```
Queens_food = Queens_venues[Queens_venues['Venue Category']=="Indian Restaurant"].reset_index(drop=True)
Queens_food.head(10)
```

Out[42]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Woodside	40.746349	-73.901842	Sagarmatha Restaurant & Bar	40.745380	-73.902000	Indian Restaurant
1	Kew Gardens	40.705179	-73.829819	Tikka Indian Grill	40.705874	-73.830942	Indian Restaurant
2	Richmond Hill	40.697947	-73.831833	Tikka Curry Express	40.699478	-73.830406	Indian Restaurant
3	Long Island City	40.750217	-73.939202	Raj's Indian Kitchen	40.749976	-73.939261	Indian Restaurant
4	Ridgewood	40.708323	-73.901435	Fresh Pond Spice	40.710140	-73.899582	Indian Restaurant
5	Rego Park	40.728974	-73.857827	Sajni 026	40.728123	-73.858071	Indian Restaurant
6	Rego Park	40.728974	-73.857827	Sanji	40.728151	-73.858036	Indian Restaurant
7	Woodhaven	40.689887	-73.858110	Hyderabadi Kitchen	40.692776	-73.858742	Indian Restaurant
8	Bayside	40.766041	-73.774274	Agra Indian Cuisine	40.765396	-73.771535	Indian Restaurant
9	Bayside	40.766041	-73.774274	Ayna Agra Indian Restaurant	40.765478	-73.771737	Indian Restaurant

In [24]:
```
Queens_food["Neighborhood"].value_counts().head(10)
```

Out[24]:
```
Floral Park          11
Jamaica Hills         5
Rego Park             2
Bayside               2
Woodside              1
Hollis                1
Briarwood             1
Richmond Hill         1
North Corona          1
Sunnyside Gardens     1
Name: Neighborhood, dtype: int64
```

Since there are 461 categories we proceed with one hot encoding for getting dummies of the venue category. We the calculate the mean of all venue groups by their neighbourhoods.

```
In [25]: ny_onehot = pd.get_dummies(Queens_venues[['Venue Category']], prefix="", prefix_sep="")   # Using dummies to Encode
         ny_onehot['Neighborhood'] = Queens_venues['Neighborhood']

         fixed_columns = [ny_onehot.columns[182]] + list(ny_onehot.columns[:182]) + list(ny_onehot.columns[183:])  # Getting
         ny_onehot = ny_onehot[fixed_columns]

         print(ny_onehot.shape)
         ny_onehot.head()
```

```
In [26]: Queens_grouped = ny_onehot.groupby('Neighborhood').mean().reset_index()  # Grouping and taking the mean
         print(Queens_grouped.shape)
         Queens_grouped.head()

         (81, 461)
```

Out[26]:

	Neighborhood	Frozen Yogurt Shop	ATM	Accessories Store	Acupuncturist	Advertising Agency	Afghan Restaurant	African Restaurant	Airport	Airport Gate	Airport Service	Airport Terminal	Alternative Healer	American Restaurant
0	Arverne	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.000000
1	Astoria	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.000000
2	Astoria Heights	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.011905	0.0	0.0	0.0	0.0	0.000000
3	Auburndale	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.000000
4	Bay Terrace	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.010309

```
In [43]: grouped_scores = Queens_grouped[['Neighborhood','School',"Doctor's Office",'Park','Indian Restaurant']]  # Only Indi

         grouped_scores.head(5)
```

Out[43]:

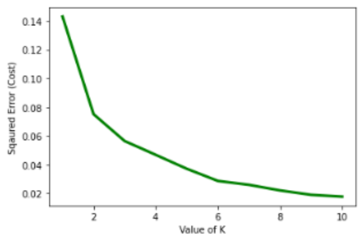
	Neighborhood	School	Doctor's Office	Park	Indian Restaurant
0	Arverne	0.019048	0.009524	0.009524	0.0
1	Astoria	0.000000	0.012195	0.000000	0.0
2	Astoria Heights	0.000000	0.000000	0.000000	0.0
3	Auburndale	0.015625	0.015625	0.000000	0.0
4	Bay Terrace	0.010309	0.092784	0.010309	0.0

CLUSTERING USING K MEANS ALGORITHMS

The neighbourhoods are then clustered using k – means algorithm which is an unsupervised algorithm that will be used to cluster the neighbourhoods into clusters based on the respective venues in the neighbourhoods and their categories . But first we use the elbow method to find the value of k .

```
In [28]: cost =[]
         for i in range(1, 11):
             KM = KMeans(n_clusters = i, max_iter = 500) # Range of k-values
             KM.fit(grouped_scores.drop(columns=['Neighborhood']))
             cost.append(KM.inertia_)      # Getting the cost
```

```
In [29]: # plot the cost against K values
         plt.plot(range(1, 11), cost, color ='g', linewidth ='3')
         plt.xlabel("Value of K")
         plt.ylabel("Sqaured Error (Cost)")
         plt.show()
```




```
In [30]: kclusters = 4      # No.of Clusters
         ny_grouped_clustering = grouped_scores.drop('Neighborhood', 1)
         # run k-means clustering
         kmeans = KMeans(n_clusters=kclusters, random_state=0).fit(ny_grouped_clustering)
         # check cluster labels generated for each row in the dataframe
         kmeans.labels_[0:10]

Out[30]: array([0, 0, 0, 0, 1, 1, 0, 0, 0, 0], dtype=int32)
```

```
In [31]: grouped = grouped_scores.copy(deep=True)
         grouped['Cluster Labels'] = kmeans.labels_     # Adding the labels to the data
         grouped['Cluster Labels'] = grouped['Cluster Labels'].astype(int)  # Float is sometimes returned
         print(grouped.shape)
         grouped.head(10)
```

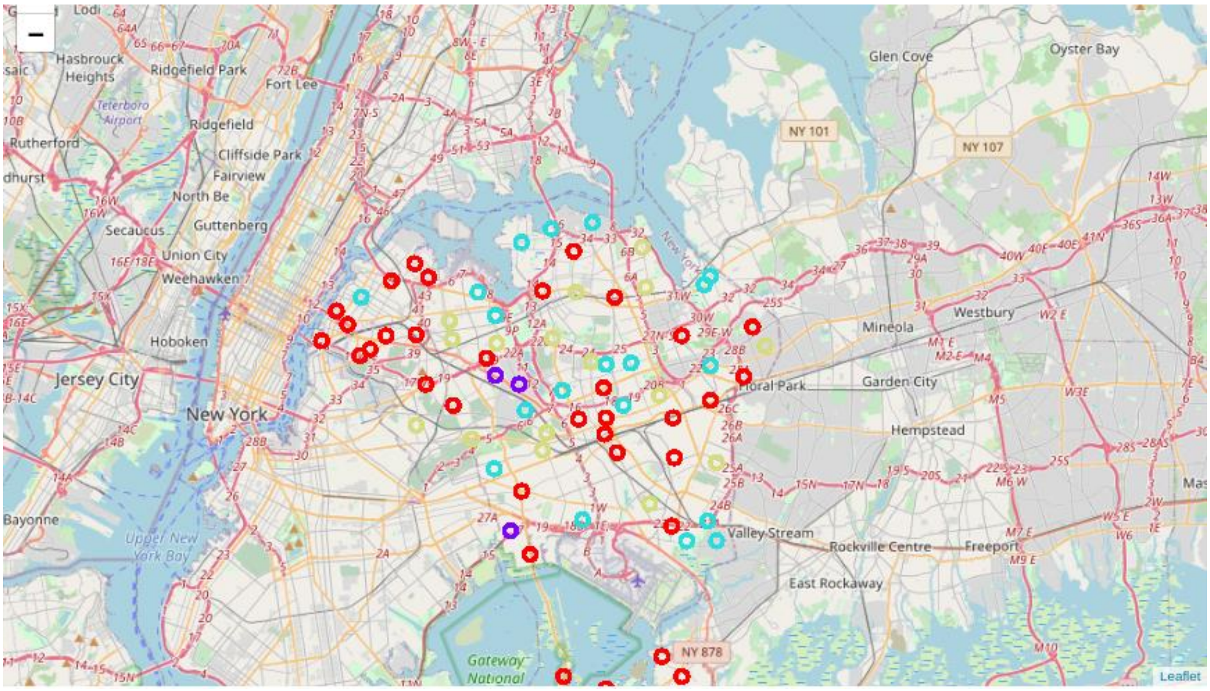
(81, 6)

Out[31]:

	Neighborhood	School	Doctor's Office	Park	Indian Restaurant	Cluster Labels
0	Arverne	0.019048	0.009524	0.009524	0.000000	0
1	Astoria	0.000000	0.012195	0.000000	0.000000	0
2	Astoria Heights	0.000000	0.000000	0.000000	0.000000	0
3	Auburndale	0.015625	0.015625	0.000000	0.000000	0
4	Bay Terrace	0.010309	0.092784	0.010309	0.000000	1
5	Bayside	0.027027	0.072072	0.000000	0.018018	1
6	Bayswater	0.009804	0.009804	0.009804	0.000000	0
7	Beechhurst	0.019417	0.038835	0.009709	0.000000	0
8	Bellaire	0.028846	0.009615	0.009615	0.009615	0
9	Belle Harbor	0.043478	0.032609	0.010870	0.000000	0

Visualization of Clusters using Folium

```
In [33]: map_clusters = folium.Map(location=[latitude,longitude], zoom_start=11)

         # 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(to_merged['Neighborhood Latitude'], to_merged['Neighborhood Longitude'], to_merged
             label = folium.Popup(str(poi) + ' - Cluster ' + str(cluster))
             folium.CircleMarker(
                 [lat, lon],
                 radius=5,
                 popup=label,
                 color=rainbow[cluster-1],
                 fill_color=rainbow[cluster-1],
                 fill_opacity=0.7).add_to(map_clusters)

         map_clusters
```



7. Results

We now take at the results of the clusters which were the output of the k – Means algorithm

```
In [38]: t = to_merged[to_merged['Cluster Labels']==0]
print('Number of Schools in Cluster 0: \t   {}'.format(t[t['Venue Category']=='School'].count()[0]))
print('Number of Doctors Offices in Cluster 0:    {}'.format(t[t['Venue Category']=="Doctor's Office"].count()[0]))
print('Number of Parks in Cluster 0: \t\t   {}'.format(t[t['Venue Category']=='Park'].count()[0]))
print('Number of Indian Restaurants in Cluster 0: {}\n\n'.format(t[t['Venue Category']=='Indian Restaurant'].count()

t = to_merged[to_merged['Cluster Labels']==1]
print('Number of Schools in Cluster 1: \t   {}'.format(t[t['Venue Category']=='School'].count()[0]))
print('Number of Doctors Offices in Cluster 1:    {}'.format(t[t['Venue Category']=="Doctor's Office"].count()[0]))
print('Number of Parks in Cluster 1:\t\t   {}'.format(t[t['Venue Category']=='Park'].count()[0]))
print('Number of Indian Restaurants in Cluster 1: {}\n\n'.format(t[t['Venue Category']=='Indian Restaurant'].count()

t = to_merged[to_merged['Cluster Labels']==2]
print('Number of Schools in Cluster 2: \t   {}'.format(t[t['Venue Category']=='School'].count()[0]))
print('Number of Doctors Offices in Cluster 2:    {}'.format(t[t['Venue Category']=="Doctor's Office"].count()[0]))
print('Number of Parks in Cluster 2: \t\t   {}'.format(t[t['Venue Category']=='Park'].count()[0]))
print('Number of Indian Restaurants in Cluster 2: {}\n\n'.format(t[t['Venue Category']=='Indian Restaurant'].count()

t = to_merged[to_merged['Cluster Labels']==3]
print('Number of Schools in Cluster 3: \t   {}'.format(t[t['Venue Category']=='School'].count()[0]))
print('Number of Doctors Offices in Cluster 3   : {}'.format(t[t['Venue Category']=="Doctor's Office"].count()[0]))
print('Number of Parks in Cluster 3:\t\t   {}'.format(t[t['Venue Category']=='Park'].count()[0]))
print('Number of Indian Restaurants in Cluster 3: {}\n\n'.format(t[t['Venue Category']=='Indian Restaurant'].count()
```

```
Number of Schools in Cluster 0:           17
Number of Doctors Offices in Cluster 0:   45
Number of Parks in Cluster 0:             20
Number of Indian Restaurants in Cluster 0: 11


Number of Schools in Cluster 1:           5
Number of Doctors Offices in Cluster 1:   34
Number of Parks in Cluster 1:             1
Number of Indian Restaurants in Cluster 1: 2


Number of Schools in Cluster 2:           51
Number of Doctors Offices in Cluster 2:   52
Number of Parks in Cluster 2:             33
Number of Indian Restaurants in Cluster 2: 4


Number of Schools in Cluster 3:           16
Number of Doctors Offices in Cluster 3   : 114
Number of Parks in Cluster 3:             14
Number of Indian Restaurants in Cluster 3: 16
```

We can see from the result that there are four clusters of which Cluster 2 has the most balanced mix of our priorities.

We then select cluster 2 to explore further as it is the most ideal.

In [42]:
```
temp = to_merged[to_merged['Cluster Labels']==2]

temp.groupby('Venue Category').count().reset_index().rename(columns={'Indian Restaurant':'Count'})[['Venue Category'
```

Out[42]:

	Venue Category	Count
0	Salon / Barbershop	98
1	Deli / Bodega	74
2	Building	73
3	Doctor's Office	52
4	School	51
5	Residential Building (Apartment / Condo)	50
6	Office	49
7	Bus Line	49
8	Laundry Service	49
9	Chinese Restaurant	46

8. Discussion

We can see from the results above that cluster 2 has the most balanced combination of our priority list and is therefore best suited for shifting of a new family into the cluster

We now further explore the neighbourhoods in detail in the Cluster 2.

In [40]:
```
x = temp[temp['Venue Category'].str.contains("Indian Restaurant|Park|Doctor's Office| School")].groupby(['Neighborho
x.rename(columns = {'School':'Count'}, inplace = True)
x = pd.DataFrame(x['Count'])
x
```

	Martial Arts School	1
Breezy Point	Doctor's Office	1
	Park	1
	Parking	2
Briarwood	Doctor's Office	2
	Elementary School	2
	Indian Restaurant	1
	Middle School	1
	Nursery School	2
	Park	1
	Parking	2
Broad Channel	Elementary School	1
	National Park	1

It is visible that the Neighbourhood of Briarwood is best suited for living for our new family. Briarwood has Elementary, Middle School, Doctors Offices,Parks and Indian restaurant which satisfies all conditions that we started out when looking for a neighbourhood.

```
In [46]: x[17:24]
```

Out[46]:

Neighborhood	Venue Category	Count
Briarwood	Doctor's Office	2
	Elementary School	2
	Indian Restaurant	1
	Middle School	1
	Nursery School	2
	Park	1
	Parking	2

9. Conclusion

This project while looking very simple utilises the power of data and more so machine learning to provide an in depth look into the neighbourhoods in our desired Geographical area . An informed decision can be taken after analysis done through the project. The same has been looked into , in the Discussion section of the project. The icing on the cake as far as the project goes is that the project is easily modifiable as per the whims and fancies of user which enhances its usability under different conditions.