Coursera Capstone

IBM Applied Data Science Capstone

“The Battle of Neighbourhoods”

Finding a location to open an Indian restaurant in Manhattan New York

By: Ankur Mehta
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 has led to a growth of new markets being created that caters to the immigrated folk. One of the most lucrative ones of them being the Indian restaurant business. Indian cooking is known across the world for its diverse flavours and spices . Both Indians and American Nationals alike have taken a great fondness to the Indian Cuisine. This has led to a spurt in the Indian restaurant business. These businesses have specifically targeted places with large Indian population earlier but due to market saturation have started moving away to other parts.

In this project we will try to find an optimal location for an India restaurant. Specifically, this report will be targeted to stakeholders interested in opening an Indian restaurant in Manhattan , New York, USA.

Since there are lots of restaurants in New York we will try to detect locations that are not already crowded with restaurants. We are also particularly interested in areas with no Indian restaurants in vicinity.

We will use our data science powers to generate a few most promising neighbourhoods based on these 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 location to open an Indian Restaurant in Manhattan New York that would have the most chance of being successful 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 Manhattan New York are best suitable for an Indian Restaurant ?”

3. Stakeholders/Target Audience

The Capstone Project will be particularly useful for people looking to open a restaurant in a given area. The project can also be modified to go beyond just the restaurant 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 the this project to better understand the option available with them to make informed business decisions thus improving their chances of being successful.

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 Manhattan Area to look for a suitable place to open our Indian Restaurant.

5. Data

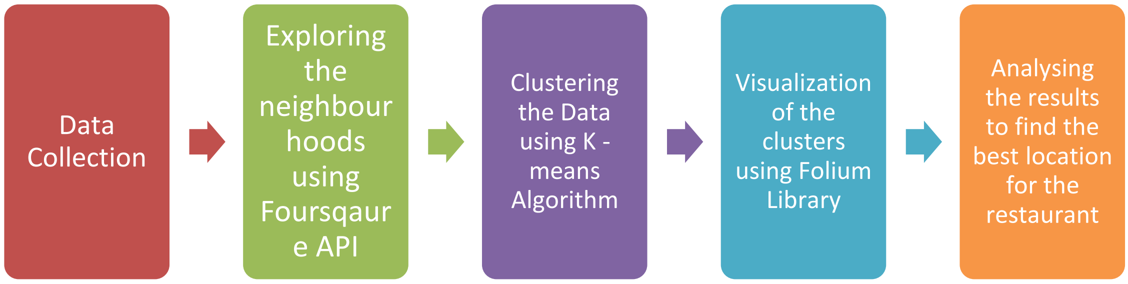
Data used in this project is the New York dataset and was sourced from: https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DS0701EN-SkillsNetwork/labs/newyork_data.json

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 (Restaurants, gyms etc)
d. Location of the Venue (Latitude , Longitude)

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/IBMD
        print('Data downloaded!')

Data downloaded!

'wget' is not recognized as an internal or external command,
operable program or batch file.
```

```
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']

        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

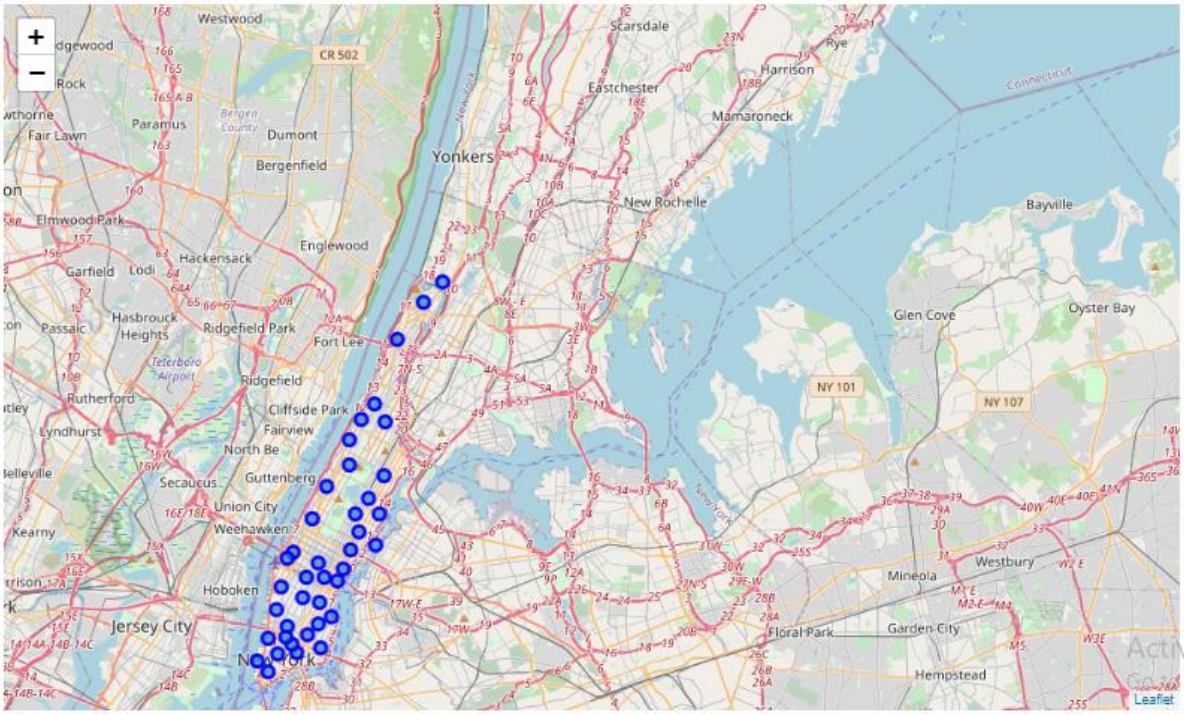
Exploring the Manhattan Borough

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

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

# add markers 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_manhattan)

map_manhattan
```



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 = 'Your Foursquare ID '
         CLIENT_SECRET = 'Your Secret Key '
         VERSION = '20201209'
```

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

```
neighborhoods = neighborhoods[neighborhoods['Borough'] == 'Manhattan'].reset_index(drop=True)
newyork_venues_north_indian = getNearbyVenues(names=neighborhoods['Neighborhood'], latitudes=neighborhoods['Latitude'], longitude
newyork_venues_north_indian.head()
```

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Marble Hill	40.876551	-73.91066	Marble Hill Unisex	40.877389	-73.909283	Salon / Barbershop
1	Marble Hill	40.876551	-73.91066	Wine & Liquors	40.874833	-73.909956	Wine Shop
2	Marble Hill	40.876551	-73.91066	Hernandez Grocery	40.875965	-73.912994	Deli / Bodega
3	Marble Hill	40.876551	-73.91066	St. Stephens UMC	40.876967	-73.909106	Church
4	Marble Hill	40.876551	-73.91066	Dunkin'	40.879296	-73.905135	Donut Shop

```
newyork_venues_north_indian.shape
```

```
(3977, 7)
```

Exploring the most common venues according to the neighbourhood

```
manhattan_venues = newyork_venues_north_indian.groupby('Venue Category').count().sort_values('Neighborhood',ascending=False).rese
manhattan_venues.head()
```

	Venue Category	Count
0	Residential Building (Apartment / Condo)	274
1	Office	250
2	Building	170
3	Doctor's Office	125
4	Salon / Barbershop	107

We also take a look at the Indian restaurants already located in the neighbourhood to find the competition.

```
manhattan_indian = newyork_venues_north_indian[newyork_venues_north_indian['Venue Category']=='Indian Restaurant'].reset_index(dr
manhattan_indian
```

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	East Harlem	40.792249	-73.944182	Om Indian Food	40.792014	-73.945332	Indian Restaurant
1	Lenox Hill	40.768113	-73.958860	Rangoli Exquisite Indian Cuisine	40.769428	-73.958382	Indian Restaurant
2	Chelsea	40.744035	-74.003116	Dil-e Punjab Deli	40.744845	-74.002528	Indian Restaurant
3	Tribeca	40.721522	-74.010683	Paisley	40.722307	-74.009397	Indian Restaurant
4	Manhattan Valley	40.797307	-73.964286	Roasted Masala	40.798433	-73.963708	Indian Restaurant

Since there are 394 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.

```
ny_grouped = ny_onehot.groupby('Neighborhood').mean().reset_index()  # Grouping and taking the mean
print(ny_grouped.shape)
ny_grouped.head()
```

```
(40, 394)
```

	Neighborhood	Harbor / Marina	Accessories Store	Acupuncturist	Adult Boutique	Advertising Agency	Afghan Restaurant	African Restaurant	American Restaurant	Animal Shelter	Antique Shop	Arcade	Arepa Restaurant	
0	Battery Park City	0.031579	0.000000	0.000000	0.0	0.0	0.0	0.000000	0.010526	0.0	0.000000	0.010526	0.000000	0
1	Carnegie Hill	0.000000	0.009091	0.018182	0.0	0.0	0.0	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0
2	Central Harlem	0.009009	0.000000	0.000000	0.0	0.0	0.0	0.000000	0.000000	0.0	0.000000	0.009009	0.000000	0
3	Chelsea	0.000000	0.000000	0.000000	0.0	0.0	0.0	0.000000	0.000000	0.0	0.009901	0.000000	0.009901	0
4	Chinatown	0.000000	0.008197	0.000000	0.0	0.0	0.0	0.008197	0.000000	0.0	0.000000	0.000000	0.000000	0

```
grouped_indian = ny_grouped[['Neighborhood','Indian Restaurant']]  # Only Indian Restaurant needs to be considered
grouped_indian
```

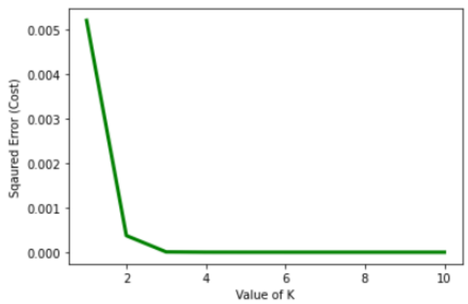
	Neighborhood	Indian Restaurant
0	Battery Park City	0.000000
1	Carnegie Hill	0.000000
2	Central Harlem	0.000000
3	Chelsea	0.009901
4	Chinatown	0.000000
5	Civic Center	0.000000
6	Clinton	0.000000
7	East Harlem	0.012195
8	East Village	0.000000
9	Financial District	0.000000
10	Flatiron	0.000000
11	Gramercy	0.000000

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 .

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

```
# 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()
```



```
kclusters = 3     # No.of Clusters
ny_grouped_clustering = grouped_indian.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]
```

```
array([0, 0, 0, 2, 0, 0, 0, 2, 0, 0])
```

```
grouped = grouped_indian.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)
```

```
(40, 3)
```

	Neighborhood	Indian Restaurant	Cluster Labels
0	Battery Park City	0.000000	0
1	Carnegie Hill	0.000000	0
2	Central Harlem	0.000000	0
3	Chelsea	0.009901	2
4	Chinatown	0.000000	0
5	Civic Center	0.000000	0
6	Clinton	0.000000	0
7	East Harlem	0.012195	2
8	East Village	0.000000	0
9	Financial District	0.000000	0

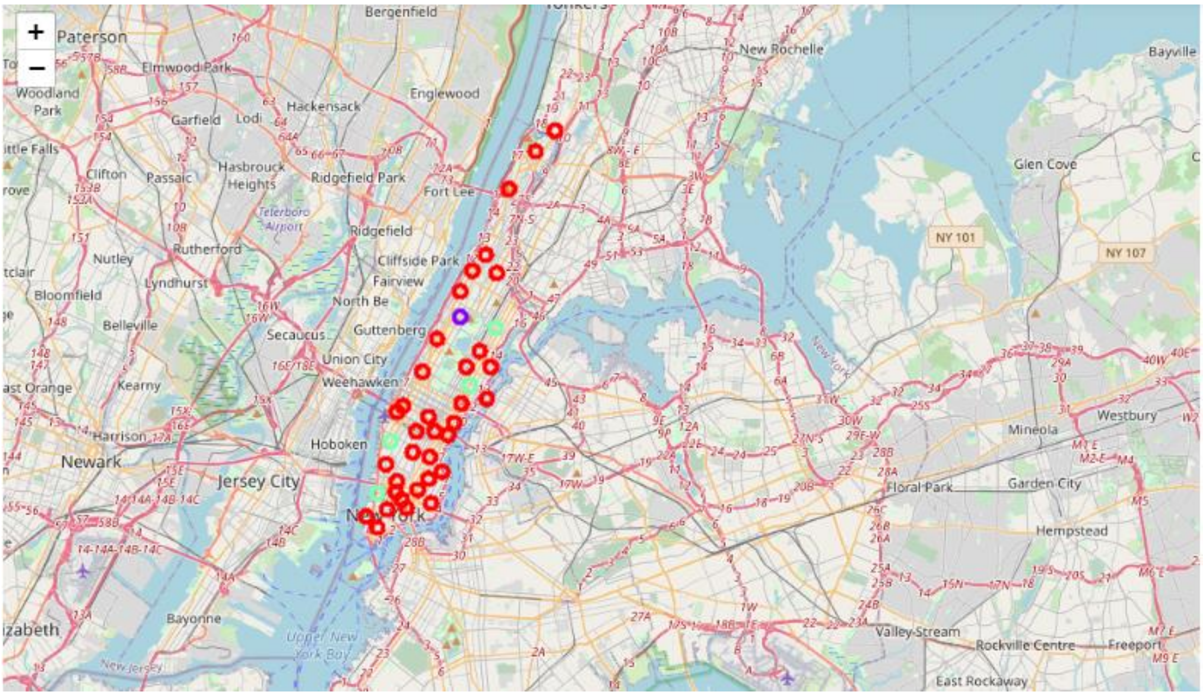
Visualization of Clusters using Folium

```
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['Neighborhoo
    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

```
t = to_merged[to_merged['Cluster Labels']==0]
print('Number of Indian Restaurants in Cluster 0: {}'.format(t[t['Venue Category']=='Indian Restaurant'].count()[0]))
t = to_merged[to_merged['Cluster Labels']==1]
print('Number of Indian Restaurants in Cluster 1: {}'.format(t[t['Venue Category']=='Indian Restaurant'].count()[0]))
t = to_merged[to_merged['Cluster Labels']==2]
print('Number of Indian Restaurants in Cluster 2: {}'.format(t[t['Venue Category']=='Indian Restaurant'].count()[0]))
```

```
Number of Indian Restaurants in Cluster 0: 0
Number of Indian Restaurants in Cluster 1: 1
Number of Indian Restaurants in Cluster 2: 4
```

We can see from the result that there are three clusters of which Cluster 0 has no Indian Restaurants and Cluster 2 has the most Indian Restaurants with 4 .

We then select cluster 0 to explore further as it is the most ideal Cluster with 0 restaurants .

```
]: temp = to_merged[to_merged['Cluster Labels']==0]
x = temp[temp['Venue Category'].str.contains('Restaurant')].groupby(['Neighborhood','Venue Category']).count()
y = x.reset_index().rename(columns={'Venue':'Count'})[['Neighborhood','Venue Category','Count']].sort_values(by='Count',ascending
y
```

]:

	Neighborhood	Venue Category	Count
0	Little Italy	Italian Restaurant	22
1	Chinatown	Chinese Restaurant	9
2	Clinton	Restaurant	5
3	Greenwich Village	Italian Restaurant	4
4	Midtown South	American Restaurant	4
5	Carnegie Hill	Mexican Restaurant	3

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

	Venue Category	Count
0	Residential Building (Apartment / Condo)	250
1	Office	238
2	Building	145
3	Doctor's Office	119
4	Salon / Barbershop	94
5	Laundry Service	54
6	Deli / Bodega	52
7	Art Gallery	52
8	Tech Startup	51
9	Food Truck	42

8. Discussion

```
t = to_merged[to_merged['Cluster Labels']==0]
print('Number of Indian Restaurants in Cluster 0: {}'.format(t[t['Venue Category']=='Indian Restaurant'].count()[0]))
t = to_merged[to_merged['Cluster Labels']==1]
print('Number of Indian Restaurants in Cluster 1: {}'.format(t[t['Venue Category']=='Indian Restaurant'].count()[0]))
t = to_merged[to_merged['Cluster Labels']==2]
print('Number of Indian Restaurants in Cluster 2: {}'.format(t[t['Venue Category']=='Indian Restaurant'].count()[0]))
```

```
Number of Indian Restaurants in Cluster 0: 0
Number of Indian Restaurants in Cluster 1: 1
Number of Indian Restaurants in Cluster 2: 4
```

We can see from the results above that cluster 0 with no restaurants is best suited for opening a new restaurant as it offers little competition in the matter of other competing Indian Restaurants.

We now further explore the neighbourhoods in detail in the Cluster 0.

```
]: temp = to_merged[to_merged['Cluster Labels']==0]
x = temp[temp['Venue Category'].str.contains('Restaurant')].groupby(['Neighborhood','Venue Category']).count()
y = x.reset_index().rename(columns={'Venue':'Count'})[['Neighborhood','Venue Category','Count']].sort_values(by='Count',ascending
y
```

]:

	Neighborhood	Venue Category	Count
0	Little Italy	Italian Restaurant	22
1	Chinatown	Chinese Restaurant	9
2	Clinton	Restaurant	5
3	Greenwich Village	Italian Restaurant	4
4	Midtown South	American Restaurant	4
5	Carnegie Hill	Mexican Restaurant	3

From the above result it can be seen that there are 5 neighbourhoods in Cluster 0 .

It is visible that the Neighbourhood of Little Italy and Chinatown are saturated with Italian and Chinese Restaurants. It is understood the these neighbourhoods will be tough to open an Indian Restaurant as it might not find enough footfall for our new restaurant.

We should focus our concentration on concentrating on other three neighbourhoods , which are Clinton ,Greenwich Village , Midtown South and Carnegie Hill.

We further take a look at the most popular venues in the cluster

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

	Venue Category	Count
0	Residential Building (Apartment / Condo)	250
1	Office	238
2	Building	145
3	Doctor's Office	119
4	Salon / Barbershop	94
5	Laundry Service	54
6	Deli / Bodega	52
7	Art Gallery	52
8	Tech Startup	51
9	Food Truck	42

We see that the cluster is a working cluster occupied mostly by Offices and shops . We also notice that as far as the eating scene is concerned ,it is dominated by Deli's and Food Trucks . Considering the offices and shops giving rise to fast moving footfall an Indian Food Tuck can also be considered an option instead of a Restaurant.

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.