RECOMMENDING WHERE TO OPEN A RESTAURANT

1. INTRODUCTION:

1.1 PROBLEM:

Data that might contribute to determine where to open a restaurant for someone who is planning to open a restaurant in a given city. This project aims to predict the correct neighbourhood in the city where a restaurant can be opened.

1.2 LOCATION:

The location which is taken in this project is the city New York which is situated in US country. New York City comprises 5 boroughs sitting where the Hudson River meets the Atlantic Ocean. At its core is Manhattan, a densely populated borough that's among the world's major commercial, financial and cultural centres. Its iconic sites include skyscrapers such as the Empire State Building and sprawling Central Park. Broadway theatre is staged in neon-lit Times Square.

2. Data Section:

- New York City's demographics show that it is a large and ethnically diverse metropolis. With its diverse culture, comes a diverse food item.
- There are many restaurants in New York City, each belonging to different categories like Chinese, Indian, and French etc.

For this project we need the following data:

New York City data that contains list Boroughs, Neighbourhoods along with their latitude and longitude.

- Data source : https://geo.nyu.edu/catalog/nyu_2451_34572
- Description: This data set contains the required information. And we will use this data set to explore various neighbourhoods of New York City.

Restaurants in each neighbourhood of New York City:

- Data source : Foursquare API
- Description: By using this API we will get all the venues in each neighbourhood. We can filter these venues to get only Indian restaurants.

Geo Space data:

- Data source : https://data.cityofnewyork.us/City-Government/BoroughBoundaries/tqmj-j8zm

- Description: By using this geo space data we will get the New York Borough boundaries that will help us visualize choropleth map.

3. Methodology section:

The data which is obtained for the data section is then used to process the information. Foursquare API is used to get all the restaurants which are there in New York City. Thus this information can be fitted to all the neighbourhoods. The Foursquare API allows application developers to interact with the foursquare platform. The API itself is a REST full set of addresses to which you can send requests, so there's really nothing to download onto your server.

The foursquare API needs client id and server to request for the restaurants.

```
[24]: CLIENT_ID = '3ZHPI4H2VNH0BYJURIRTVF244KLZEA2N51VUYOXD233GG3WR' # your Foursquare ID
      CLIENT_SECRET = 'ZQZNVKQQFFVQLMZJERF2V1B0GUCPQMUSCPVHOIURDHGWQRVU' # your Foursquare Secret
      VERSION = '20180605' # Foursquare API version

      print('Your credentails:')
      print('CLIENT_ID: ' + CLIENT_ID)
      print('CLIENT_SECRET:' + CLIENT_SECRET)

      Your credentails:
      CLIENT_ID: 3ZHPI4H2VNH0BYJURIRTVF244KLZEA2N51VUYOXD233GG3WR
      CLIENT_SECRET:ZQZNVKQQFFVQLMZJERF2V1B0GUCPQMUSCPVHOIURDHGWQRVU
```

URL is used to search for all the restaurants in the given city using search query option in the foursquare API URL.

Get the Newyork city latitude and longitude values.

The url is used to search for all the restaurants in the city

```
[26]: neighborhood_latitude=40.7280
      neighborhood_longitude=-74.0060
      radius=1000
      LIMIT=100
      query='Restaurant'
      url = 'https://api.foursquare.com/v2/venues/search?client_id={}&client_secret={}&ll={},{}&v={}&query={}&radius={}&limit={}'.
      url

[26]: 'https://api.foursquare.com/v2/venues/search?client_id=3ZHPI4H2VNH0BYJURIRTVF244KLZEA2N51VUYOXD233GG3WR&client_secret=ZQZNVK
      QQFFVQLMZJERF2V1B0GUCPQMUSCPVHOIURDHGWQRVU&ll=40.728,-74.006&v=20180605&query=Restaurant&radius=1000&limit=100'
```

The collected data is obtained in the results section with the GET request which is used to extract the collected data in the json format.

Send the GET request and examine the resutls

```
[27]: results = requests.get(url).json()
      results
```

```
[27]: {'meta': {'code': 200, 'requestId': '5eb978f86001fe001b8419bb'},
       'response': {'venues': [{'id': '4bbf6b6430c99c74302a5511',
          'name': 'PJ Charlton Italian Restaurant',
          'location': {'address': '549 Greenwich St',
           'lat': 40.72744633153894,
           'lng': -74.00865622880806,
           'labeledLatLngs': [{'label': 'display',
             'lat': 40.72744633153894,
             'lng': -74.00865622880806},
            {'label': 'entrance', 'lat': 40.727328, 'lng': -74.008749}],
           'distance': 232,
           'postalCode': '10013',
           'cc': 'US',
           'city': 'New York',
           'state': 'NY',
           'country': 'United States',
           'formattedAddress': ['549 Greenwich St',
```

Thus the obtained json data is now converted into the data frame which can be merged to our New York data.

```
newyork_venues = getNearbyVenues(names=neighborhoods['Neighborhood'],
                                 latitudes=neighborhoods['Latitude'],
                                 longitudes=neighborhoods['Longitude']
                                )

newyork_venues.head()
```

Fox Hills

[30]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude
0	Wakefield	40.894705	-73.847201	Big Daddy's Caribbean Taste Restaurant	40.899767	-73.857135
1	Wakefield	40.894705	-73.847201	Bay 241 Restaurant & Lounge	40.901347	-73.846554
2	Wakefield	40.894705	-73.847201	Kaieteur Restaurant & Bakery	40.899768	-73.857184
3	Wakefield	40.894705	-73.847201	Cool Running Restaurant	40.898399	-73.848810
4	Wakefield	40.894705	-73.847201	Bay restaurant	40.890850	-73.848860

Checking how many restaurants are there at each neighbourhood.

Let's check how many venues were returned for each neighborhood

```
[80]: restaurants=newyork_venues.groupby('Neighborhood').count()
      restaurants=restaurants.sort_values(by='Venue',ascending=True)
      restaurants=restaurants.reset_index()
      restaurants.head()
```

[80]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude
0	Somerville	1	1	1	1	1
1	Neponsit	1	1	1	1	1
2	Lighthouse Hill	1	1	1	1	1
3	Breezy Point	1	1	1	1	1
4	Manor Heights	1	1	1	1	1

Applying K means clustering algorithm to the above data will split the neighbourhoods into different clusters. K-means clustering is a method of vector quantization, originally from signal processing,

which aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. We here try to divide the neighbourhoods into 3 clusters where one cluster shows about the neighbourhoods with very less restaurants and the second cluster shows about neighbourhoods with medium restaurants and the last cluster tells about the neighbourhoods with many types of hotels and restaurants.

Cluster neighborhoods

[86]:
```
kclusters = 3

restaurants_grouped_clustering = restaurants.drop('Neighborhood',axis=1)

# run k-means clustering
kmeans = KMeans(n_clusters=kclusters, random_state=0).fit(restaurants_grouped_clustering)

# check cluster labels generated for each row in the dataframe
kmeans.labels_[0:190]
```

[86]:
```
array([2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], dtype=int32)
```

Thus the labels are provided to the neighbourhoods, if the label is 2 then there are less restaurants in the neighbourhood similarly if it is 1 then medium and if it is 0 then there are plenty restaurants around the neighbourhood. The obtained divided neighbourhoods are then circled on the map using folium. Folium makes it easy to visualize data that's been manipulated in Python on an interactive leaflet map. It enables both the binding of data to a map for choropleth visualizations as well as passing rich vector/raster/HTML visualizations as markers on the map.

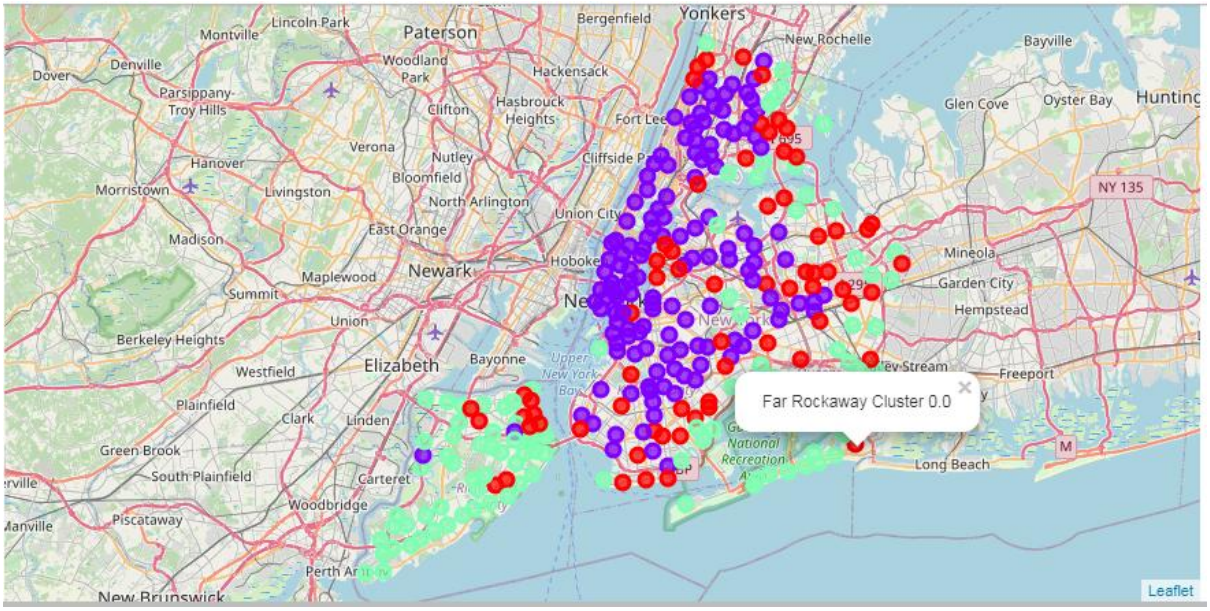
36]:
```
# create map
map_clusters = folium.Map(location=[latitude, longitude], zoom_start=11)

# 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(restaurants_merged1['Latitude'], restaurants_merged1['Longitude'], restaurants_merged1['N
    label = folium.Popup(str(poi) + ' Cluster ' + str(cluster), parse_html=True)
    folium.CircleMarker(
        [lat, lon],
        radius=5,
        popup=label,
        color=rainbow[int(cluster)-1],
        fill=True,
        fill_color=rainbow[int(cluster)-1],
        fill_opacity=0.7).add_to(map_clusters)

map_clusters
```

The green circles are for the areas with less restaurants, violet one with moderate and red one with plenty of restaurants.



.4. RESULTS SECTION:

The results section consists of the different neighbourhoods at different cluster labels. These are used by the person who is planning to open a restaurant. He uses the cluster label consisting of the neighbourhoods with less number of restaurants to open his/her restaurant.

Cluster 1 ¶

```
[156]: restaurants_merged1.loc[restaurants_merged1['Cluster Labels'] == 2, restaurants_merged1.columns[[1] + list(range(5, restaura
```

[156]:

	Neighborhood
1	Co-op City
2	Eastchester
10	Baychester
12	City Island
24	Hunts Point
27	Clason Point
28	Throgs Neck
36	North Riverdale
40	Castle Hill
42	Pelham Gardens

The next cluster tells about the neighbourhoods with moderate number of restaurants.

```
[157]: restaurants_merged1.loc[restaurants_merged1['Cluster Labels'] == 1, restaurants_merged1.columns[[1] + list(range(5, restaura
```

[157]:

	Neighborhood
0	Wakefield
5	Kingsbridge
6	Marble Hill
8	Norwood
9	Williamsbridge
11	Pelham Parkway
13	Bedford Park
14	University Heights
15	Morris Heights
16	Fordham
17	East Tremont
18	West Farms

The last cluster label tells about the neighbourhoods with plenty number of restaurants.

Cluster 3

```
[158]: restaurants_merged1.loc[restaurants_merged1['Cluster Labels'] == 0, restaurants_merged1.columns[[1] + list(range(5, restaura
```

[158]:

	Neighborhood
3	Fieldston
4	Riverdale
7	Woodlawn
22	Port Morris
26	Soundview
29	Country Club
31	Westchester Square
33	Morris Park
35	Spuyten Duyvil
37	Pelham Bay
38	Schuylerville

5. Discussion section:

Here from the results section we can observe how many neighbourhoods are present at each cluster and what all neighbourhoods are present at each cluster this makes easier for the person to choose a correct neighbourhood to open his restaurant.

```
[153]: restaurants1=restaurants.iloc[0:298,0:4]
       restaurants1.head()
       restaurants1['Cluster Labels'].value_counts()
```

```
[153]: 1    137
       2     94
       0     67
       Name: Cluster Labels, dtype: int64
```

6. Conclusion:

Thus based on the results section, this project concludes all the neighbourhoods which are great to open a restaurant as those neighbourhoods have very few restaurants and hotels. Thus it makes a great place in those neighbourhoods for anyone who is willing to open a restaurant or a hotel.

Cluster 1 ¶

[156]:
```
restaurants_merged1.loc[restaurants_merged1['Cluster Labels'] == 2, restaurants_merged1.columns[[1] + list(range(5, restaura
```

[156]:

	Neighborhood
1	Co-op City
2	Eastchester
10	Baychester
12	City Island
24	Hunts Point
27	Clason Point
28	Throgs Neck
36	North Riverdale
40	Castle Hill
42	Pelham Gardens