Coursera Capstone – "The Battle of the Neighborhoods"

IBM Applied Data Science Capstone 06/22/2020

F.Brouwn

<u>Exploring the cuisine of NYC neighborhoods</u>

Table of Contents:

1. Introduction

Background

New York City is the most populous city in the United States, residence of many headquarters e.g.: United Nations, international diplomacy, corporate companies. Residence of people from all over the world with different cultural backgrounds. Maybe it is the most diverse city (8.3 million people 800 languages) on the planet. These characteristics lead to high demand for various cuisine in NYC neighborhoods. This could be a very interesting opportunity for restaurant investors/owners to explore.

Problem

Finding the right neighborhood for opening a new restaurant can be hard. Especially, in big cities like NYC. A restaurant owner/investor needs a lot of data to make a substantiated decision, in which neighborhood/location he is willing to open a restaurant for a certain cuisine.

The idea of this project is to use Foursquare cuisine venue location data (using 'Places API') and regional clustering (using 'K-means clustering unsupervised machine learning techniques') to determine, which neighborhoods have potential to open a restaurant for a certain cuisine. This project will help to understand the cuisine diversity in neighborhoods and their surroundings. Exploratory Data Analysis will help to discover further details about cuisine diversity of the neighborhood.

Stakeholders

This analysis can be used for anyone who is interested in the distribution of different cuisines in NYC. It could be of interest for restaurant investors/owners, food specialist companies, Government for study propose etc.

2. Data

To examine the above said, following data sources are used:

New York City Dataset

Link: https://geo.nyu.edu/catalog/nyu_2451_34572

This New York City Neighborhood Names point file was created as a guide to New York City's neighborhoods that appear on the web resource, 'New York: A City of Neighborhoods.' This dataset will provide the addresses of neighborhood of NYC in json format. An extract of the json is as follows:


```
{'type': 'Feature',
'id': 'nyu_2451_34572.306',
'geometry': {'type': 'Point',
'coordinates': [-74.08173992211962, 40.61731079252983]},
'geometry_name': 'geom',
'properties': {'name': 'Fox Hills',
'stacked': 2,
'annoline1': 'Fox',
'annoline2': 'Hills',
'annoline3': None,
'annoangle': 0.0,
'borough': 'Staten Island',
'bbox': [-74.08173992211962,
```


Foursquare API

Link: https://developer.foursquare.com/docs

Foursquare API, a location data provider, will be used to make API calls to retrieve data about venues in different neighborhoods. Venues retrieved from all the neighborhoods are categorized broadly into 'Arts & Entertainment', 'College & University', 'Event', 'Food',

‘Nightlife Spot’, ‘Outdoors & Recreation’, etc. An extract of an API call is as follows:


```
'categories': [{'id': '4bf58dd8d48988d110941735',
'name': 'Italian Restaurant',
'pluralName': 'Italian Restaurants',
'shortName': 'Italian',
'icon': {'prefix': 'https://ss3.4sqi.net/img/categories_v2/food/italian_',
'suffix': '.png'},
'primary': True}],
'verified': False,
'stats': {'tipCount': 17},
'url': 'http://eccorestaurantny.com',
'price': {'tier': 4, 'message': 'Very Expensive', 'currency'
```

3. Methodology

In order to segment the neighborhoods of New York City, a dataset is required that contains the 5 boroughs and the neighborhoods, that exist in each borough, with respective latitude and longitude coordinates. This dataset is downloaded using the mentioned URL.

Once the .json file is downloaded, it is analyzed to understand the structure of the file. A python dictionary is returned by the URL and all the relevant data is found to be in the features key, which is basically a list of the neighborhoods. The dictionary is transformed, into a pandas dataframe, by looping through the data and filling the dataframe rows one at a time.

As a result, a dataframe is created with Borough, Neighborhood, Latitude and Longitude details of the New York City's neighborhood.

	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

Upon analysis, it is found that the dataframe consists of 5 boroughs and 306 neighborhoods.

'Geopy' library is used to get the latitude and longitude values of New York City, which was returned to be Latitude: 40.71, Longitude: -74.01.

The curated dataframe is then used to visualize by creating a map of New York City with neighborhoods superimposed on top. The following depiction is a map generated using python ‘folium’ library.

The Foursquare API is used to explore the neighborhoods and segment them. To access the API, ‘CLIENT_ID’, ‘CLIENT_SECRET’ and ‘VERSION’ is defined.

There are many endpoints available on Foursquare for various GET requests. To explore the cuisines, it is required that all the venues extracted are from ‘Food’ category. Foursquare Venue Category Hierarchy is retrieved and returned request is further analyzed.

Upon analysis, it is found that there are 10 major or parent categories of venues, under which all the other sub-categories are included.

4d4b7105d754a06379d81259 Travel & Transport

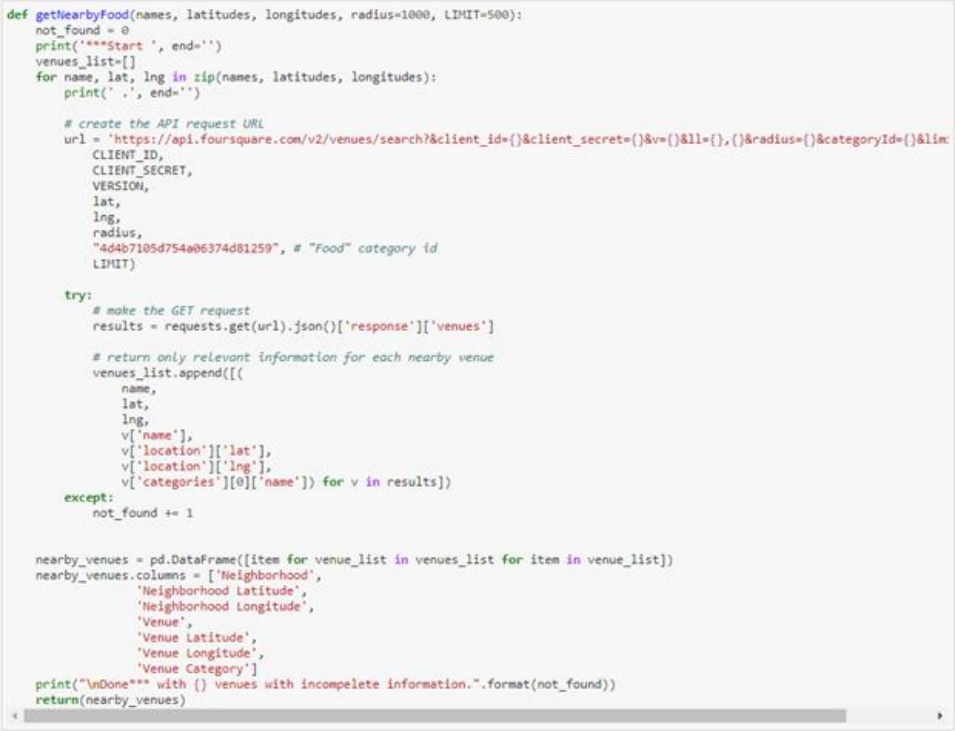
The ‘FOOD’ category (ID’ = ‘*4d4b7105d754a06374d81259*’) is the matter of interest. A function is created to return a dictionary with ‘Category ID’ & ‘Category Name’ of ‘Food’ & it’s sub-categories.

```
[ ]  # function to flatten a 'parent_id' category, returns all categories if checkParentID = False
     def flatten_Hierarchy(category_list, checkParentID, category_dict, parent_id = ''):
         for data in category_list:

             if checkParentID == True and data['id'] == parent_id:
                 category_dict[data['id']] = data['name']
                 flatten_Hierarchy(category_list = data['categories'], checkParentID = False, category_dict = category_dict)

             elif checkParentID == False:
                 category_dict[data['id']] = data['name']
                 if len(data['categories']) != 0:
                     flatten_Hierarchy(category_list = data['categories'], checkParentID = False, category_dict = category_dict)

         return category_dict
```

A GET request is created to search for Venue with ‘Category ID’ = ‘*4d4b7105d754a06374d81259*’, which is the ‘Category ID’ for ‘Food’, and radius = 500 meters.

The aim is to segment the neighborhoods of New York City with respect to the ‘Food’ in its vicinity, it is further required to fetch the data from all the 306 neighborhoods’ venues.

To overcome redundancy, a function ‘getNearbyFood’ is created. This functions loop through all the neighborhoods of New York City and creates an API request URL with radius = 500, LIMIT = 100. By limit, it is defined that maximum 100 nearby venues should be returned. The data is then appended to a python ‘list’. From the python ‘list’ a ‘dataframe’ is created which is returned by the function.



```
def getNearbyFood(names, latitudes, longitudes, radius=1000, LIMIT=500):
    not_found = 0
    print('***Start ', end='')
    venues_list=[]
    for name, lat, lng in zip(names, latitudes, longitudes):
        print(' .', end='')

        # create the API request URL
        url = 'https://api.foursquare.com/v2/venues/search?&client_id={}&client_secret={}&v={}&ll={},{}&radius={}&categoryId={}&lim
            CLIENT_ID,
            CLIENT_SECRET,
            VERSION,
            lat,
            lng,
            radius,
            "4d4b7105d754a06374d81259", # "Food" category id
            LIMIT)

        try:
            # make the GET request
            results = requests.get(url).json()['response']['venues']

            # return only relevant information for each nearby venue
            venues_list.append([(
                name,
                lat,
                lng,
                v['name'],
                v['location']['lat'],
                v['location']['lng'],
                v['categories'][0]['name']) for v in results])
        except:
            not_found += 1


    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']
    print("\nDone*** with {} venues with incompelete information.".format(not_found))
    return(nearby_venues)
```

The Pickle library is used to create a dump file '.pkl', which is collected from the GET request command. This file can be used to retrieve a python object structure. This is a crucial step as it will counter any redundant requests to the Foursquare API, which is chargeable over the threshold limits.

The returned 'nyc_venues' dataframe is as follows:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Wakefield	40.894705	-73.847201	Central Deli	40.896728	-73.844387	Deli / Bodega
1	Wakefield	40.894705	-73.847201	Carvel Ice Cream	40.890487	-73.848568	Ice Cream Shop
2	Wakefield	40.894705	-73.847201	Cooler Runnings Jamaican Restaurant Inc	40.898083	-73.850259	Caribbean Restaurant
3	Wakefield	40.894705	-73.847201	Him Health Food Market	40.897665	-73.854638	Food
4	Wakefield	40.894705	-73.847201	Wakefield Deli	40.901998	-73.846910	Deli / Bodega

There are now two python 'dataframe' are available:

1. 'neighborhoods' which contains the Borough, Neighborhood, Latitude and Longitude details of the New York City's neighborhood, and

2. 'nyc_venues' which is a merger between 'neighborhoods' dataframe and its 'Food' category venues searched with 'Radius' = 500 meters and 'Limit' = 100. Also, each venue has its own Latitude, Longitude and Category.

4. Exploratory Data Analysis

The merged dataframe ‘nyc_venues’ has all the required information (13,724 venues total).

```
[ ] print(nyc_venues.shape)
    nyc_venues.head()
```

```
(13724, 7)
```

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Wakefield	40.894705	-73.847201	Central Deli	40.896728	-73.844387	Deli / Bodega
1	Wakefield	40.894705	-73.847201	Carvel Ice Cream	40.890487	-73.848568	Ice Cream Shop
2	Wakefield	40.894705	-73.847201	Cooler Runnings Jamaican Restaurant Inc	40.898083	-73.850259	Caribbean Restaurant
3	Wakefield	40.894705	-73.847201	Him Health Food Market	40.897665	-73.854638	Food
4	Wakefield	40.894705	-73.847201	Wakefield Deli	40.901998	-73.846910	Deli / Bodega

There are 190 unique categories:

```
There are 190 uniques categories.
Venue Category
Deli / Bodega                 1266
Pizza Place                   1084
Coffee Shop                    936
Chinese Restaurant             684
Donut Shop                     644
Fast Food Restaurant           601
Bakery                         584
Italian Restaurant             447
Bagel Shop                     397
Café                           379
Mexican Restaurant             377
Ice Cream Shop                 332
Sandwich Place                 325
Caribbean Restaurant           322
Fried Chicken Joint            310
American Restaurant            304
```

5. Data Cleaning

Where interested in the variation of cuisine types within a neighborhood, so we remove all the venues which have a general food category like f.e. coffee shop, café etc.. First, we collect al unique food categories, then manually we create a file with all ‘general’ general food categories.

```
'Soup Place','Rock Club','Residential Building (Apartment / Condo)','Laundry Service',
'Government Building','Bowling Alley','Nightclub','Park','Moving Target','Bike Shop','Beer Store',
'Hobby Shop','Chocolate Shop','Food Service','Indoor Play Area','Record Shop','Whisky Bar','Dosa Place']
```

By subtraction of the ‘Full’ - and the ‘’General’ food categories, we are able to create a dataframe with the required venues/food categories for further analyzing (now we got 89 food categories, before it was 190, that is almost 50% reduction).

6. Feature Engineering

Each neighborhood is analyzed individually to understand the most common cuisine being served within its 500 meters of the vicinity.

The above process is taken forth by using ‘one hot encoding’ function of python ‘pandas’ library. One hot encoding converts the categorical variables (which are ‘Venue Category’) into a form that could be provided to ML algorithms to do a better job in prediction.

```
[ ] # one hot encoding
    nyc_onehot = pd.get_dummies(nyc_venues[['Venue Category']], prefix="", prefix_sep="")
    nyc_onehot.head()
```

	Afghan Restaurant	African Restaurant	American Restaurant	Arepa Restaurant	Argentinian Restaurant	Asian Restaurant	Australian Restaurant	Austrian Restaurant	BBQ Joint	Brazilian Restaurant	Burrito Place	Cajun / Creole Restaurant	Car Rest
0	0	0	0	0	0	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	0	0	0	

The ‘Neighborhood’ column should be added again, after above converting, the size of the 6483 records.

The top 10 ‘Venue Categories’ by concurrency:

```
[ ] venue_counts_described = venue_counts.describe().transpose()
```

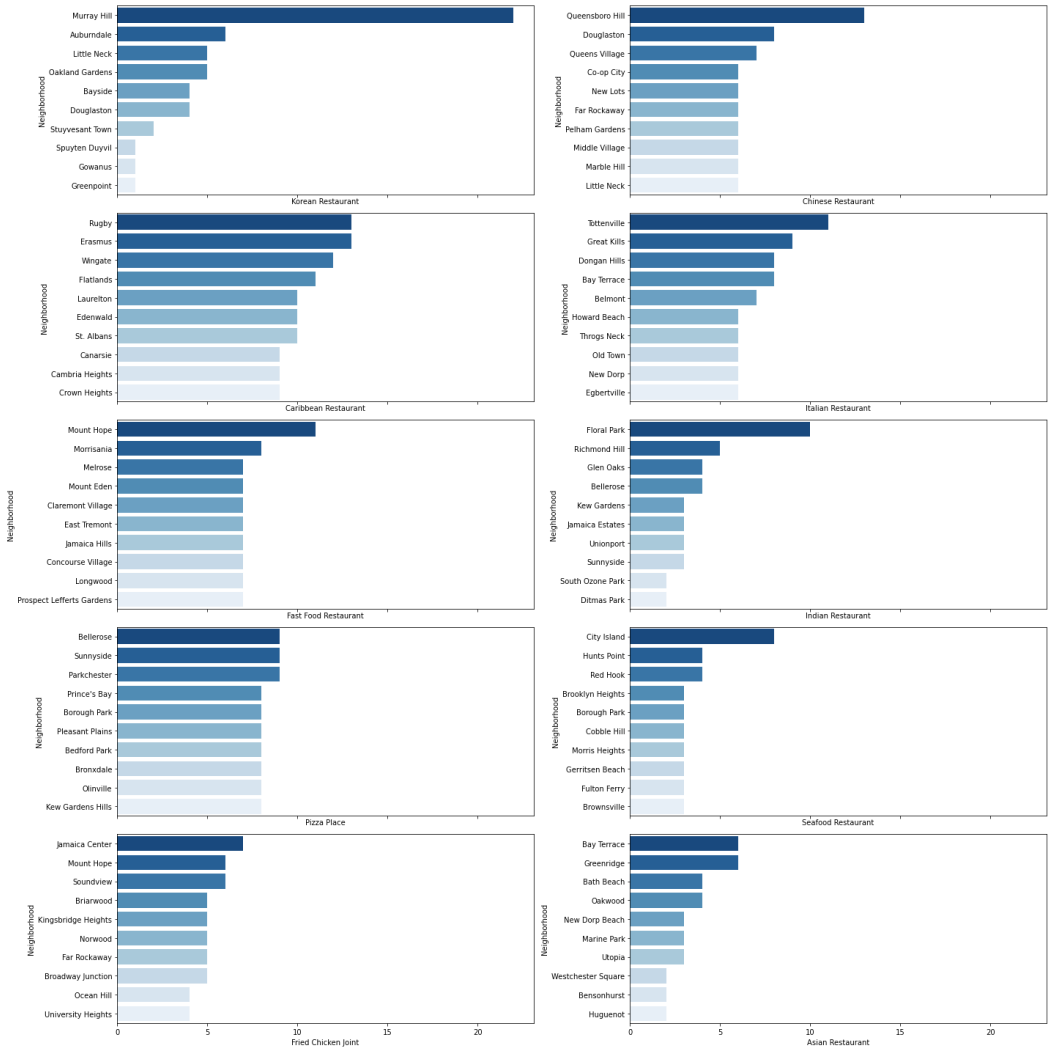
```
[ ] venue_top10 = venue_counts_described.sort_values('max', ascending=False)[0:10]
    venue_top10
```

	count	mean	std	min	25%	50%	75%	max
Korean Restaurant	302.0	0.238411	1.426883	0.0	0.0	0.0	0.0	22.0
Chinese Restaurant	302.0	2.264901	1.887895	0.0	1.0	2.0	3.0	13.0
Caribbean Restaurant	302.0	1.066225	2.393704	0.0	0.0	0.0	1.0	13.0
Italian Restaurant	302.0	1.480132	1.760947	0.0	0.0	1.0	2.0	11.0
Fast Food Restaurant	302.0	1.990066	1.941820	0.0	0.0	2.0	3.0	11.0
Indian Restaurant	302.0	0.311258	0.894319	0.0	0.0	0.0	0.0	10.0
Pizza Place	302.0	3.589404	1.995915	0.0	2.0	3.0	5.0	9.0
Seafood Restaurant	302.0	0.533113	0.906178	0.0	0.0	0.0	1.0	8.0
Fried Chicken Joint	302.0	1.026490	1.314105	0.0	0.0	1.0	2.0	7.0
Asian Restaurant	302.0	0.500000	0.857835	0.0	0.0	0.0	1.0	6.0

7. Data Visualization

These top 10 categories are further plotted individually on bar graph using python ‘seaborn’ library.

Top 10 neighborhoods for a food category:



The neighborhoods cuisine can be defined by the top 5 food categories. A dataframe is created with the top 5 most common venues categories in the neighborhood.

```
[ ] for ind in np.arange(nyc_grouped.shape[0]):
        neighborhoods_venues_sorted.iloc[ind, 1:] = return_most_common_venues(nyc_grouped.iloc[ind, :], num_top_venues)

    neighborhoods_venues_sorted.head()
```

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue
0	Allerton	Pizza Place	Chinese Restaurant	Mexican Restaurant	Fast Food Restaurant	Fried Chicken Joint
1	Annadale	Pizza Place	American Restaurant	Italian Restaurant	Sushi Restaurant	Japanese Restaurant
2	Arden Heights	Pizza Place	American Restaurant	Italian Restaurant	Mexican Restaurant	Chinese Restaurant
3	Arlington	Pizza Place	Fast Food Restaurant	American Restaurant	Spanish Restaurant	Latin American Restaurant
4	Arrochar	Italian Restaurant	Pizza Place	Latin American Restaurant	Japanese Restaurant	Mediterranean Restaurant

8. Machine Learning

'k-means' is an unsupervised machine learning algorithm which creates clusters of data points aggregated together because of certain similarities. This algorithm will be used to count neighborhoods for each cluster label for variable cluster size.

To implement this algorithm, it is very important to determine the optimal number of clusters (i.e. k). There are 2 most popular methods for the same, namely 'The Elbow Method' and 'The Silhouette Method'.

The Elbow Method

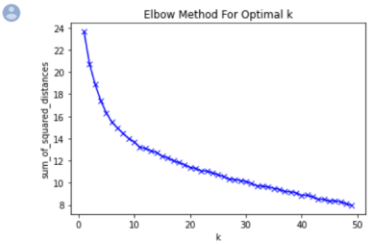
The Elbow Method calculates the sum of squared distances of samples to their closest cluster center for different values of 'k'. The optimal number of clusters is the value after which there is no significant decrease in the sum of squared distances.

Following is an implementation of this method (with varying number of clusters from 1 to 49):

```
[ ] sum_of_squared_distances = []
    K = range(1,50)
    for k in K:
        print(k, end=' ')
        kmeans = KMeans(n_clusters=k).fit(nyc_grouped_clustering)
        sum_of_squared_distances.append(kmeans.inertia_)
```

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49
```

```
[ ] plt.plot(K, sum_of_squared_distances, 'bx-')
    plt.xlabel('k')
    plt.ylabel('sum_of_squared_distances')
    plt.title('Elbow Method For Optimal k');
```



Elbow method does not seem to help us to determine the optimal number of clusters (due to gradual decrease). Let's use another method.

The Silhouette Method

As quoted in Wikipedia — "The Silhouette Method measures how similar a point is to its own cluster (cohesion) compared to other clusters (separation)."

The following is an implementation of this method. As it requires minimum 2 clusters to define dissimilarity number of clusters (i.e. 'k') will vary from 2 to 49:

There is a peak at k = 2, k = 4 and k = 8. Two and four clusters will give a very broad classification of the venues.

There is a peak at k=2 and k=4. These would give a very broad classification of the venues, let's use K=8.

k-Means

Following code block runs the k-Means algorithm with the number of clusters = 8 and prints the counts of neighborhoods assigned to different clusters:

```
[ ] # set number of clusters
    kclusters = 8

    # run k-means clustering
    kmeans = KMeans(init="k-means++", n_clusters=kclusters, n_init=50).fit(nyc_grouped_clustering)

    print(Counter(kmeans.labels_))

    Counter({0: 61, 3: 55, 4: 53, 2: 46, 7: 32, 1: 27, 6: 25, 5: 3})
```

The cluster labels curated are added to the dataframe to get the desired results of segmenting the neighborhood based upon the most common venues in its vicinity.

4	7	Arrochar	Italian Restaurant	Pizza Place	Japanese Restaurant	Polish Restaurant	Latin American Restaurant	Staten Island	40.596313	-74.067124

The New York City's neighborhoods are visualized utilizing the python 'folium' library, showing the cluster segmentation of the New York City's neighborhoods:

9. Results

Cluster — 0

Following are the results of the Cluster analysis:

```
[ ] for col in required_column:
        print(cluster_0[col].value_counts(ascending = False))
        print("-----------------------------------------------")
```

```
Chinese Restaurant       27
Pizza Place              26
Sushi Restaurant          2
Korean Restaurant         2
Greek Restaurant          1
American Restaurant       1
Fried Chicken Joint       1
Vietnamese Restaurant     1
Caribbean Restaurant      1
Name: 1st Most Common Venue, dtype: int64
-----------------------------------------------
Chinese Restaurant       22
Pizza Place              12
Italian Restaurant        4
Fast Food Restaurant      4
Korean Restaurant         3
Cantonese Restaurant      2
Greek Restaurant          2
Fried Chicken Joint       2
Indian Restaurant         2
Japanese Restaurant       2
Sushi Restaurant          1
Vietnamese Restaurant     1
Russian Restaurant        1
Mexican Restaurant        1
Caribbean Restaurant      1
American Restaurant       1
Taco Place                1
Name: 2nd Most Common Venue, dtype: int64
-----------------------------------------------
Queens           30
Brooklyn         11
Staten Island    10
Bronx             6
Manhattan         5
Name: Borough, dtype: int64
-----------------------------------------------
```

Cluster — 1

```
[ ]  for col in required_column:
         print(cluster_1[col].value_counts(ascending = False))
         print("---------------------------------------------")
```

```
Pizza Place                   7
Italian Restaurant            6
American Restaurant           6
Seafood Restaurant            3
Sushi Restaurant              2
Eastern European Restaurant   1
New American Restaurant       1
Fast Food Restaurant          1
Spanish Restaurant            1
Name: 1st Most Common Venue, dtype: int64
---------------------------------------------
American Restaurant       8
Italian Restaurant        5
Pizza Place               5
Seafood Restaurant        4
Russian Restaurant        2
Turkish Restaurant        1
New American Restaurant   1
Mexican Restaurant        1
Spanish Restaurant        1
Name: 2nd Most Common Venue, dtype: int64
---------------------------------------------
Brooklyn         13
Manhattan         6
Bronx             5
Staten Island     2
Queens            2
Name: Borough, dtype: int64
---------------------------------------------
```

Cluster — 2

```
[ ]  for col in required_column:
         print(cluster_2[col].value_counts(ascending = False))
         print("----------------------------------------------")
```

```
American Restaurant        7
Pizza Place                6
Seafood Restaurant         3
New American Restaurant    1
Spanish Restaurant         1
Name: 1st Most Common Venue, dtype: int64
----------------------------------------------
American Restaurant    6
Pizza Place            6
Italian Restaurant     3
Seafood Restaurant     3
Name: 2nd Most Common Venue, dtype: int64
----------------------------------------------
Brooklyn         8
Bronx            4
Queens           3
Staten Island    2
Manhattan        1
Name: Borough, dtype: int64
----------------------------------------------
```

Cluster — 3

```
----------------------------------------------
```

Cluster — 4

```
Bronx            9
Manhattan        1
```

Cluster — 5

```
[ ] for col in required_column:
        print(cluster_5[col].value_counts(ascending = False))
        print("---------------------------------------------")
```

```
Caribbean Restaurant    22
Fast Food Restaurant     2
Name: 1st Most Common Venue, dtype: int64
---------------------------------------------
Chinese Restaurant      8
Pizza Place             6
Fried Chicken Joint     4
Fast Food Restaurant    4
Caribbean Restaurant    2
Name: 2nd Most Common Venue, dtype: int64
---------------------------------------------
Brooklyn    12
Queens       8
Bronx        4
Name: Borough, dtype: int64
---------------------------------------------
```

Cluster — 6

```
[ ] for col in required_column:
        print(cluster_6[col].value_counts(ascending = False))
        print("---------------------------------------------")
```

```
Caribbean Restaurant    23
American Restaurant      1
Fast Food Restaurant     1
Name: 1st Most Common Venue, dtype: int64
---------------------------------------------
Chinese Restaurant      8
Pizza Place             7
Fried Chicken Joint     5
Fast Food Restaurant    4
Caribbean Restaurant    1
Name: 2nd Most Common Venue, dtype: int64
---------------------------------------------
Brooklyn         12
Queens            8
Bronx             4
Staten Island     1
Name: Borough, dtype: int64
---------------------------------------------
```

Cluster — 7

```
[ ]  for col in required_column:
         print(cluster_7[col].value_counts(ascending = False))
         print("---------------------------------------------")
```

```
Italian Restaurant     26
Pizza Place             5
Seafood Restaurant      1
American Restaurant     1
Name: 1st Most Common Venue, dtype: int64
---------------------------------------------
Pizza Place             14
Italian Restaurant       7
Mexican Restaurant       4
Fast Food Restaurant     3
Chinese Restaurant       2
Asian Restaurant         1
American Restaurant      1
Japanese Restaurant      1
Name: 2nd Most Common Venue, dtype: int64
---------------------------------------------
Staten Island    15
Queens           11
Bronx             3
Manhattan         2
Brooklyn          2
Name: Borough, dtype: int64
---------------------------------------------
```

10. Conclusion

Tabulating the results of the k-Mean unsupervised machine learning algorithm:

Cluster	**1st most common venue**	**1st most common venue**	**Borough**
0	Chinese restaurant	Pizza Place	Queens
1	Pizza Place, Italian restaurant	American restaurant, Italian restaurant	Brooklyn, Manhattan
2	American restaurant, Pizza Place	American restaurant, Pizza Place	Brooklyn, Bronx
3	Chinese restaurant	Chinese restaurant	Queens
4	Pizza Place	Chinese restaurant	Staten Island

5	Caribbean restaurant	Chinese restaurant	Brooklyn
6	Caribbean restaurant	Chinese restaurant	Brooklyn, Queens
7	Italian restaurant	Pizza place	Staten island, Queens

It is obvious from the analysis that Pizza Place is the most common venue across all the clusters or neighborhoods. As Pizza Place is a ready-to-go place for New York City, it is kept aside to rename the clusters.

Following could be the name of the clusters segmented and curated by k-Means unsupervised machine learning algorithm:

. Cluster 0 — *Chinese*

· Cluster 1 — *Italian*

· Cluster 2 — *American*

· Cluster 3 — *Chinese*

· Cluster 4 — *Pizza Place, Chinese*

· Cluster 5 — *Caribbean*

· Cluster 6 — *Caribbean*

· Cluster 7 — *Italian*