Exploring the Taste of New York

26 December, 2019

Table of Contents

Introduction

Background

New York City is the most populous city in the United States, home to the headquarters of the United Nations and an important center for international diplomacy. It just might be the most diverse city on the planet, as it is home to over
8.6 million people and over 800 languages.

As quoted in an article - What Food Tells Us About Culture
"Traditional cuisine is passed down from one generation to the next. It also operates as an expression of cultural identity. Immigrants bring the food of their countries with them wherever they go and cooking traditional food is a way of preserving their culture when they move to new places."

Problem

Undoubtedly, Food Diversity is an important part of an ethnically diverse metropolis. The idea of this project is to categorically segment the neighborhoods of New York City into major clusters and examine their cuisines. A desirable intention is to examine the neighborhood cluster's food habits and taste. Further examination might reveal if food has any relationship with the diversity of a neighborhood. This project will help to understand the diversity of a neighborhood by leveraging venue data from Foursquare's 'Places API' and 'k-means clustering' unsupervised machine learning algorithm. Exploratory Data Analysis (EDA) will help to discover further about the culture and diversity of the neighborhood.

Stakeholders

This quantifiable analysis can be used to understand the distribution of different cultures and cuisines over 'the most diverse city on the planet – New York City'. Also, it can be utilized by a new food vendor who is willing to open his or her restaurant. Or by a government authority to examine and study their city's culture diversity better.

Data

To examine the above said, following data sources will be 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.' Best estimates of label centroids were established at a 1:1,000 scale, but are ideally viewed at a 1:50,000 scale. 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 RESTful API calls to retrieve data about venues in different neighborhoods. This is the link to Foursquare Venue Category Hierarchy. 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'
```


Methodology

Download and Explore New York City Dataset

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 using the following depicted loop.

```
for data in neighborhoods_data:
    borough = 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)
```

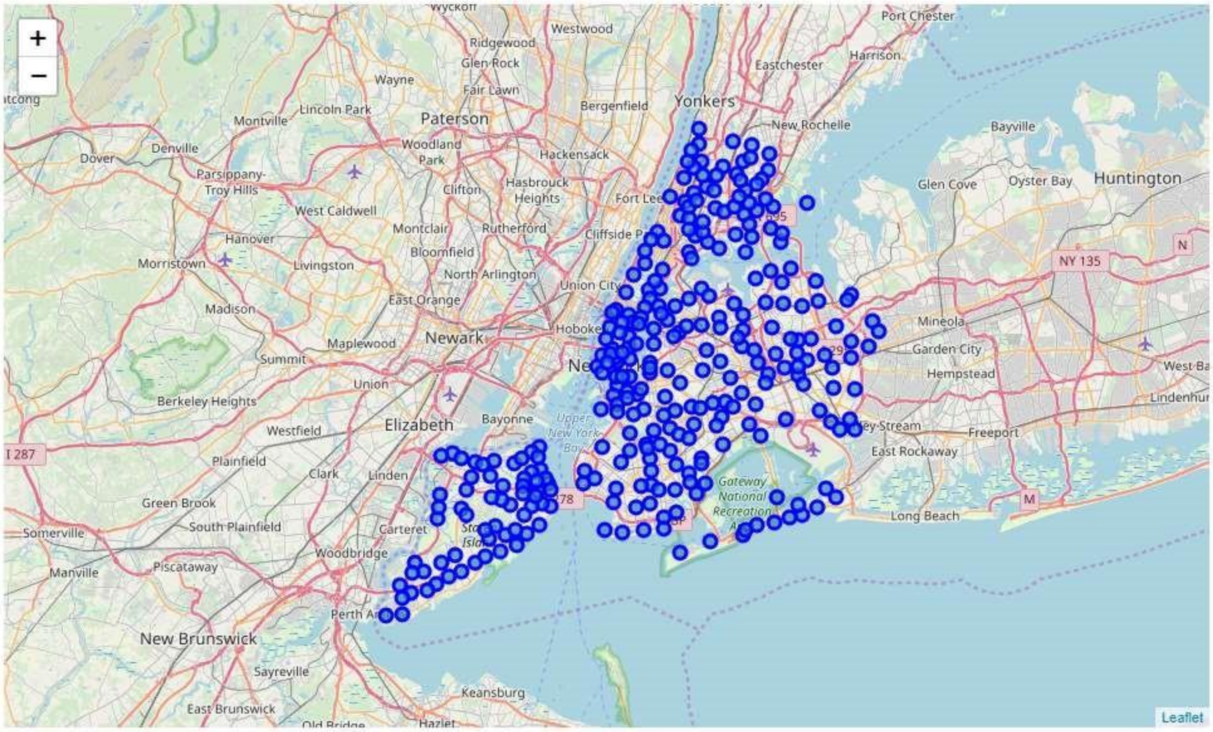
As a result, a dataframe is created with Borough, Neighborhood, Latitude and Longitude details of the New York City's neighborhood.

	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.

Further, '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.



RESTful API Calls to Foursquare

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. But, to explore the cuisines, it is required that all the venues extracted are from ‘Food’ category. Foursquare Venue Category Hierarchy is retrieved using the following code block:

```
url = 'https://api.foursquare.com/v2/venues/categories?&client_id={}&client_secret={}&v={}'.format(
    CLIENT_ID,
    CLIENT_SECRET,
    VERSION)
category_results = requests.get(url).json()
```

The returned request is further analyzed:

```
for key, value in category_results['response']['categories'][0].items():
    print(key, len(str(value)))
```

```
id 24
name 20
pluralName 20
shortName 20
icon 98
categories 15910
```

Upon analysis, it is found that there are 10 major or parent categories of venues, under which all the other sub-categories are included. Following depiction shows the ‘Category ID’ and ‘Category Name’ retrieved from API:

```python
for data in category_list:
    print(data['id'], data['name'])
```

```
4d4b7104d754a06370d81259 Arts & Entertainment
4d4b7105d754a06372d81259 College & University
4d4b7105d754a06373d81259 Event
4d4b7105d754a06374d81259 Food
4d4b7105d754a06376d81259 Nightlife Spot
4d4b7105d754a06377d81259 Outdoors & Recreation
4d4b7105d754a06375d81259 Professional & Other Places
4e67e38e036454776db1fb3a Residence
4d4b7105d754a06378d81259 Shop & Service
4d4b7105d754a06379d81259 Travel & Transport
```

As said earlier, the 'FOOD' category in the above depiction is the matter of interest. A function is created to return a dictionary with 'Category ID' & 'Category Name' of 'Food' & it's sub-categories.

```python
# 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
```

This above function takes the parent 'Category ID' and returns the 'Category Name' and 'Category ID' of all the sub-categories.

```python
category_dict
```

```
{'4d4b7105d754a06374d81259': 'Food',
 '503288ae91d4c4b30a586d67': 'Afghan Restaurant',
 '4bf58dd8d48988d1c8941735': 'African Restaurant',
 '4bf58dd8d48988d10a941735': 'Ethiopian Restaurant',
 '4bf58dd8d48988d14e941735': 'American Restaurant',
 '4bf58dd8d48988d157941735': 'New American Restaurant',
 '4bf58dd8d48988d142941735': 'Asian Restaurant',
 '56aa371be4b08b9a8d573568': 'Burmese Restaurant',
 '52e81612bcbc57f1066b7a03': 'Cambodian Restaurant',
 '4bf58dd8d48988d145941735': 'Chinese Restaurant',
 '52af3a5e3cf9994f4e043bea': 'Anhui Restaurant',
 '52af3a723cf9994f4e043bec': 'Beijing Restaurant',
 '52af3a7c3cf9994f4e043bed': 'Cantonese Restaurant',
 '58daa1558bbb0b01f18ec1d3': 'Cha Chaan Teng',
 '52af3a673cf9994f4e043beb': 'Chinese Aristocrat Restaurant',
 '52af3a903cf9994f4e043bee': 'Chinese Breakfast Place',
 '4bf58dd8d48988d1f5931735': 'Dim Sum Restaurant',
 '52af3a9f3cf9994f4e043bef': 'Dongbei Restaurant',
 '52af3aaa3cf9994f4e043bf0': 'Fujian Restaurant',
 '52af3ab53cf9994f4e043bf1': 'Guizhou Restaurant',
 '52af3abe3cf9994f4e043bf2': 'Hainan Restaurant',
 '52af3ac83cf9994f4e043bf3': 'Hakka Restaurant',
 '52af3ad23cf9994f4e043bf4': 'Henan Restaurant',
 '52af3add3cf9994f4e043bf5': 'Hong Kong Restaurant',
 '52af3af23cf9994f4e043bf7': 'Huaiyang Restaurant',
 '52af3ae63cf9994f4e043bf6': 'Hubei Restaurant',
 '52af3afc3cf9994f4e043bf8': 'Hunan Restaurant',
 '52af3b053cf9994f4e043bf9': 'Imperial Restaurant',
 '52af3b213cf9994f4e043bfa': 'Jiangsu Restaurant',
```

To further understand the results of GET Request, the first neighborhood of the 'New York City' dataset is explored. The first neighborhood returned is 'Wakefield' with Latitude 40.89 and Longitude -73.85.

Then, a GET request URL is created to search for Venue with 'Category ID' = '4d4b7105d754a06374d81259', which is the 'Category ID' for 'Food', and radius = 500 meters.

```
LIMIT = 1 # limit of number of venues returned by Foursquare API
radius = 500 # define radius
categoryId = '4d4b7105d754a06374d81259' # category ID for "Food"

# create URL

url = 'https://api.foursquare.com/v2/venues/search?&client_id={}&client_secret={}&v={}&ll={},{}&radius={}&categoryId={}&limit={}'.fo
    CLIENT_ID,
    CLIENT_SECRET,
    VERSION,
    neighborhood_latitude,
    neighborhood_longitude,
    radius,
    categoryId,
    LIMIT)
url # display URL
```

The returned request is then examined, which is as follows:

```
results['response']['venues']
```


```
[{'id': '4c783cef3badb1f7e4244b54',
  'name': 'Carvel Ice Cream',
  'location': {'address': '1006 E 233rd St',
   'lat': 40.890486685759605,
   'lng': -73.84856772568665,
   'labeledLatLngs': [{'label': 'display',
     'lat': 40.890486685759605,
     'lng': -73.84856772568665}],
   'distance': 483,
   'postalCode': '10466',
   'cc': 'US',
   'city': 'Bronx',
   'state': 'NY',
   'country': 'United States',
   'formattedAddress': ['1006 E 233rd St',
    'Bronx, NY 10466',
    'United States']},
  'categories': [{'id': '4bf58dd8d48988d1c9941735',
    'name': 'Ice Cream Shop',
    'pluralName': 'Ice Cream Shops',
    'shortName': 'Ice Cream',
    'icon': {'prefix': 'https://ss3.4sqi.net/img/categories_v2/food/icecream_',
     'suffix': '.png'},
    'primary': True}],
  'referralId': 'v-1571614359',
  'hasPerk': False}]
```


The request returned the ‘Category Name’ of the venue as 'Carvel Ice Cream' which is of 'Food' category.

As, 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 this data from all the 306 neighborhoods' venues.

To overcome the redundancy of the process followed above, 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. Further, the GET request is made to Foursquare API and only relevant information for each nearby venue is extracted from it. The data is then appended to a python 'list'. Lastly the python 'list' is unfolded or flattened to append it to dataframe being returned by the function.

It is inquisitive to know that Foursquare API returns all the sub-categories, if a toplevel category is specified in the GET Request.

```
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)
```

Pickle

Pickle is a very important and easy-to-use library. It is used to serialize the information retrieved from GET requests, to make a persistent '.pkl' file. This file can later be deserialized to retrieve an exact 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.

```
import pickle # to serialize and deserialize a Python object structure
try:
    with open('nyc_food_venues.pkl', 'rb') as f:
        nyc_venues = pickle.load(f)
    print("---Dataframe Existed and Deserialized---")
except:
    nyc_venues = getNearbyFood(names=neighborhoods['Neighborhood'],
                                    latitudes=neighborhoods['Latitude'],
                                    longitudes=neighborhoods['Longitude']
                                   )
    with open('nyc_food_venues.pkl', 'wb') as f:
        pickle.dump(nyc_venues, f)
    print("---Dataframe Created and Serialized---")
```

```
---Dataframe Existed and Deserialized---
```

The returned ‘dataframe’ is as follows:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Wakefield	40.894705	-73.847201	Lollipops Gelato	40.894123	-73.845892	Dessert Shop
1	Wakefield	40.894705	-73.847201	Pitman Deli	40.894149	-73.845748	Food
2	Wakefield	40.894705	-73.847201	Carvel Ice Cream	40.890487	-73.848568	Ice Cream Shop
3	Wakefield	40.894705	-73.847201	Burger King	40.895532	-73.856436	Fast Food Restaurant
4	Wakefield	40.894705	-73.847201	Dunkin'	40.890459	-73.849089	Donut Shop

As of now, two python ‘dataframe’ are created:

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.

Exploratory Data Analysis

The merged dataframe ‘nyc_venues’ has all the required information. The size of this dataframe is determined, and it is found that there are total 14,047 venues.

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

```
(14047, 7)
```

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Wakefield	40.894705	-73.847201	Lollipops Gelato	40.894123	-73.845892	Dessert Shop
1	Wakefield	40.894705	-73.847201	Pitman Deli	40.894149	-73.845748	Food
2	Wakefield	40.894705	-73.847201	Carvel Ice Cream	40.890487	-73.848568	Ice Cream Shop
3	Wakefield	40.894705	-73.847201	Burger King	40.895532	-73.856436	Fast Food Restaurant
4	Wakefield	40.894705	-73.847201	Dunkin'	40.890459	-73.849089	Donut Shop

Now, it is important to find out that how many unique categories can be curated from all the returned venues. There are 194 such categories, with most occurring venues as follows:

```
print('There are {} uniques categories.'.format(len(nyc_venues['Venue Category'].unique())))
nyc_venues.groupby('Venue Category')['Venue Category'].count().sort_values(ascending=False)
```

```
There are 194 uniques categories.

Venue Category
Deli / Bodega                     1136
Pizza Place                       1078
Coffee Shop                        919
Donut Shop                         710
Fast Food Restaurant               664
Chinese Restaurant                 619
Italian Restaurant                 544
Bakery                             544
American Restaurant                428
Café                               401
Caribbean Restaurant               359
Bagel Shop                         357
Mexican Restaurant                 352
Sandwich Place                     332
Diner                              313
Ice Cream Shop                     256
Fried Chicken Joint                254
Restaurant                         224
Food                               204
Burger Joint                       204
Seafood Restaurant                 155
Sushi Restaurant                   152
Latin American Restaurant          144
Asian Restaurant                   142
Spanish Restaurant                 140
Japanese Restaurant                136
Food Truck                         136
Bar                                120
Juice Bar                          111
New American Restaurant            108
Dessert Shop                       104
Breakfast Spot                      99
Indian Restaurant                   98
Food Court                          95
Thai Restaurant                     94
Taco Place                          92
Korean Restaurant                   88
BBQ Joint                           88
```

Data Cleaning

It is crucial to understand that the point of interest in the project is to understand the cultural diversity of a neighborhood by clustering it categorically, using the venues' categories. Thus, it is important to remove all the venues from the 'dataframe' which have generalized categories. Here, by generalized, it means that these categorized venues are common across different cultures and food habits. Example of categories of this type of venues are Coffee Shop, Cafe, etc.

So, firstly all the unique categories are fed into a python 'list'.

```
# list all the categories
unique_categories = nyc_venues['Venue Category'].unique().tolist()
unique_categories
```

```
['Dessert Shop',
 'Food',
 'Ice Cream Shop',
 'Fast Food Restaurant',
 'Donut Shop',
 'Caribbean Restaurant',
 'Bakery',
 'Sandwich Place',
 'Italian Restaurant',
 'Comfort Food Restaurant',
 'Fried Chicken Joint',
 'Deli / Bodega',
 'Food Truck',
 'Chinese Restaurant',
 'Pizza Place',
 'Southern / Soul Food Restaurant',
 'Halal Restaurant',
 'Asian Restaurant',
 'Bagel Shop',
 'American Restaurant',
 'Burger Joint',
 'Restaurant',
 'Mexican Restaurant',
 'Seafood Restaurant',
 'Frozen Yogurt Shop',
 'Spanish Restaurant',
```

Then, manually the categories are determined to be ‘general’ (as explained above). This data pre-preparation totally depends upon the ‘Data Analyst’ discretion and can be modified as required. Following are the categories listed as ‘general’:

```
# manually create a list of generalized categories
general_categories = ['Dessert Shop','Food','Ice Cream Shop','Donut Shop','Bakery','Sandwich Place','Comfort Food Restaurant',
                      'Deli / Bodega','Food Truck','Bagel Shop','Burger Joint','Restaurant','Frozen Yogurt Shop','Coffee Shop',
                      'Diner','Wings Joint','Café','Juice Bar','Breakfast Spot','Grocery Store','Bar','Cupcake Shop',
                      'Pub','Fish & Chips Shop','Cafeteria','Other Nightlife','Arcade','Hot Dog Joint','Food Court',
                      'Health Food Store','Convenience Store','Food & Drink Shop','Cocktail Bar','Cheese Shop',
                      'Snack Place','Sports Bar','Lounge','Theme Restaurant','Buffet','Bubble Tea Shop','Building',
                      'Irish Pub','College Cafeteria','Tea Room','Supermarket','Hotpot Restaurant','Gastropub','Beer Garden',
                      'Fish Market','Beer Bar','Clothing Store','Music Venue','Bistro','Salad Place','Wine Bar','Gourmet Shop',
                      'Indie Movie Theater','Art Gallery','Gift Shop','Pie Shop','Fruit & Vegetable Store',
                      'Street Food Gathering','Dive Bar','Factory','Farmers Market','Mac & Cheese Joint','Creperie',
                      'Candy Store','Event Space','Skating Rink','Miscellaneous Shop','Gas Station','Organic Grocery',
                      'Pastry Shop','Club House','Flea Market','Hotel','Furniture / Home Store','Bookstore','Pet Café',
                      'Gym / Fitness Center','Flower Shop','Financial or Legal Service','Hotel Bar','Hookah Bar','Poke Place',
                      'Market','Gluten-free Restaurant','Smoothie Shop','Butcher','Food Stand','Beach Bar','Beach',
                      'Soup Place','Rock Club','Residential Building (Apartment / Condo)','Laundry Service',
                      'Government Building','Bowling Alley','Nightclub','Park','Moving Target']
```

A simple subtraction of two python ‘list’ i.e ‘unique_categories’ and ‘general_categories’ gives a ‘list’ of all the categories which are required for further analysis.

Following image depicts the result of the above activity:

```
# fetch all the required food categories
food_categories =  list(set(unique_categories) - set(general_categories))
food_categories
```

```
['Paella Restaurant',
 'Brazilian Restaurant',
 'Steakhouse',
 'Mexican Restaurant',
 'Vegetarian / Vegan Restaurant',
 'Italian Restaurant',
 'Peruvian Restaurant',
 'Jewish Restaurant',
 'African Restaurant',
 'Ukrainian Restaurant',
 'Sri Lankan Restaurant',
 'Mediterranean Restaurant',
 'Caucasian Restaurant',
 'English Restaurant',
 'American Restaurant',
 'Korean Restaurant',
 'Vietnamese Restaurant',
 'Cantonese Restaurant',
 'Tapas Restaurant',
 'Hong Kong Restaurant',
 'Persian Restaurant',
 'Dosa Place',
 'Greek Restaurant',
 'Australian Restaurant',
 'Himalayan Restaurant',
 'Taiwanese Restaurant',
 'Seafood Restaurant',
 'Salvadoran Restaurant',
 'Japanese Restaurant',
 'Venezuelan Restaurant',
 'Ramen Restaurant',
```

The python 'list' curated above, is used to remove all the venues with categories not in 'food_categories', and the following dataframe is retrieved:

```
nyc_venues = nyc_venues[nyc_venues['Venue Category'].isin(food_categories)].reset_index()
nyc_venues.head(5)
```

	index	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	3	Wakefield	40.894705	-73.847201	Burger King	40.895532	-73.856436	Fast Food Restaurant
1	5	Wakefield	40.894705	-73.847201	Cooler Runnings Jamaican Restaurant Inc	40.898276	-73.850381	Caribbean Restaurant
2	9	Wakefield	40.894705	-73.847201	McDonald's	40.902645	-73.849485	Fast Food Restaurant
3	10	Wakefield	40.894705	-73.847201	Ripe Kitchen & Bar	40.898152	-73.838875	Caribbean Restaurant
4	11	Wakefield	40.894705	-73.847201	Frank and Johnies	40.905019	-73.858392	Italian Restaurant

Again, the number of unique categories is examined, and it is found that there are only 92 of them, as compared to 194 earlier. That means, almost 50% of the data was a noise for the analysis. This essential step, data cleaning, helped to capture the data points of interest.

Feature Engineering

Now, each neighborhood is analyzed individually to understand the most common cuisine being served within its 500 meters of 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	Burmese Restaurant	Burrito Place	Cajun / Creole Restaurant	Ca Re
0	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	0	
2	0	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	0	
4	0	0	0	0	0	0	0	0	0	0	0	0	0	

Upon converting the categorical variables, as shown above, 'Neighborhood' column is added back which results into the following:

```
# move neighborhood column to the first column
Neighborhood = nyc_onehot['Neighborhood']

nyc_onehot.drop(labels=['Neighborhood'], axis=1,inplace = True)
nyc_onehot.insert(0, 'Neighborhood', Neighborhood)

nyc_onehot.head()
```

	Neighborhood	Afghan Restaurant	African Restaurant	American Restaurant	Arepa Restaurant	Argentinian Restaurant	Asian Restaurant	Australian Restaurant	Austrian Restaurant	BBQ Joint	Brazilian Restaurant	Burmese Restaurant	Burrito Place
0	Wakefield	0	0	0	0	0	0	0	0	0	0	0	0
1	Wakefield	0	0	0	0	0	0	0	0	0	0	0	0
2	Wakefield	0	0	0	0	0	0	0	0	0	0	0	0
3	Wakefield	0	0	0	0	0	0	0	0	0	0	0	0
4	Wakefield	0	0	0	0	0	0	0	0	0	0	0	0

The size of the new dataframe 'nyc_onehot' is examined and it is found that there are around 6,846 data points all together.

Further, number of venues of each category in each neighborhood are counted.

```
venue_counts = nyc_onehot.groupby('Neighborhood').sum()
venue_counts.head(5)
```

Neighborhood	Afghan Restaurant	African Restaurant	American Restaurant	Arepa Restaurant	Argentinian Restaurant	Asian Restaurant	Australian Restaurant	Austrian Restaurant	BBQ Joint	Brazilian Restaurant	Burmese Restaurant	Burrito Place	Re:
Allerton	0	0	0	0	0	0	0	0	0	0	0	0	
Annadale	0	0	3	0	0	0	0	0	0	0	0	0	
Arden Heights	0	0	3	0	0	0	0	0	1	0	0	0	
Arlington	0	0	2	0	0	1	0	0	0	0	0	0	
Arrochar	0	0	0	0	0	0	0	0	0	0	0	0	

It is observed that, in the first five neighborhoods of the dataframe, 'Annadale', 'Arden Heights' and 'Arlington' has 3, 3, 2 'American Restaurant' in its 500 meters vicinity.

The top 10 'Venue Categories' can also be found by counting their occurrences. This analysis is depicted below which shows that 'Korean Restaurant', 'Chinese Restaurant', 'Caribbean Restaurant', 'Indian Restaurant', and 'Fast Food Restaurant' are among the top 5.

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

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

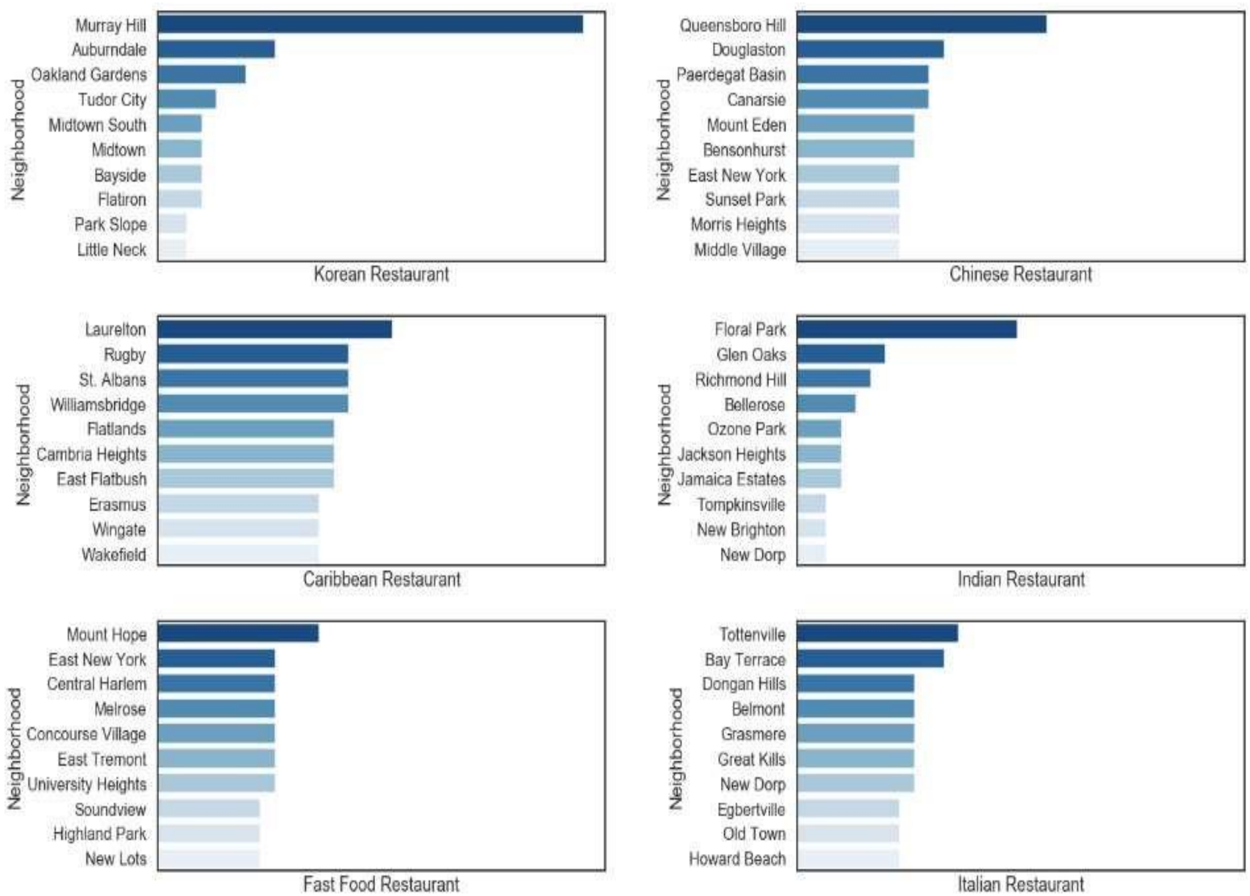
	count	mean	std	min	25%	50%	75%	max
Korean Restaurant	302.0	0.291391	1.828491	0.0	0.0	0.0	0.0	29.0
Chinese Restaurant	302.0	2.049669	2.083199	0.0	1.0	2.0	3.0	17.0
Caribbean Restaurant	302.0	1.188742	2.785965	0.0	0.0	0.0	1.0	16.0
Indian Restaurant	302.0	0.324503	1.123885	0.0	0.0	0.0	0.0	15.0
Fast Food Restaurant	302.0	2.198675	2.054150	0.0	1.0	2.0	3.0	11.0
Italian Restaurant	302.0	1.801325	1.983386	0.0	0.0	1.0	3.0	11.0
Pizza Place	302.0	3.569536	2.190314	0.0	2.0	3.0	5.0	10.0
Seafood Restaurant	302.0	0.513245	0.849950	0.0	0.0	0.0	1.0	7.0
New American Restaurant	302.0	0.357616	0.745702	0.0	0.0	0.0	0.0	6.0
Thai Restaurant	302.0	0.311258	0.726210	0.0	0.0	0.0	0.0	6.0

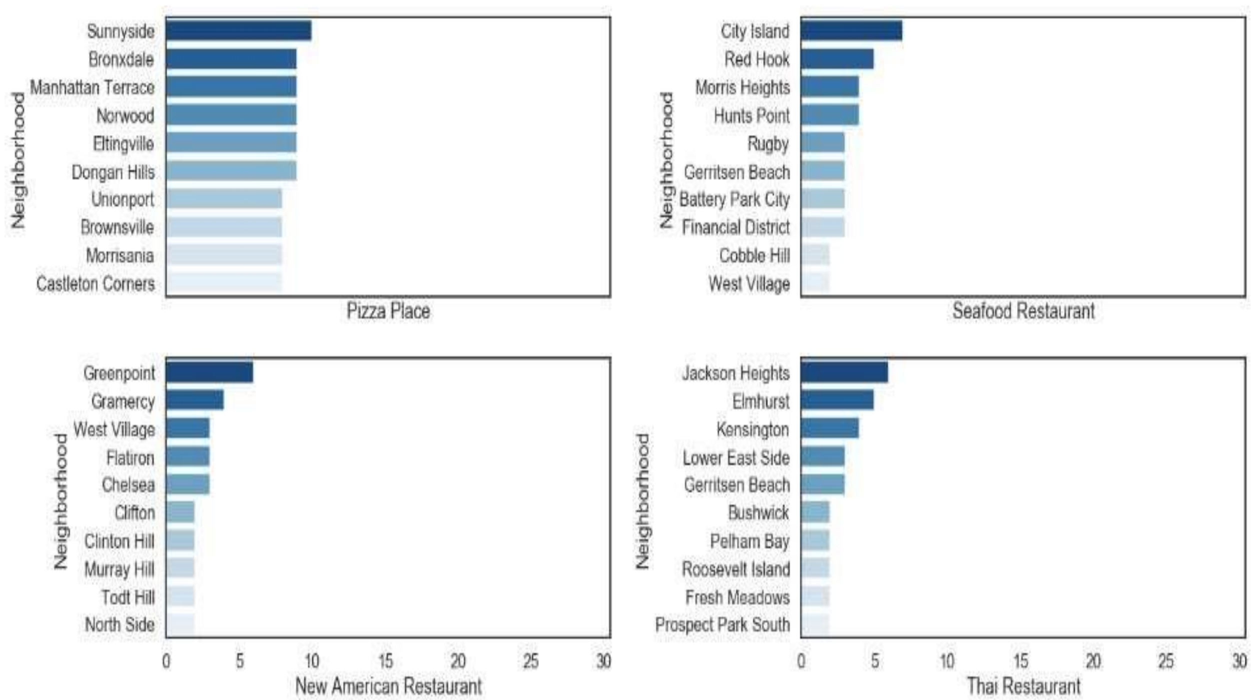
Data Visualization

These top 10 categories are further plotted individually on bar graph using python 'seaborn' library. The following code block creates the graph of top 10 neighborhoods for a category.

```python
import seaborn as sns
import matplotlib.pyplot as plt

fig, axes =plt.subplots(5, 2, figsize=(20,20), sharex=True)
axes = axes.flatten()
object_bol = df.dtypes == 'object'

for ax, category in zip(axes, venue_top10_list):
    data = venue_counts[[category]].sort_values([category], ascending=False)[0:10]
    pal = sns.color_palette("Blues", len(data))
    sns.barplot(x=category, y=data.index, data=data, ax=ax, palette=np.array(pal[::-1]))

plt.tight_layout()
plt.show()
```

The result of the above code block returns the following bar graphs:





Next, the rows of the neighborhood are grouped together and the frequency of occurrence of each category is calculated by taking the mean.

```
nyc_grouped = nyc_onehot.groupby('Neighborhood').mean().reset_index()
nyc_grouped.head()
```

	Neighborhood	Afghan Restaurant	African Restaurant	American Restaurant	Arepa Restaurant	Argentinian Restaurant	Asian Restaurant	Australian Restaurant	Austrian Restaurant	BBQ Joint	Brazilian Restaurant	Burmese Restaurant	Burri Pla
0	Allerton	0.0	0.0	0.000000	0.0	0.0	0.000000	0.0	0.0	0.000000	0.0	0.0	0
1	Annadale	0.0	0.0	0.176471	0.0	0.0	0.000000	0.0	0.0	0.000000	0.0	0.0	0
2	Arden Heights	0.0	0.0	0.176471	0.0	0.0	0.000000	0.0	0.0	0.058824	0.0	0.0	0
3	Arlington	0.0	0.0	0.105263	0.0	0.0	0.052632	0.0	0.0	0.000000	0.0	0.0	0
4	Arrochar	0.0	0.0	0.000000	0.0	0.0	0.000000	0.0	0.0	0.000000	0.0	0.0	0

As the limit is set to be 100, there will be many venues being returned by the Foursquare API. But a neighborhood food habit can be defined by the top 5 venues in its vicinity. Following ‘for’ loop creates a dataframe to record the abovementioned data points:

```
num_top_venues = 5

indicators = ['st', 'nd', 'rd']

# create columns according to number of top venues
columns = ['Neighborhood']
for ind in np.arange(num_top_venues):
    try:
        columns.append('{}{} Most Common Venue'.format(ind+1, indicators[ind]))
    except:
        columns.append('{}th Most Common Venue'.format(ind+1))
```

```
# create a new dataframe
neighborhoods_venues_sorted = pd.DataFrame(columns=columns)
neighborhoods_venues_sorted['Neighborhood'] = nyc_grouped['Neighborhood']
```

Further, the above created dataframe is fed with the top 5 most common venues categories in the respective 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	Mexican Restaurant	Fried Chicken Joint	Pizza Place	Chinese Restaurant	Fast Food Restaurant
1	Annadale	Pizza Place	American Restaurant	Sushi Restaurant	Italian 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	Peruvian Restaurant	Spanish Restaurant
4	Arrochar	Italian Restaurant	Pizza Place	Steakhouse	Middle Eastern Restaurant	Chinese Restaurant

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

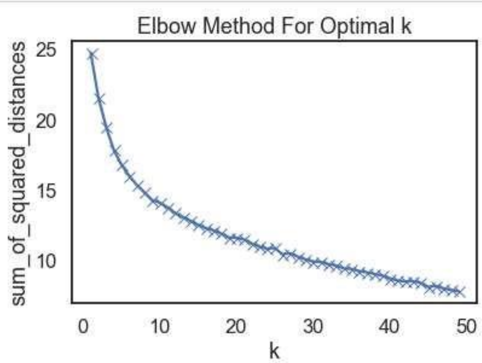
The Elbow Method calculates the sum of squared distances of samples to their closest cluster center for different values of 'k'. The optimal number of clusters is the value after which there is no significant decrease in the sum of squared distances.

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
```

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



Sometimes, Elbow method does not give the required result, which happened in this case. As, there is a gradual decrease in the sum of squared distances, optimal number of clusters can not be determined. To counter this, another method can be implemented, as discussed below.

The Silhouette Method

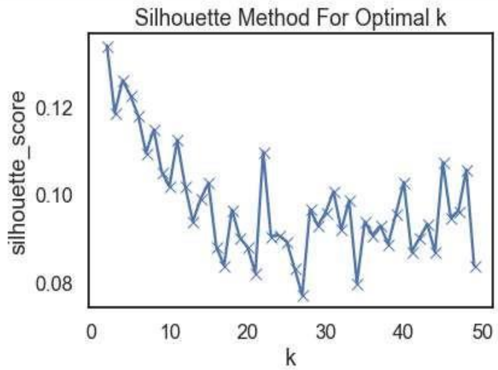
As quoted in Wikipedia – “The Silhouette Method measures how similar a point is to its own cluster (cohesion) compared to other clusters (separation).”

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:

```
from sklearn.metrics import silhouette_score

sil = []
K_sil = range(2,50)
# minimum 2 clusters required, to define dissimilarity
for k in K_sil:
    print(k, end=' ')
    kmeans = KMeans(n_clusters = k).fit(nyc_grouped_clustering)
    labels = kmeans.labels_
    sil.append(silhouette_score(nyc_grouped_clustering, labels, metric = 'euclidean'))
```

```
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 4
```

```
plt.plot(K_sil, sil, 'bx-')
plt.xlabel('k')
plt.ylabel('silhouette_score')
plt.title('Silhouette Method For Optimal k')
plt.show()
```



There is a peak at k = 2, k = 4 and k = 8. Two and four number of clusters will cluster the neighborhoods very broadly. Therefore, number of clusters (i.e. ‘k’) is chosen to be 8.

k-Means

Following code block runs the k-Means algorithm with 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({2: 91, 3: 53, 1: 50, 4: 50, 0: 26, 5: 20, 6: 11, 7: 1})
```

Further 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:

```
# add clustering labels
try:
    neighborhoods_venues_sorted.drop('Cluster Labels', axis=1)
except:
    neighborhoods_venues_sorted.insert(0, 'Cluster Labels', kmeans.labels_)
```

```
neighborhoods_venues_sorted.head(50)
```

	Cluster Labels	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue
0	1	Allerton	Mexican Restaurant	Fried Chicken Joint	Pizza Place	Chinese Restaurant	Fast Food Restaurant
1	3	Annadale	Pizza Place	American Restaurant	Sushi Restaurant	Italian Restaurant	Japanese Restaurant
2	3	Arden Heights	Pizza Place	American Restaurant	Italian Restaurant	Mexican Restaurant	Chinese Restaurant
3	1	Arlington	Pizza Place	Fast Food Restaurant	American Restaurant	Peruvian Restaurant	Spanish Restaurant
4	4	Arrochar	Italian Restaurant	Pizza Place	Steakhouse	Middle Eastern Restaurant	Chinese Restaurant
5	0	Arverne	Chinese Restaurant	Pizza Place	American Restaurant	Asian Restaurant	Thai Restaurant
6	2	Astoria	Greek Restaurant	Pizza Place	Fast Food Restaurant	Vietnamese Restaurant	American Restaurant
7	4	Astoria Heights	Italian Restaurant	Chinese Restaurant	Greek Restaurant	Pizza Place	Sushi Restaurant
8	2	Auburndale	Korean Restaurant	Greek Restaurant	Pizza Place	Italian Restaurant	American Restaurant
9	1	Bath Beach	Chinese Restaurant	Fast Food Restaurant	Cantonese Restaurant	Sushi Restaurant	Vietnamese Restaurant
10	2	Battery Park City	Seafood Restaurant	Italian Restaurant	Pizza Place	Fast Food Restaurant	New American Restaurant

Now, ‘neighborhoods_venues_sorted’ is merged with ‘nyc_data’ to add the Borough, Latitude and Longitude for each neighborhood.

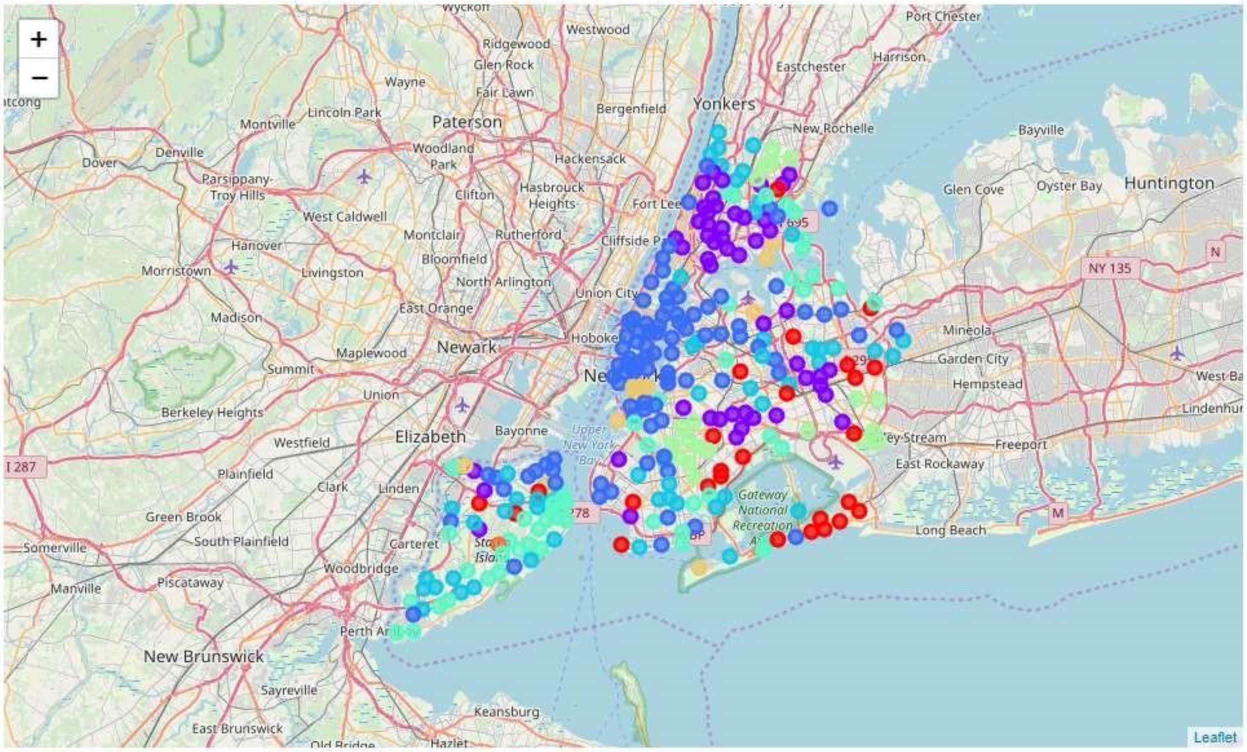
```
# merge neighborhoods_venues_sorted with nyc_data to add latitude/longitude for each neighborhood
nyc_merged = neighborhoods_venues_sorted.join(neighborhoods.set_index('Neighborhood'), on='Neighborhood')
nyc_merged.head()
```

	Cluster Labels	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	Borough	Latitude	Longitude
0	1	Allerton	Mexican Restaurant	Fried Chicken Joint	Pizza Place	Chinese Restaurant	Fast Food Restaurant	Bronx	40.865788	-73.859319
1	3	Annadale	Pizza Place	American Restaurant	Sushi Restaurant	Italian Restaurant	Japanese Restaurant	Staten Island	40.538114	-74.178549
2	3	Arden Heights	Pizza Place	American Restaurant	Italian Restaurant	Mexican Restaurant	Chinese Restaurant	Staten Island	40.549286	-74.185887
3	1	Arlington	Pizza Place	Fast Food Restaurant	American Restaurant	Peruvian Restaurant	Spanish Restaurant	Staten Island	40.635325	-74.165104
4	4	Arrochar	Italian Restaurant	Pizza Place	Steakhouse	Middle Eastern Restaurant	Chinese Restaurant	Staten Island	40.596313	-74.067124

Again, the New York City’s neighborhoods are visualized by using the code block as shown, which utilizes the python ‘folium’ library.

```
# create map
map_clusters = folium.Map(location=[latitude, longitude], zoom_start=10)

# set color scheme for the clusters
colors_array = cm.rainbow(np.linspace(0, 1, kclusters))
rainbow = [colors.rgb2hex(i) for i in colors_array]

# add markers to the map
markers_colors = []
for lat, lon, poi, cluster in zip(nyc_merged['Latitude'], nyc_merged['Longitude'], nyc_merged['Neighborhood'], nyc_merged['Cluster L
    label = folium.Popup(str(poi) + ' Cluster ' + str(cluster), parse_html=True)
    folium.CircleMarker(
        [lat, lon],
        radius=5,
        popup=label,
        color=rainbow[cluster-1],
        fill=True,
        fill_color=rainbow[cluster-1],
        fill_opacity=0.7).add_to(map_clusters)

map_clusters
```

Following map is generated which shows the desired segmentation of the New York City's neighborhoods:



Results

Cluster – 0

```
cluster_0 = nyc_merged.loc[nyc_merged['Cluster Labels'] == 0, nyc_merged.columns[1:12]]
cluster_0.head(5)
```

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	Borough	Latitude	Longitude
36	Brookville	Fried Chicken Joint	Caribbean Restaurant	Pizza Place	Chinese Restaurant	Fast Food Restaurant	Queens	40.660003	-73.751753
41	Cambria Heights	Caribbean Restaurant	Chinese Restaurant	Latin American Restaurant	Pizza Place	Fast Food Restaurant	Queens	40.692775	-73.735269
68	Crown Heights	Caribbean Restaurant	Fast Food Restaurant	Pizza Place	French Restaurant	Mexican Restaurant	Brooklyn	40.670829	-73.943291
77	East Flatbush	Caribbean Restaurant	Pizza Place	Fried Chicken Joint	Chinese Restaurant	Fast Food Restaurant	Brooklyn	40.641718	-73.936103
83	Eastchester	Caribbean Restaurant	Pizza Place	Fast Food Restaurant	Asian Restaurant	Chinese Restaurant	Bronx	40.887556	-73.827806

Following are the results of the Cluster – 0 analysis:

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

```
Caribbean Restaurant    18
Pizza Place              1
American Restaurant      1
Fried Chicken Joint      1
Name: 1st Most Common Venue, dtype: int64
---------------------------------------------
Fast Food Restaurant    7
Pizza Place             7
Chinese Restaurant      3
Caribbean Restaurant    2
Falafel Restaurant      1
Fried Chicken Joint     1
Name: 2nd Most Common Venue, dtype: int64
---------------------------------------------
Brooklyn         9
Queens           6
Bronx            5
Staten Island    1
Name: Borough, dtype: int64
---------------------------------------------
```

‘Caribbean Restaurant’ holds a massive accountability for this cluster with 18 occurrences in ‘1st Most Common Venue’ across different neighborhoods followed by ‘Fast Food Restaurant’ and ‘Pizza Place’ with 7 occurrences in ‘2nd Most Common Venue’. Also, ‘Caribbean Restaurant’ occurs 2 times in ‘2nd Most Common Venue’. To add on, it is inquisitive to know that majority of these neighborhoods are in ‘Brooklyn’ borough of New York City.

So, Cluster – 0 is a ‘Caribbean Restaurant’ dominant cluster.

Cluster – 1

```
cluster_1 = nyc_merged.loc[nyc_merged['Cluster Labels'] == 1, nyc_merged.columns[1:12]]
cluster_1.head(5)
```

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	Borough	Latitude	Longitude
5	Arverne	Chinese Restaurant	Pizza Place	American Restaurant	Asian Restaurant	Thai Restaurant	Queens	40.589144	-73.791992
9	Bath Beach	Chinese Restaurant	Fast Food Restaurant	Cantonese Restaurant	Sushi Restaurant	Vietnamese Restaurant	Brooklyn	40.599519	-73.998752
13	Baychester	Pizza Place	Chinese Restaurant	Fast Food Restaurant	Italian Restaurant	American Restaurant	Bronx	40.866858	-73.835798
15	Bayswater	Chinese Restaurant	Pizza Place	Fried Chicken Joint	Caribbean Restaurant	American Restaurant	Queens	40.611322	-73.765968
23	Bensonhurst	Chinese Restaurant	Pizza Place	Cantonese Restaurant	Japanese Restaurant	Italian Restaurant	Brooklyn	40.611009	-73.995180

Following are the results of the Cluster – 1 analysis:

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

```
Chinese Restaurant      18
Pizza Place              8
Caribbean Restaurant     2
Indian Restaurant        2
Seafood Restaurant       1
Name: 1st Most Common Venue, dtype: int64
---------------------------------------------
Pizza Place             10
Chinese Restaurant       8
Caribbean Restaurant     5
Fast Food Restaurant     3
American Restaurant      1
Fried Chicken Joint      1
Mexican Restaurant       1
Italian Restaurant       1
Cantonese Restaurant     1
Name: 2nd Most Common Venue, dtype: int64
---------------------------------------------
Queens           18
Brooklyn          9
Bronx             3
Staten Island     1
Name: Borough, dtype: int64
---------------------------------------------
```

'Chinese Restaurant' holds a massive accountability for this cluster with 18 occurrences followed by 'Pizza Place' with 8 occurrences in '1st Most Common Venue' across different neighborhoods. Also, 'Pizza Place' occurs whopping 10 times followed by 'Chinese Restaurant' occurrences of 8 times in '2nd Most Common Venue'. To add on, it is inquisitive to know that majority of these neighborhoods are in 'Queens' borough of New York City.

So, Cluster – 1 is a combination of 'Chinese Restaurant' and 'Pizza Place'.

Cluster – 2

```
cluster_2 = nyc_merged.loc[nyc_merged['Cluster Labels'] == 2, nyc_merged.columns[1:12]]
cluster_2.head(5)
```

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	Borough	Latitude	Longitude
4	Arrochar	Italian Restaurant	Pizza Place	Steakhouse	Middle Eastern Restaurant	Chinese Restaurant	Staten Island	40.596313	-74.067124
7	Astoria Heights	Italian Restaurant	Chinese Restaurant	Greek Restaurant	Pizza Place	Sushi Restaurant	Queens	40.770317	-73.894680
12	Bay Terrace	Italian Restaurant	Asian Restaurant	Pizza Place	Chinese Restaurant	American Restaurant	Queens	40.782843	-73.776802
12	Bay Terrace	Italian Restaurant	Asian Restaurant	Pizza Place	Chinese Restaurant	American Restaurant	Staten Island	40.553988	-74.139166
18	Beechhurst	Italian Restaurant	Pizza Place	Chinese Restaurant	Japanese Restaurant	Vietnamese Restaurant	Queens	40.792781	-73.804365

Following are the results of the Cluster – 2 analysis:

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

```
Italian Restaurant      27
Pizza Place             16
Fast Food Restaurant     2
Falafel Restaurant       1
Name: 1st Most Common Venue, dtype: int64
---------------------------------------------
Italian Restaurant      16
Pizza Place             15
Chinese Restaurant       5
Asian Restaurant         4
Mexican Restaurant       2
Fast Food Restaurant     2
American Restaurant      2
Name: 2nd Most Common Venue, dtype: int64
---------------------------------------------
Staten Island    22
Queens           10
Bronx             8
Brooklyn          6
Name: Borough, dtype: int64
---------------------------------------------
```

'Italian Restaurant' holds a massive accountability for this cluster with 27 occurrences followed by 'Pizza Place' with 16 occurrences in '1st Most Common Venue' across different neighborhoods. Also, 'Italian Restaurant' occurs whopping 16 times followed by 'Pizza Place' occurrences of 15 times in '2nd Most Common Venue'. To add on, it is inquisitive to know that majority of these neighborhoods are in 'Staten Island' and 'Queens' borough of New York City.

So, Cluster – 2 is a combination of 'Italian Restaurant' and 'Pizza Place'. Pizza is an Italian cuisine, hence cluster – 2 can be termed as 'Italian Restaurant' dominant cluster.

Cluster – 3

```
cluster_3 = nyc_merged.loc[nyc_merged['Cluster Labels'] == 3, nyc_merged.columns[1:12]]
cluster_3.head(5)
```

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	Borough	Latitude	Longitude
6	Astoria	Greek Restaurant	Pizza Place	Fast Food Restaurant	Vietnamese Restaurant	American Restaurant	Queens	40.768509	-73.915654
8	Auburndale	Korean Restaurant	Greek Restaurant	Pizza Place	Italian Restaurant	American Restaurant	Queens	40.761730	-73.791762
10	Battery Park City	Seafood Restaurant	Italian Restaurant	Pizza Place	Fast Food Restaurant	New American Restaurant	Manhattan	40.711932	-74.016869
11	Bay Ridge	Fast Food Restaurant	Pizza Place	Italian Restaurant	Middle Eastern Restaurant	Japanese Restaurant	Brooklyn	40.625801	-74.030621
14	Bayside	American Restaurant	Korean Restaurant	Italian Restaurant	Greek Restaurant	Fast Food Restaurant	Queens	40.766041	-73.774274

Following are the results of the Cluster – 3 analysis:

```python
for col in required_column:
    print(cluster_3[col].value_counts(ascending = False))
    print("--------------------------------------------")
```

```
Pizza Place                    21
Italian Restaurant             18
American Restaurant            14
Korean Restaurant               7
Seafood Restaurant              5
Fast Food Restaurant            5
New American Restaurant         3
Thai Restaurant                 3
Mexican Restaurant              2
Greek Restaurant                2
Vegetarian / Vegan Restaurant   2
Sushi Restaurant                2
Middle Eastern Restaurant       1
Vietnamese Restaurant           1
Indian Restaurant               1
Ramen Restaurant                1
Eastern European Restaurant     1
Name: 1st Most Common Venue, dtype: int64
--------------------------------------------

Italian Restaurant             17
Pizza Place                    12
American Restaurant            11
Fast Food Restaurant            7
French Restaurant               7
Mexican Restaurant              6
BBQ Joint                       4
Vietnamese Restaurant           4
Turkish Restaurant              2
Middle Eastern Restaurant       2
Korean Restaurant               2
Russian Restaurant              1
Sushi Restaurant                1
Ramen Restaurant                1
Noodle House                    1
Indian Restaurant               1
Japanese Restaurant             1
Latin American Restaurant       1
Sri Lankan Restaurant           1
Shanghai Restaurant             1
Seafood Restaurant              1
Asian Restaurant                1
Thai Restaurant                 1
Caribbean Restaurant            1
Greek Restaurant                1
Vegetarian / Vegan Restaurant   1
Name: 2nd Most Common Venue, dtype: int64
--------------------------------------------
Manhattan        28
Brooklyn         25
Queens           22
Staten Island    12
Bronx             2
Name: Borough, dtype: int64
--------------------------------------------
```

This is the biggest cluster of all, which means that majority of the neighborhoods are clustered in it. Again, as seen in Cluster – 2, 'Pizza Place' and 'Italian Restaurant' holds the top 2 places with respect to the count of occurrences of these categories in '1st Most Common Venue' as well as in '2nd Most Common Venue', But, it is interesting to know that the 3rd place is dominantly held by 'American Restaurant' category, which is why this cluster is segregated from Cluster – 2.

To add on, it is inquisitive to know that neighborhoods in this cluster is spread equally across 'Manhattan', 'Brooklyn' and 'Queens' with substantial number of neighborhoods in 'Staten Island'.

So, Cluster – 3 is a combination of ‘Italian Restaurant’, ‘Pizza Place’ and ‘American Restaurant’, with ‘American Restaurant’ showing the dominance to segment this cluster from Cluster – 2

Cluster – 4

```
cluster_4 = nyc_merged.loc[nyc_merged['Cluster Labels'] == 4, nyc_merged.columns[1:12]]
cluster_4.head(5)
```

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	Borough	Latitude	Longitude
1	Annadale	Pizza Place	American Restaurant	Sushi Restaurant	Italian Restaurant	Japanese Restaurant	Staten Island	40.538114	-74.178549
2	Arden Heights	Pizza Place	American Restaurant	Italian Restaurant	Mexican Restaurant	Chinese Restaurant	Staten Island	40.549286	-74.185887
16	Bedford Park	Pizza Place	Chinese Restaurant	Fast Food Restaurant	Salvadoran Restaurant	Fried Chicken Joint	Bronx	40.870185	-73.885512
19	Bellaire	Pizza Place	Chinese Restaurant	Fast Food Restaurant	Italian Restaurant	American Restaurant	Queens	40.733014	-73.738892
21	Bellerose	Pizza Place	Indian Restaurant	American Restaurant	Halal Restaurant	Seafood Restaurant	Queens	40.728573	-73.720128

Following are the results of the Cluster – 4 analysis:

```
for col in required_column:
    print(cluster_4[col].value_counts(ascending = False))
    print("---------------------------------------------")
```

```
Pizza Place    52
Taco Place      1
Name: 1st Most Common Venue, dtype: int64
---------------------------------------------
Chinese Restaurant      12
American Restaurant      9
Japanese Restaurant      5
Fast Food Restaurant     5
Italian Restaurant       4
Mexican Restaurant       4
Sushi Restaurant         3
Asian Restaurant         3
Taco Place               2
BBQ Joint                2
Spanish Restaurant       1
Indian Restaurant        1
Pizza Place              1
Caribbean Restaurant     1
Name: 2nd Most Common Venue, dtype: int64
---------------------------------------------
Staten Island    20
Queens           14
Bronx            11
Brooklyn          8
Name: Borough, dtype: int64
---------------------------------------------
```

In this cluster, ‘Pizza Place’ has taken over every other category with shooting 52 occurrences in ‘1st Most Common Venue’ across different neighborhoods followed by ‘Chinese Restaurant’ with 12 occurrences in ‘2nd Most Common Venue’. To add on, it is inquisitive to know that majority of these neighborhoods are spread approximately equally across ‘Staten Island’, ‘Queens’ and ‘Bronx’ borough of New York City.

So, Cluster – 4 can be termed as ‘Pizza Place’ dominant cluster.

Cluster – 5

```
cluster_5 = nyc_merged.loc[nyc_merged['Cluster Labels'] == 5, nyc_merged.columns[1:12]]
cluster_5.head(5)
```

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	Borough	Latitude	Longitude
152	Lighthouse Hill	Italian Restaurant	Vietnamese Restaurant	Halal Restaurant	English Restaurant	Ethiopian Restaurant	Staten Island	40.576506	-74.137927

Following are the results of the Cluster – 5 analysis:

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

```
Italian Restaurant    1
Name: 1st Most Common Venue, dtype: int64
---------------------------------------------
Vietnamese Restaurant    1
Name: 2nd Most Common Venue, dtype: int64
---------------------------------------------
Staten Island    1
Name: Borough, dtype: int64
---------------------------------------------
```

It is clear, that only one neighborhood 'Lighthouse Hill' is curated under this cluster. This segmentation can be understood from the fact that 'Lighthouse Hill' is a tourist attraction for its heritage and is situated at the southernmost of the chain of hills that radiate from the northeast corner of Staten Island. This neighborhood has diverse cuisine in its top 5 most common venues list and hence a separate cluster.

So, Cluster – 5 can be termed as exceptional as of now.

Cluster – 6

```
cluster_6 = nyc_merged.loc[nyc_merged['Cluster Labels'] == 6, nyc_merged.columns[1:12]]
cluster_6.head(5)
```

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	Borough	Latitude	Longitude
0	Allerton	Mexican Restaurant	Fried Chicken Joint	Pizza Place	Chinese Restaurant	Fast Food Restaurant	Bronx	40.865788	-73.859319
3	Arlington	Pizza Place	Fast Food Restaurant	American Restaurant	Peruvian Restaurant	Spanish Restaurant	Staten Island	40.635325	-74.165104
17	Bedford Stuyvesant	Pizza Place	Fast Food Restaurant	Caribbean Restaurant	French Restaurant	Ramen Restaurant	Brooklyn	40.687232	-73.941785
30	Briarwood	Fast Food Restaurant	Pizza Place	Fried Chicken Joint	Chinese Restaurant	Japanese Restaurant	Queens	40.710935	-73.811748
33	Broadway Junction	Fast Food Restaurant	Fried Chicken Joint	Mexican Restaurant	Chinese Restaurant	Pizza Place	Brooklyn	40.677861	-73.903317

Following are the results of the Cluster – 6 analysis:

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

```
Fast Food Restaurant    28
Pizza Place             14
Mexican Restaurant       5
Chinese Restaurant       2
Fried Chicken Joint      1
Name: 1st Most Common Venue, dtype: int64
---------------------------------------------
Fast Food Restaurant              16
Pizza Place                       10
Chinese Restaurant                 8
Fried Chicken Joint                5
Mexican Restaurant                 4
Latin American Restaurant          2
Southern / Soul Food Restaurant    1
Indian Restaurant                  1
Filipino Restaurant                1
Caribbean Restaurant               1
Italian Restaurant                 1
Name: 2nd Most Common Venue, dtype: int64
---------------------------------------------
Bronx            22
Queens           10
Brooklyn          9
Manhattan         5
Staten Island     4
Name: Borough, dtype: int64
---------------------------------------------
```

'Fast Food Restaurant' holds a massive accountability for this cluster with 28 occurrences followed by 'Pizza Place' with 14 occurrences in '1st Most Common Venue' across different neighborhoods. Also, the same proportionate distribution can be seen in '2nd Most Common Venue' occurrence counts.

To add on, it is inquisitive to know that majority of these neighborhoods are in 'Bronx' and 'Queens' borough of New York City.

It is known that, although pizza is an Italian cuisine, it is also a fast food. So, Cluster – 6 can be termed as 'Fast Food Restaurant' dominant cluster.

Cluster – 7

```
cluster_7 = nyc_merged.loc[nyc_merged['Cluster Labels'] == 7, nyc_merged.columns[1:12]]
cluster_7.head(5)
```

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	Borough	Latitude	Longitude
29	Breezy Point	American Restaurant	Pizza Place	Vietnamese Restaurant	Halal Restaurant	Ethiopian Restaurant	Queens	40.557401	-73.925512
35	Brooklyn Heights	American Restaurant	Pizza Place	Mexican Restaurant	Seafood Restaurant	Mediterranean Restaurant	Brooklyn	40.695864	-73.993782
43	Carnegie Hill	American Restaurant	Vietnamese Restaurant	BBQ Joint	Pizza Place	Fast Food Restaurant	Manhattan	40.782683	-73.953256
55	Clason Point	American Restaurant	Seafood Restaurant	South American Restaurant	Greek Restaurant	English Restaurant	Bronx	40.806551	-73.854144
74	Dumbo	American Restaurant	Pizza Place	Italian Restaurant	Mexican Restaurant	Seafood Restaurant	Brooklyn	40.703176	-73.988753

Following are the results of the Cluster – 7 analysis:

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

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

'American Restaurant' hold a massive accountability for this cluster with 14 occurrences in '1st Most Common Venue' across different neighborhoods. Also, there is a mix of cuisines in the '2nd Most Common Venue'. To add on, it is inquisitive to know that majority of these neighborhoods are in 'Manhattan' and 'Brooklyn' borough of New York City.

So, Cluster – 7 can be termed as 'American Restaurant' dominant cluster.

Discussion

To understand the clusters, three analysis were done, namely:

1. Count of 'Borough'
2. Count of '1st Most Common Venue'
3. Count of '2nd Most Common Venue'

The above information speaks a lot about the ground reality of clustering based on the similarity metrics between the neighborhoods.

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

	Count of Occurrences within the Cluster		
Cluster	1stMostCommonVenue	2ndMostCommonVenue	Borough
0	Caribbean Restaurant	Fast Food Restaurant, Pizza Place	Brooklyn
1	Chinese Restaurant, Pizza Place	Pizza Place, Chinese Restaurant	Queens
2	Italian Restaurant, Pizza Place	Italian Restaurant, Pizza Place	Staten Island, Queens

3	Pizza Place, Italian Restaurant, American Restaurant	Pizza Place, Italian Restaurant, American Restaurant	Manhattan, Brooklyn, Queens
4	Pizza Place	Chinese Restaurant	Staten Island, Queens, Bronx
5	Italian Restaurant	Vietnamese Restaurant	Lighthouse Hill
6	Fast Food Restaurant, Pizza Place	Fast Food Restaurant, Pizza Place	Bronx, Queens
7	American Restaurant	Pizza Place	Manhattan, Brooklyn

Pizza, who does not like it. And it is obvious from the analysis that Pizza Place is the most common venue across all the clusters or neighborhoods. So, 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 – Caribbean
- Cluster 1 – Chinese
- Cluster 2 – Italian
- Cluster 3 – Italian American
- Cluster 4 – Pizza
- Cluster 5 – Mix of Cuisines
- Cluster 6 – Fast Food
- Cluster 7 – American

Conclusion

On application of Clustering Algorithm, k-Means or others, to a multi-dimensional dataset, a very inquisitive results can be curated which helps to understand and visualize the data. The neighborhoods of New York City were very briefly segmented into eight clusters and upon analysis it was possible to rename them basis upon the categories of venues in and around that neighborhood. Along with the American cuisine, Italian and Chinese are very dominant in New York City and so is the diversity statistics.

The results of this project can be improved and made more inquisitive by using a current New York City's dataset along with API platforms which are more interested in Food Venues (like Yelp, etc.) The scope of this project can be expanded further to understand the dynamics of each neighborhood and suggest a new vendor a profitable location to open his or her food place. Also, a government authority can utilize it to examine and study their city's culture diversity better.

References

Notebook created by Alex Aklson and Polong Lin for the ‘Applied Data Science Capstone’ course on Coursera