Machine Learning

Exploring the Taste of Chicago

FEBRUARY 12

Coursera
Authored by: Utkarsh Kapoor

Introduction

Background

Chicago is the most popular 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 Chicago 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 – Chicago'. 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:

Chicago Dataset

Link: https://en.wikipedia.org/wiki/List_of_neighborhoods_in_Chicago

There are sometimes said to be more than 200 neighborhoods in Chicago, though few residents would agree on their names and boundaries. A city ordinance prescribing and mapping 178 neighborhoods is almost unknown and ignored even by municipal departments. Neighborhood names and identities have evolved over time due to real estate development and changing demographics. The City of Chicago is also divided into 77 community areas which were drawn by University of Chicago researchers in the late 1920s. Chicago's community areas are well-defined, generally contain multiple neighborhoods, and are less commonly used by city residents. More historical images of Chicago neighborhoods can be found in Explore Chicago Collections, a digital repository made available by Chicago Collections archives, libraries and other cultural institutions in the city. Here is an example of webpage table data:

Neighborhood	Community area
Albany Park	Albany Park
Altgeld Gardens	Riverdale
Andersonville	Edgewater
Archer Heights	Archer Heights
Armour Square	Armour Square

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 Chicago Dataset

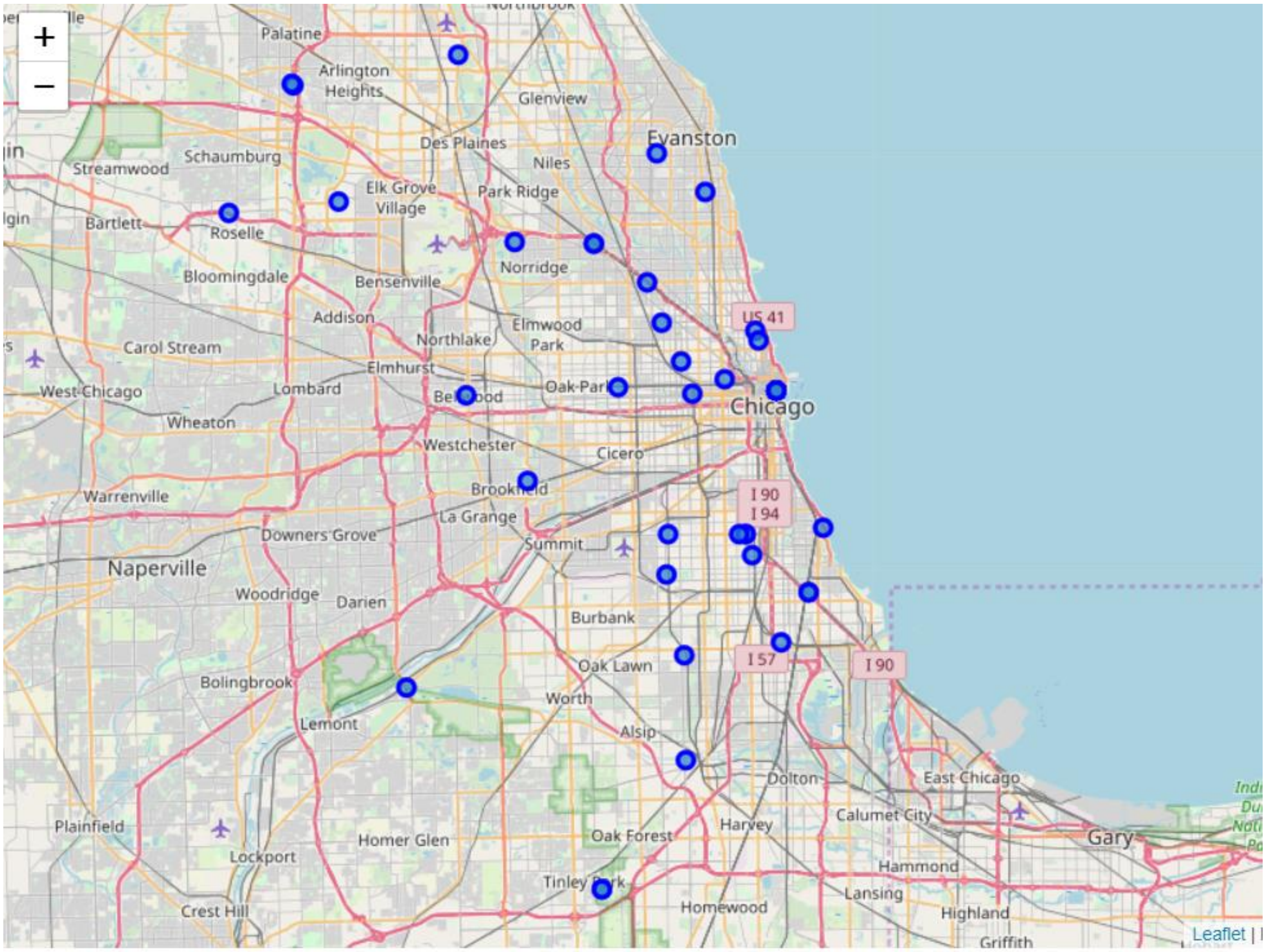
Here we will consider the original Wikipedia page for creating our data set. We will use Beautiful soup to get the table and create a pandas data frame from there.

```
res = requests.get("https://en.wikipedia.org/wiki/List_of_neighborhoods_in_Chicago")
soup = BeautifulSoup(res.content,'lxml')
table = soup.find_all('table')[0]
data = pd.read_html(str(table))
df=pd.DataFrame(data[0])

# More than one community area can exist in one neighbourhood.


df1=df.groupby("Community area").agg(lambda x:','.join(set(x)))

df1.head()
```

Now we will include the latitude and longitude for each Borough.

	Borough	Neighborhood	Latitude	Longitude
0	Albany Park	North Mayfair,Mayfair,Ravenswood Manor,Albany ...	41.717189	-87.699098
1	Archer Heights	Archer Heights	41.696598	-87.936453
2	Armour Square	Armour Square,Wentworth Gardens,Chinatown	41.892001	-87.665688
3	Ashburn	Crestline,Ashburn Estates,Parkview,Beverly Vie...	41.885310	-87.622130
4	Auburn Gresham	Auburn Gresham,Gresham	42.078163	-88.031678
5	Austin	Galewood,South Austin,North Austin,The Island	41.568075	-87.769531

Further, 'geopy' library is used to get the latitude and longitude values of Chicago City. The curated dataframe is then used to visualize by creating a map of Chicago 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:

```
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.

```
# 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.

To further understand the results of GET Request, the first neighborhood of the ‘Chicago City’ dataset is explored. The first neighborhood returned is ‘North Mayfair’ with Latitude 41.71 and Longitude - 87.69. 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={}'.format(
    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': '42c1e480f964a520c2251fe3',
    'name': 'Chi Tung Restaurant',
    'location': {'address': '9560 S Kedzie Ave',
     'crossStreet': 'at 95th St',
     'lat': 41.7193530332884,
     'lng': -87.7019967639367,
     'labeledLatLngs': [{'label': 'display',
       'lat': 41.7193530332884,
       'lng': -87.7019967639367}],
     'distance': 340,
     'postalCode': '60805',
     'cc': 'US',
     'city': 'Evergreen Park',
     'state': 'IL',
     'country': 'United States',
     'formattedAddress': ['9560 S Kedzie Ave (at 95th St)',
      'Evergreen Park, IL 60805',
      'United States']},
    'categories': [{'id': '4bf58dd8d48988d145941735',
      'name': 'Chinese Restaurant',
      'pluralName': 'Chinese Restaurants',
      'shortName': 'Chinese',
      'icon': {'prefix': 'https://ss3.4sqi.net/img/categories_v2/food/asian_',
       'suffix': '.png'},
      'primary': True}],
    'delivery': {'id': '629506',
     'url': 'https://www.grubhub.com/restaurant/chi-tung-9560-s-kedzie-ave-evergreen-park/629506?affiliate=1131&utm_source=foursquare-affiliate-network&utm_medium=affiliat
  e&utm_campaign=1131&utm_content=629506',
     'provider': {'name': 'grubhub',
      'icon': {'prefix': 'https://fastly.4sqi.net/img/general/cap/',
       'sizes': [40, 50],
       'name': '/delivery_provider_grubhub_20180129.png'}}},
```


The category name of the venue 'Chi Tung Restaurant' is 'Food' which is returned here.

As, the aim is to segment the neighborhoods of Chicago City with respect to the ‘Food’ in its vicinity, it is further required to fetch this data from all the neighborhoods' venues.

To overcome the redundancy of the process followed above, a function ‘getNearbyFood’ is created. This functions loop through all the neighborhoods of Chicago 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 data frame being returned by the function. It is inquisitive to know that Foursquare API returns all the sub-categories, if a top-level category is specified in the GET Request.

```python
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={}&limit={}'.format(
            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:
        chi_venues = pickle.load(f)
    print("---Dataframe Existed and Deserialized---")
except:
    chi_venues = getNearbyFood(names=neighborhoods['Neighborhood'],
                                    latitudes=neighborhoods['Latitude'],
                                    longitudes=neighborhoods['Longitude']
                                   )
    with open('chi_food_venues.pkl', 'wb') as f:
        pickle.dump(chi_venues, f)
    print("---Dataframe Created and Serialized---")
```

The returned 'dataframe' is as follows:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	North Mayfair,Mayfair,Ravenswood Manor,Albany ...	41.717189	-87.699098	Chi Tung Restaurant	41.719353	-87.701997	Chinese Restaurant
1	North Mayfair,Mayfair,Ravenswood Manor,Albany ...	41.717189	-87.699098	Brown's Chicken & Pasta	41.720925	-87.707587	Fried Chicken Joint
2	North Mayfair,Mayfair,Ravenswood Manor,Albany ...	41.717189	-87.699098	Wu's House	41.720964	-87.696523	Japanese Restaurant
3	North Mayfair,Mayfair,Ravenswood Manor,Albany ...	41.717189	-87.699098	Wolf's Bakery	41.720565	-87.703548	Bakery
4	North Mayfair,Mayfair,Ravenswood Manor,Albany ...	41.717189	-87.699098	La Cocina Jalisciense	41.720873	-87.702548	Mexican Restaurant

As of now, two python 'dataframe' are created:

1) 'neighborhoods' which contains the Borough, Neighborhood, Latitude and Longitude details of the Chicago City's neighborhood, and

2) 'chi_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 'chi_venues' has all the required information. The size of this dataframe is determined, and it is found that there are total 3556 venues.

```
print(chi_venues.shape)
chi_venues.head()
```

```
(3556, 7)
```

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Ravenswood Manor,Albany Park,North Mayfair,May...	41.717189	-87.699098	Chi Tung Restaurant	41.719353	-87.701997	Chinese Restaurant
1	Ravenswood Manor,Albany Park,North Mayfair,May...	41.717189	-87.699098	Wolf's Bakery	41.720565	-87.703548	Bakery
2	Ravenswood Manor,Albany Park,North Mayfair,May...	41.717189	-87.699098	Brown's Chicken & Pasta	41.720925	-87.707587	Fried Chicken Joint
3	Ravenswood Manor,Albany Park,North Mayfair,May...	41.717189	-87.699098	La Cocina Jalisciense	41.720873	-87.702548	Mexican Restaurant
4	Ravenswood Manor,Albany Park,North Mayfair,May...	41.717189	-87.699098	Wu's House	41.720964	-87.696523	Japanese Restaurant

Now, it is important to find out that how many unique categories can be curated from all the returned venues. There are 109 such categories, with most occurring venues as follows:

```
There are 109 uniques categories.

: Venue Category
Coffee Shop                        575
Food Court                         196
Pizza Place                        181
Italian Restaurant                 166
American Restaurant                164
New American Restaurant            151
Fast Food Restaurant               138
Mexican Restaurant                 133
Café                               133
Mediterranean Restaurant           100
Pub                                 97
Bakery                              93
Donut Shop                          81
Breakfast Spot                      76
BBQ Joint                           70
Burger Joint                        66
Hot Dog Joint                       64
Seafood Restaurant                  61
Sushi Restaurant                    56
Asian Restaurant                    55
Gastropub                           53
Sandwich Place                      51
Noodle House                        49
Shopping Mall                       48
Sports Bar                          48
Hotel                               48
Chinese Restaurant                  47
Salad Place                         47
Tiki Bar                            47
Gourmet Shop                        47
Fried Chicken Joint                 43
Food                                43
Restaurant                          36
Ice Cream Shop                      30
Deli / Bodega                       22
Caribbean Restaurant                16
Southern / Soul Food Restaurant     16
Diner                               14
```

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 = chi_venues['Venue Category'].unique().tolist()
print(', '.join(str(x) for x in unique_categories))
```

```
Chinese Restaurant, Fried Chicken Joint, Japanese Restaurant, Bakery, Mexican Restaurant, Latin American R
estaurant, Thai Restaurant, Fast Food Restaurant, Donut Shop, Wings Joint, Pizza Place, Taco Place, Americ
an Restaurant, Indian Restaurant, Italian Restaurant, BBQ Joint, Breakfast Spot, Coffee Shop, Ice Cream Sh
op, Food, Cafeteria, Deli / Bodega, Bagel Shop, Sandwich Place, Sushi Restaurant, Event Space, Café, Greek
Restaurant, Portuguese Restaurant, Restaurant, Pie Shop, Filipino Restaurant, Bar, Concert Hall, Brewery,
German Restaurant, Burger Joint, Gastropub, Music Venue, New American Restaurant, Modern European Restaura
nt, Whisky Bar, Food Court, Hotel, Pub, Asian Restaurant, Gourmet Shop, Hot Dog Joint, Shopping Mall, Tiki
Bar, Salad Place, Mediterranean Restaurant, Noodle House, Sports Bar, Seafood Restaurant, Food Truck, Conv
enience Store, Vietnamese Restaurant, Eastern European Restaurant, Spanish Restaurant, African Restaurant,
Diner, Southern / Soul Food Restaurant, Brazilian Restaurant, Caribbean Restaurant, Buffet, Kosher Restaur
ant, Cajun / Creole Restaurant, Professional & Other Places, French Restaurant, Performing Arts Venue, Dim
Sum Restaurant, Bistro, Tapas Restaurant, Israeli Restaurant, English Restaurant, Cupcake Shop, Hawaiian R
estaurant, Korean Restaurant, Beer Garden, Fondue Restaurant, Szechuan Restaurant, Dessert Shop, Colombian
Restaurant, Burrito Place, Liquor Store, Snack Place, Moroccan Restaurant, Cuban Restaurant, Grocery Stor
e, Peruvian Restaurant, Halal Restaurant, Middle Eastern Restaurant, Polish Restaurant, Juice Bar, Cantone
se Restaurant, Dumpling Restaurant, Souvlaki Shop, Vegetarian / Vegan Restaurant, Cocktail Bar, Ramen Rest
aurant, Poke Place, Movie Theater, Comfort Food Restaurant, Ukrainian Restaurant, Falafel Restaurant, Czec
h Restaurant, Irish Pub, Steakhouse
```

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':

```
# fetch all the required food categories
food_categories =  list(set(unique_categories) - set(general_categories))
print(', '.join(str(x) for x in food_categories))
```

```
Seafood Restaurant, Israeli Restaurant, Performing Arts Venue, Dim Sum Restaurant, Tiki Bar, Liquor Store,
Cantonese Restaurant, Szechuan Restaurant, Indian Restaurant, Professional & Other Places, Noodle House, C
hinese Restaurant, Mexican Restaurant, Pizza Place, BBQ Joint, Southern / Soul Food Restaurant, Tapas Rest
aurant, Mediterranean Restaurant, Moroccan Restaurant, Ukrainian Restaurant, Souvlaki Shop, Thai Restauran
t, Burrito Place, Asian Restaurant, German Restaurant, Italian Restaurant, Kosher Restaurant, Eastern Euro
pean Restaurant, Cuban Restaurant, Halal Restaurant, Fondue Restaurant, Brazilian Restaurant, Czech Restau
rant, African Restaurant, Greek Restaurant, Cajun / Creole Restaurant, Vegetarian / Vegan Restaurant, Span
ish Restaurant, Hawaiian Restaurant, English Restaurant, Concert Hall, Peruvian Restaurant, Ramen Restaura
nt, Taco Place, Falafel Restaurant, Fried Chicken Joint, Latin American Restaurant, New American Restauran
t, Sushi Restaurant, Caribbean Restaurant, Movie Theater, Middle Eastern Restaurant, Dumpling Restaurant,
Fast Food Restaurant, Whisky Bar, Japanese Restaurant, Portuguese Restaurant, Modern European Restaurant,
French Restaurant, Steakhouse, American Restaurant, Brewery, Shopping Mall, Filipino Restaurant, Vietnames
e Restaurant, Korean Restaurant, Colombian Restaurant, Polish Restaurant
```

```
# manually create a list of generalized categories
general_categories = ['Dessert Shop','Food','Ice Cream Shop','Donut Shop','Bakery','Sandwich Place','Comfo
rt Food Restaurant',
                      'Deli / Bodega','Food Truck','Bagel Shop','Burger Joint','Restaurant','Frozen Yogurt S
hop','Coffee Shop',
                      'Diner','Wings Joint','Café','Juice Bar','Breakfast Spot','Grocery Store','Bar','Cupca
ke 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 Sho
p',
                      'Snack Place','Sports Bar','Lounge','Theme Restaurant','Buffet','Bubble Tea Shop','Bui
lding',
                      'Irish Pub','College Cafeteria','Tea Room','Supermarket','Hotpot Restaurant','Gastropu
b','Beer Garden',
                      'Fish Market','Beer Bar','Clothing Store','Music Venue','Bistro','Salad Place','Wine B
ar','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','Cr
eperie',
                      'Candy Store','Event Space','Skating Rink','Miscellaneous Shop','Gas Station','Organic
Grocery',
                      'Pastry Shop','Club House','Flea Market','Hotel','Furniture / Home Store','Bookstor
e','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 Ba
r','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))
print(', '.join(str(x) for x in food_categories))
```

```
Seafood Restaurant, Israeli Restaurant, Performing Arts Venue, Dim Sum Restaurant, Tiki Bar, Liquor Store,
Cantonese Restaurant, Szechuan Restaurant, Indian Restaurant, Professional & Other Places, Noodle House, C
hinese Restaurant, Mexican Restaurant, Pizza Place, BBQ Joint, Southern / Soul Food Restaurant, Tapas Rest
aurant, Mediterranean Restaurant, Moroccan Restaurant, Ukrainian Restaurant, Souvlaki Shop, Thai Restauran
t, Burrito Place, Asian Restaurant, German Restaurant, Italian Restaurant, Kosher Restaurant, Eastern Euro
pean Restaurant, Cuban Restaurant, Halal Restaurant, Fondue Restaurant, Brazilian Restaurant, Czech Restau
rant, African Restaurant, Greek Restaurant, Cajun / Creole Restaurant, Vegetarian / Vegan Restaurant, Span
ish Restaurant, Hawaiian Restaurant, English Restaurant, Concert Hall, Peruvian Restaurant, Ramen Restaura
nt, Taco Place, Falafel Restaurant, Fried Chicken Joint, Latin American Restaurant, New American Restauran
t, Sushi Restaurant, Caribbean Restaurant, Movie Theater, Middle Eastern Restaurant, Dumpling Restaurant,
Fast Food Restaurant, Whisky Bar, Japanese Restaurant, Portuguese Restaurant, Modern European Restaurant,
French Restaurant, Steakhouse, American Restaurant, Brewery, Shopping Mall, Filipino Restaurant, Vietnames
e Restaurant, Korean Restaurant, Colombian Restaurant, Polish 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:

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

	index	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	0	North Mayfair,Mayfair,Ravenswood Manor,Albany ...	41.717189	-87.699098	Chi Tung Restaurant	41.719353	-87.701997	Chinese Restaurant
1	1	North Mayfair,Mayfair,Ravenswood Manor,Albany ...	41.717189	-87.699098	Brown's Chicken & Pasta	41.720925	-87.707587	Fried Chicken Joint
2	2	North Mayfair,Mayfair,Ravenswood Manor,Albany ...	41.717189	-87.699098	Wu's House	41.720964	-87.696523	Japanese Restaurant
3	4	North Mayfair,Mayfair,Ravenswood Manor,Albany ...	41.717189	-87.699098	La Cocina Jalisciense	41.720873	-87.702548	Mexican Restaurant
4	5	North Mayfair,Mayfair,Ravenswood Manor,Albany ...	41.717189	-87.699098	Unidad	41.720398	-87.706084	Latin American Restaurant

Again, the number of unique categories is examined, and it is found that there are only 68 of them, as compared to 108 earlier. That means, almost 40% 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
chi_onehot = pd.get_dummies(chi_venues[['Venue Category']], prefix="", prefix_sep="")
chi_onehot.head()
```

	African Restaurant	American Restaurant	Asian Restaurant	BBQ Joint	Brazilian Restaurant	Brewery	Burrito Place	Cajun / Creole Restaurant	Cantonese Restaurant	Caribbean Restaurant	Chinese Restauran
0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0
4	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 = chi_onehot['Neighborhood']

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

chi_onehot.head()
```

	Neighborhood	African Restaurant	American Restaurant	Asian Restaurant	BBQ Joint	Brazilian Restaurant	Brewery	Burrito Place	Cajun / Creole Restaurant	Cantor Restau
0	North Mayfair,Mayfair,Ravenswood Manor,Albany ...	0	0	0	0	0	0	0	0	0
1	North Mayfair,Mayfair,Ravenswood Manor,Albany ...	0	0	0	0	0	0	0	0	0
2	North Mayfair,Mayfair,Ravenswood Manor,Albany ...	0	0	0	0	0	0	0	0	0
3	North Mayfair,Mayfair,Ravenswood Manor,Albany ...	0	0	0	0	0	0	0	0	0
4	North Mayfair,Mayfair,Ravenswood Manor,Albany ...	0	0	0	0	0	0	0	0	0

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

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

	African Restaurant	American Restaurant	Asian Restaurant	BBQ Joint	Brazilian Restaurant	Brewery	Burrito Place	Cajun / Creole Restaurant	Cantonese Restaurant	Cari Res
Neighborhood										
Armour Square,Wentworth Gardens,Chinatown	0	2	0	0	0	1	0	0	0	0
Auburn Gresham,Gresham	0	3	0	0	0	0	0	0	0	0
Avalon Park,Stony Island Park,Marynook	0	2	1	1	0	0	0	0	0	0
Bridgeport	0	2	1	1	0	0	0	0	0	0
Brighton Park	0	2	1	1	0	0	0	0	0	0

The top 10 ‘Venue Categories’ can also be found by counting their occurrences. This analysis is depicted below which shows that ‘Mexican Restaurant’, ‘Fast food Restaurant’,

Fried Chicken Restaurant’, ‘Pizza Restaurant’, and ‘Fast Food Restaurant’ are among the top 5.

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

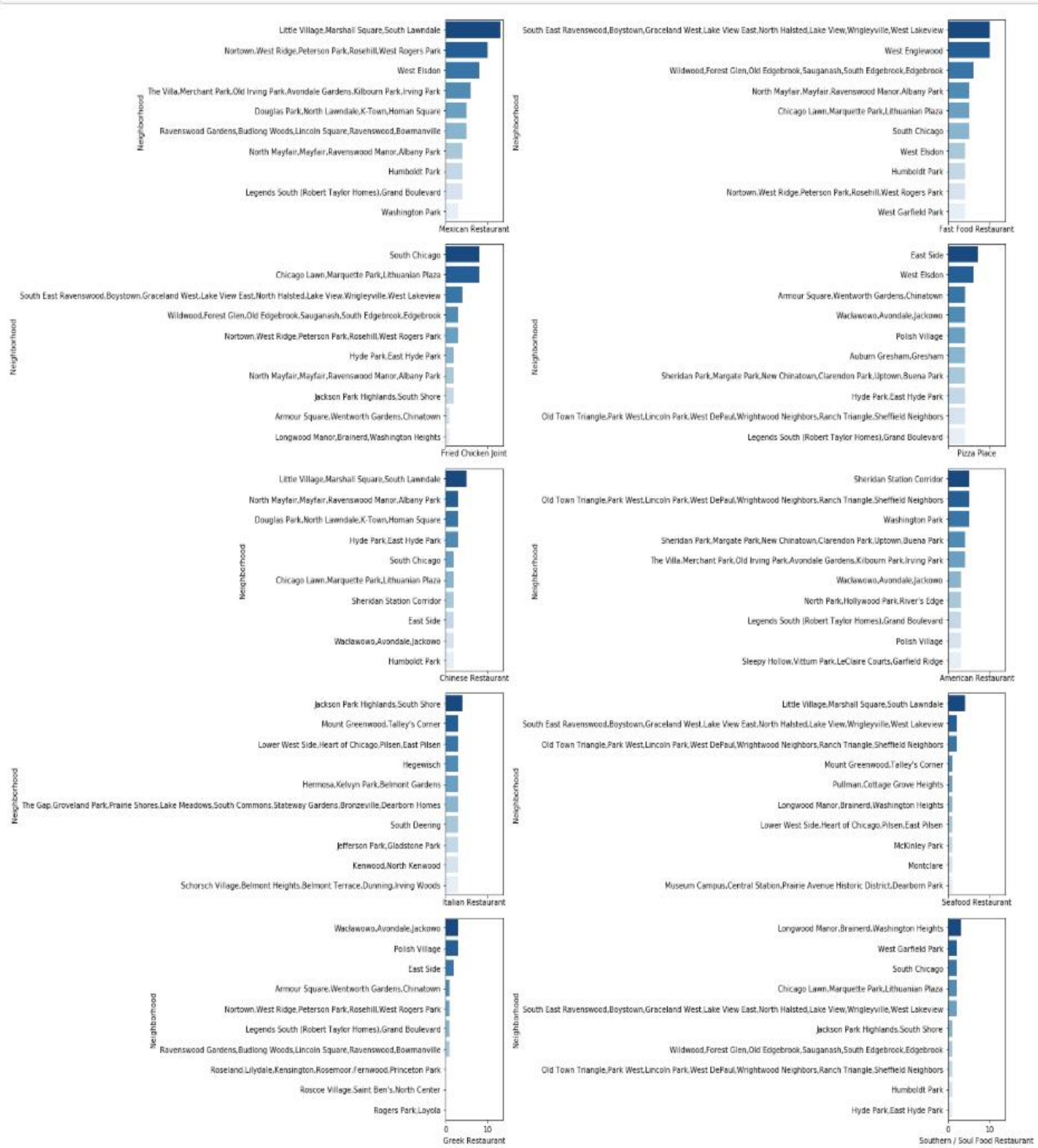
```
venue_top10 = venue_counts_described.sort_values('max', ascending=False)[0:10]
venue_top10
```

	count	mean	std	min	25%	50%	75%	max
Mexican Restaurant	81.0	1.641975	2.087514	0.0	1.0	1.0	1.0	13.0
Fast Food Restaurant	81.0	1.703704	1.791957	0.0	1.0	1.0	2.0	10.0
Fried Chicken Joint	81.0	0.530864	1.423784	0.0	0.0	0.0	0.0	8.0
Pizza Place	81.0	2.234568	1.075542	0.0	2.0	2.0	2.0	7.0
Chinese Restaurant	81.0	0.580247	0.985700	0.0	0.0	0.0	1.0	5.0
American Restaurant	81.0	2.024691	0.987108	0.0	2.0	2.0	2.0	5.0
Italian Restaurant	81.0	2.049383	1.312452	0.0	1.0	3.0	3.0	4.0
Seafood Restaurant	81.0	0.753086	0.623114	0.0	0.0	1.0	1.0	4.0
Greek Restaurant	81.0	0.148148	0.550252	0.0	0.0	0.0	0.0	3.0
Southern / Soul Food Restaurant	81.0	0.197531	0.579218	0.0	0.0	0.0	0.0	3.0

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.

```
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();
```


Little Village,Marshall Square,South Lawndale
Nortown,West Ridge,Peterson Park,Rosehill,West Rogers Park
West Elsdon
The Villa,Merchant Park,Old Irving Park,Avondale Gardens,Kilbourn Park,Irving Park
Douglas Park,North Lawndale,K-Town,Homan Square
Ravenswood Gardens,Budlong Woods,Lincoln Square,Ravenswood,Bowmanville
North Mayfair,Mayfair,Ravenswood Manor,Albany Park
Humboldt Park
Legends South (Robert Taylor Homes),Grand Boulevard
Washington Park
Neighborhood
Mexican Restaurant
South East Ravenswood,Boystown,Graceland West,Lake View East,North Halsted,Lake View,Wrigleyville,West Lakeview
West Englewood
Wildwood,Forest Glen,Old Edgebrook,Sauganash,South Edgebrook,Edgebrook
North Mayfair,Mayfair,Ravenswood Manor,Albany Park
Chicago Lawn,Marquette Park,Lithuanian Plaza
South Chicago
West Elsdon
Humboldt Park
Nortown,West Ridge,Peterson Park,Rosehill,West Rogers Park
West Garfield Park
Neighborhood
Fast Food Restaurant
South Chicago
Chicago Lawn,Marquette Park,Lithuanian Plaza
South East Ravenswood,Boystown,Graceland West,Lake View East,North Halsted,Lake View,Wrigleyville,West Lakeview
Wildwood,Forest Glen,Old Edgebrook,Sauganash,South Edgebrook,Edgebrook
Nortown,West Ridge,Peterson Park,Rosehill,West Rogers Park
Hyde Park,East Hyde Park
North Mayfair,Mayfair,Ravenswood Manor,Albany Park
Jackson Park Highlands,South Shore
Armour Square,Wentworth Gardens,Chinatown
Longwood Manor,Brainerd,Washington Heights
Neighborhood
Fried Chicken Joint
East Side
West Elsdon
Armour Square,Wentworth Gardens,Chinatown
Waclawowo,Avondale,Jackowo
Polish Village
Auburn Gresham,Gresham
Sheridan Park,Margate Park,New Chinatown,Clarendon Park,Uptown,Buena Park
Hyde Park,East Hyde Park
Old Town Triangle,Park West,Lincoln Park,West DePaul,Wrightwood Neighbors,Ranch Triangle,Sheffield Neighbors
Legends South (Robert Taylor Homes),Grand Boulevard
Neighborhood
Pizza Place
Little Village,Marshall Square,South Lawndale
North Mayfair,Mayfair,Ravenswood Manor,Albany Park
Douglas Park,North Lawndale,K-Town,Homan Square
Hyde Park,East Hyde Park
South Chicago
Chicago Lawn,Marquette Park,Lithuanian Plaza
Sheridan Station Corridor
East Side
Waclawowo,Avondale,Jackowo
Humboldt Park
Neighborhood
Chinese Restaurant
Sheridan Station Corridor
Old Town Triangle,Park West,Lincoln Park,West DePaul,Wrightwood Neighbors,Ranch Triangle,Sheffield Neighbors
Washington Park
Sheridan Park,Margate Park,New Chinatown,Clarendon Park,Uptown,Buena Park
The Villa,Merchant Park,Old Irving Park,Avondale Gardens,Kilbourn Park,Irving Park
Waclawowo,Avondale,Jackowo
North Park,Hollywood Park,River's Edge
Legends South (Robert Taylor Homes),Grand Boulevard
Polish Village
Sleepy Hollow,Vittum Park,LeClaire Courts,Garfield Ridge
Neighborhood
American Restaurant
Jackson Park Highlands,South Shore
Mount Greenwood,Talley's Corner
Lower West Side,Heart of Chicago,Pilsen,East Pilsen
Hegewisch
Hermosa,Kelvyn Park,Belmont Gardens
The Gap,Groveland Park,Prairie Shores,Lake Meadows,South Commons,Stateway Gardens,Bronzeville,Dearborn Homes
South Deering
Jefferson Park,Gladstone Park
Kenwood,North Kenwood
Schorsch Village,Belmont Heights,Belmont Terrace,Dunning,Irving Woods
Neighborhood
Italian Restaurant
Little Village,Marshall Square,South Lawndale
South East Ravenswood,Boystown,Graceland West,Lake View East,North Halsted,Lake View,Wrigleyville,West Lakeview
Old Town Triangle,Park West,Lincoln Park,West DePaul,Wrightwood Neighbors,Ranch Triangle,Sheffield Neighbors
Mount Greenwood,Talley's Corner
Pullman,Cottage Grove Heights
Longwood Manor,Brainerd,Washington Heights
Lower West Side,Heart of Chicago,Pilsen,East Pilsen
McKinley Park
Montclare
Museum Campus,Central Station,Prairie Avenue Historic District,Dearborn Park
Neighborhood
Seafood Restaurant
Waclawowo,Avondale,Jackowo
Polish Village
East Side
Armour Square,Wentworth Gardens,Chinatown
Nortown,West Ridge,Peterson Park,Rosehill,West Rogers Park
Legends South (Robert Taylor Homes),Grand Boulevard
Ravenswood Gardens,Budlong Woods,Lincoln Square,Ravenswood,Bowmanville
Roseland,Lilydale,Kensington,Rosemoor,Fernwood,Princeton Park
Roscoe Village,Saint Ben's,North Center
Rogers Park,Loyola
Neighborhood
0
10
Greek Restaurant
Longwood Manor,Brainerd,Washington Heights
West Garfield Park
South Chicago
Chicago Lawn,Marquette Park,Lithuanian Plaza
South East Ravenswood,Boystown,Graceland West,Lake View East,North Halsted,Lake View,Wrigleyville,West Lakeview
Jackson Park Highlands,South Shore
Wildwood,Forest Glen,Old Edgebrook,Sauganash,South Edgebrook,Edgebrook
Old Town Triangle,Park West,Lincoln Park,West DePaul,Wrightwood Neighbors,Ranch Triangle,Sheffield Neighbors
Humboldt Park
Hyde Park,East Hyde Park
Neighborhood
0
10
Southern / Soul Food Restaurant

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

```
chi_grouped = chi_onehot.groupby('Neighborhood').mean().reset_index()
chi_grouped.head()
```

	Neighborhood	African Restaurant	American Restaurant	Asian Restaurant	BBQ Joint	Brazilian Restaurant	Brewery	Burrito Place	Cajun / Creole Restaurant
0	Armour Square,Wentworth Gardens,Chinatown	0.0	0.076923	0.000000	0.000000	0.0	0.038462	0.0	0.0
1	Auburn Gresham,Gresham	0.0	0.250000	0.000000	0.000000	0.0	0.000000	0.0	0.0
2	Avalon Park,Stony Island Park,Marynook	0.0	0.095238	0.047619	0.047619	0.0	0.000000	0.0	0.0
3	Bridgeport	0.0	0.095238	0.047619	0.047619	0.0	0.000000	0.0	0.0
4	Brighton Park	0.0	0.095238	0.047619	0.047619	0.0	0.000000	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'] = chi_grouped['Neighborhood']
```

Further, the above created dataframe is fed with the top 5 most common venues categories in the respective neighborhood.

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue
0	Armour Square,Wentworth Gardens,Chinatown	Pizza Place	American Restaurant	Italian Restaurant	Thai Restaurant	Fast Food Restaurant
1	Auburn Gresham,Gresham	Pizza Place	American Restaurant	Fast Food Restaurant	Fried Chicken Joint	Mediterranean Restaurant
2	Avalon Park,Stony Island Park,Marynook	New American Restaurant	Italian Restaurant	Mediterranean Restaurant	American Restaurant	Pizza Place
3	Bridgeport	New American Restaurant	Italian Restaurant	Mediterranean Restaurant	American Restaurant	Pizza Place
4	Brighton Park	New American Restaurant	Italian Restaurant	Mediterranean Restaurant	American Restaurant	Pizza Place

Machine Learning

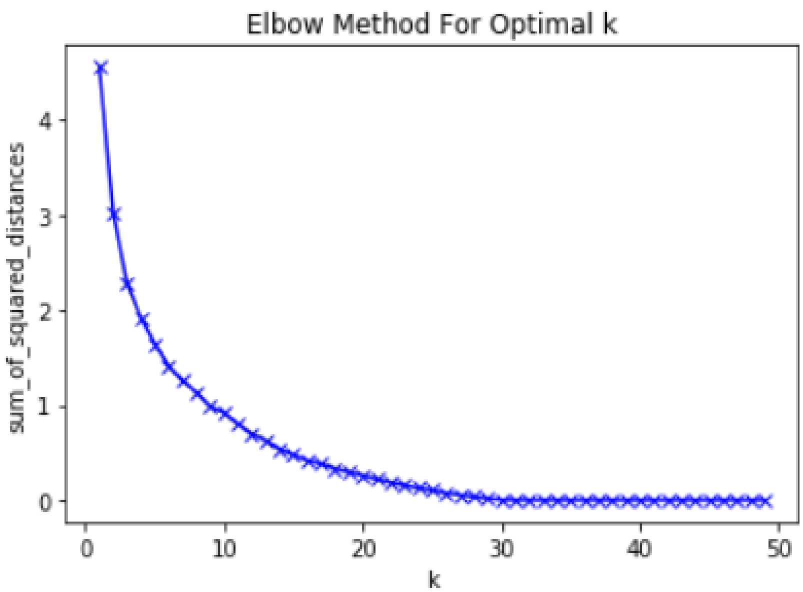
'k-means' is an unsupervised machine learning algorithm which creates clusters of data points aggregated together because of certain similarities. This algorithm will be used to count neighborhoods for each cluster label for variable cluster size. To implement this algorithm, it is very important to 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(chi_grouped_clustering)
    sum_of_squared_distances.append(kmeans.inertia_)
```

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



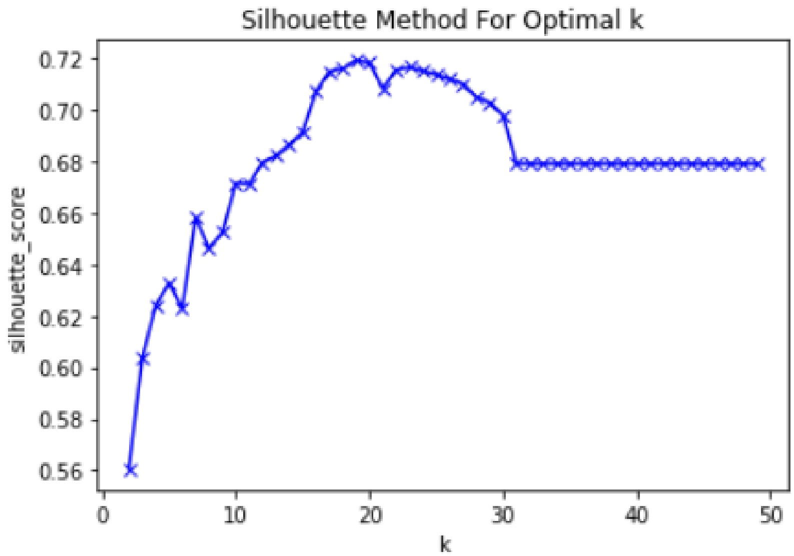
Sometimes, Elbow method does not give the required result, which happened in this case. As, there is a gradual decrease in the sum of squared distances, optimal number of clusters cannot be determined. To counter this, another method can be implemented, as discussed below.

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(chi_grouped_clustering)
    labels = kmeans.labels_
    sil.append(silhouette_score(chi_grouped_clustering, labels, metric = 'euclidean'))
```

```
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 = 6 and k = 8. Four clusters will give a very broad classification of the venues.

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(chi_grouped_clustering)

print(Counter(kmeans.labels_))
```

```
Counter({0: 50, 1: 15, 6: 3, 4: 3, 5: 3, 3: 3, 7: 2, 2: 2})
```

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(5)
```

	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	Armour Square,Wentworth Gardens,Chinatown	Pizza Place	American Restaurant	Italian Restaurant	Thai Restaurant	Fast Food Restaurant
1	6	Auburn Gresham,Gresham	Pizza Place	American Restaurant	Fast Food Restaurant	Fried Chicken Joint	Mediterranean Restaurant
2	0	Avalon Park,Stony Island Park,Marynook	New American Restaurant	Italian Restaurant	Mediterranean Restaurant	American Restaurant	Pizza Place
3	0	Bridgeport	New American Restaurant	Italian Restaurant	Mediterranean Restaurant	American Restaurant	Pizza Place
4	0	Brighton Park	New American Restaurant	Italian Restaurant	Mediterranean Restaurant	American Restaurant	Pizza Place

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

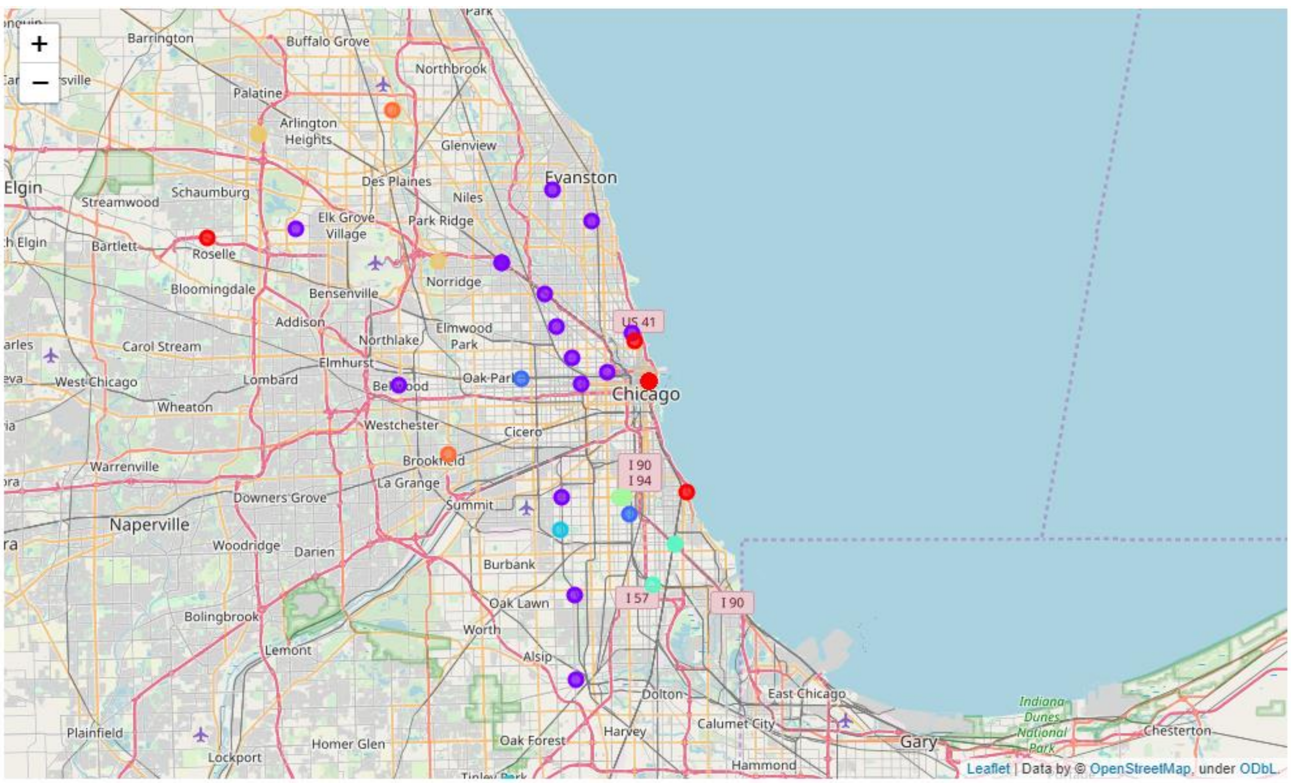
```
# merge neighborhoods_venues_sorted with nyc_data to add latitude/longitude for each neighborhood
chi_merged = neighborhoods_venues_sorted.join(neighborhoods.set_index('Neighborhood'), on='Neighborhood')
chi_merged.head()
```

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	Longituc
1	Armour Square,Wentworth Gardens,Chinatown	Pizza Place	American Restaurant	Italian Restaurant	Thai Restaurant	Fast Food Restaurant	Armour Square	41.892001	-87.6656
6	Auburn Gresham,Gresham	Pizza Place	American Restaurant	Fast Food Restaurant	Fried Chicken Joint	Mediterranean Restaurant	Auburn Gresham	42.078163	-88.0316
0	Avalon Park,Stony Island Park,Marynook	New American Restaurant	Italian Restaurant	Mediterranean Restaurant	American Restaurant	Pizza Place	Avalon Park	41.885310	-87.6221
0	Bridgeport	New American Restaurant	Italian Restaurant	Mediterranean Restaurant	American Restaurant	Pizza Place	Bridgeport	41.885310	-87.6221
0	Brighton Park	New American Restaurant	Italian Restaurant	Mediterranean Restaurant	American Restaurant	Pizza Place	Brighton Park	41.885310	-87.6221

Again, the Chicago 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(chi_merged['Latitude'], chi_merged['Longitude'], chi_merged['Neighborhoo
d'], chi_merged['Cluster Labels']):
    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 Chicago's neighborhoods:



Results

Cluster 0

```
cluster_0 = chi_merged.loc[chi_merged['Cluster Labels'] == 0, chi_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
2	Avalon Park,Stony Island Park,Marynook	New American Restaurant	Italian Restaurant	Mediterranean Restaurant	American Restaurant	Pizza Place	Avalon Park	41.885310	-87.622130
3	Bridgeport	New American Restaurant	Italian Restaurant	Mediterranean Restaurant	American Restaurant	Pizza Place	Bridgeport	41.885310	-87.622130
4	Brighton Park	New American Restaurant	Italian Restaurant	Mediterranean Restaurant	American Restaurant	Pizza Place	Brighton Park	41.885310	-87.622130
5	Bucktown,Logan Square,Palmer Square,Kosciuszko...	Italian Restaurant	Dumpling Restaurant	Mediterranean Restaurant	Souvlaki Shop	Chinese Restaurant	Logan Square	41.997596	-88.087459
6	Burnside	New American Restaurant	Italian Restaurant	Mediterranean Restaurant	American Restaurant	Pizza Place	Burnside	41.885310	-87.622130

Following are the results of the Cluster – 0 analysis:

```
New American Restaurant    47
Italian Restaurant          2
American Restaurant         1
Name: 1st Most Common Venue, dtype: int64
---------------------------------------------
Italian Restaurant       47
Caribbean Restaurant      1
Dumpling Restaurant       1
Pizza Place               1
Name: 2nd Most Common Venue, dtype: int64
---------------------------------------------
```

Cluster 1

Cluster 1

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

59]:

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	Borough	Latitude	Longitude
30	Hyde Park,East Hyde Park	Pizza Place	Fast Food Restaurant	Chinese Restaurant	Mexican Restaurant	Fried Chicken Joint	Hyde Park	41.881518	-87.885680
38	Merchant Park,Irving Park,The Villa,Avondale G...	Mexican Restaurant	American Restaurant	Pizza Place	Asian Restaurant	Thai Restaurant	Irving Park	41.953613	-87.731348
52	Ravenswood Gardens,Lincoln Square,Bowmanville,...	Mexican Restaurant	Pizza Place	Fast Food Restaurant	Fried Chicken Joint	Chinese Restaurant	Lincoln Square	42.003933	-87.994238
71	West Elsdon	Mexican Restaurant	Pizza Place	Fast Food Restaurant	American Restaurant	Fried Chicken Joint	West Elsdon	41.793816	-87.713849
77	West Ridge,Peterson Park,West Rogers Park,Rose...	Mexican Restaurant	Fast Food Restaurant	Pizza Place	Caribbean Restaurant	Fried Chicken Joint	West Ridge	42.010058	-87.682287

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

```
Mexican Restaurant    4
Pizza Place           2
Name: 1st Most Common Venue, dtype: int64
--------------------------------------------
Fast Food Restaurant    2
Pizza Place             2
Mexican Restaurant      1
American Restaurant     1
Name: 2nd Most Common Venue, dtype: int64
--------------------------------------------
West Elsdon       1
Hyde Park         1
Irving Park       1
West Ridge        1
Lincoln Square    1
Woodlawn          1
Name: Borough, dtype: int64
--------------------------------------------
```

Cluster 2

```
cluster_2 = chi_merged.loc[chi_merged['Cluster Labels'] == 2, chi_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
65	South East Ravenswood,Boystown,Graceland West,...	Fast Food Restaurant	Fried Chicken Joint	BBQ Joint	Southern / Soul Food Restaurant	Seafood Restaurant	Lake View	41.887056	-87.756631
73	West Englewood	Fast Food Restaurant	American Restaurant	BBQ Joint	Seafood Restaurant	Mexican Restaurant	West Englewood	41.780689	-87.642713

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

```
Fast Food Restaurant    2
Name: 1st Most Common Venue, dtype: int64
--------------------------------------------
Fried Chicken Joint    1
American Restaurant    1
Name: 2nd Most Common Venue, dtype: int64
--------------------------------------------
Lake View         1
West Englewood    1
Name: Borough, dtype: int64
--------------------------------------------
```

Cluster 3

Cluster 3

```
cluster_3 = chi_merged.loc[chi_merged['Cluster Labels'] == 3, chi_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
16	East Garfield Park,Fifth City	American Restaurant	Fast Food Restaurant	BBQ Joint	African Restaurant	Caribbean Restaurant	East Garfield Park	41.793750	-87.647518
22	Garfield Ridge,Vittum Park,LeClaire Courts,Sle...	American Restaurant	Fast Food Restaurant	BBQ Joint	African Restaurant	Caribbean Restaurant	Garfield Ridge	41.793750	-87.647518
72	West Englewood	Fast Food Restaurant	American Restaurant	BBQ Joint	Seafood Restaurant	Mexican Restaurant	West Englewood	41.780689	-87.642713
73	West Garfield Park	Fast Food Restaurant	Southern / Soul Food Restaurant	American Restaurant	African Restaurant	Pizza Place	West Garfield Park	41.793681	-87.652373
80	Wrigleyville,North Halsted,South East Ravenswo...	Fast Food Restaurant	Fried Chicken Joint	BBQ Joint	Southern / Soul Food Restaurant	Seafood Restaurant	Lake View	41.887056	-87.756631

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

```
Fast Food Restaurant    3
American Restaurant     2
Name: 1st Most Common Venue, dtype: int64
---------------------------------------------
Fast Food Restaurant               2
Fried Chicken Joint                1
American Restaurant                1
Southern / Soul Food Restaurant    1
Name: 2nd Most Common Venue, dtype: int64
---------------------------------------------
West Englewood        1
Lake View             1
Garfield Ridge        1
West Garfield Park    1
East Garfield Park    1
Name: Borough, dtype: int64
---------------------------------------------
```

Cluster 4

Cluster 4

```
cluster_4 = chi_merged.loc[chi_merged['Cluster Labels'] == 4, chi_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
0	Albany Park,Ravenswood Manor,Mayfair,North May...	Fast Food Restaurant	Mexican Restaurant	BBQ Joint	Chinese Restaurant	Taco Place	Albany Park	41.717189	-87.699098
1	Armour Square,Wentworth Gardens,Chinatown	Pizza Place	American Restaurant	Italian Restaurant	Thai Restaurant	Fast Food Restaurant	Armour Square	41.892001	-87.665688
17	East Side	Pizza Place	Fast Food Restaurant	Chinese Restaurant	Greek Restaurant	Cajun / Creole Restaurant	East Side	42.034514	-87.723638
24	Grand Boulevard,Legends South (Robert Taylor H...	Mexican Restaurant	Pizza Place	American Restaurant	French Restaurant	Fast Food Restaurant	Grand Boulevard	41.922830	-87.638832
28	Homan Square,Douglas Park,K-Town,North Lawndale	Mexican Restaurant	Pizza Place	Chinese Restaurant	Latin American Restaurant	Taco Place	North Lawndale	41.928230	-87.719382

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

```
Pizza Place             4
Fast Food Restaurant    2
Mexican Restaurant      2
Italian Restaurant      2
American Restaurant     1
BBQ Joint               1
Name: 1st Most Common Venue, dtype: int64
---------------------------------------------
Pizza Place                        3
Greek Restaurant                   2
Mexican Restaurant                 2
Fast Food Restaurant               1
Caribbean Restaurant               1
American Restaurant                1
Dumpling Restaurant                1
Southern / Soul Food Restaurant    1
Name: 2nd Most Common Venue, dtype: int64
---------------------------------------------
```

Cluster 5

Cluster 5

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

7]:

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	Borough	Latitude	Longitude
35	Marquette Park,Lithuanian Plaza,Chicago Lawn	Fried Chicken Joint	Fast Food Restaurant	Pizza Place	Chinese Restaurant	Southern / Soul Food Restaurant	Chicago Lawn	41.756784	-87.593334
59	South Chicago	Fried Chicken Joint	Fast Food Restaurant	Pizza Place	Chinese Restaurant	Southern / Soul Food Restaurant	South Chicago	41.756784	-87.593334
79	Wildwood,Edgebrook,Sauganash,South Edgebrook,F...	Fast Food Restaurant	Fried Chicken Joint	Pizza Place	American Restaurant	Southern / Soul Food Restaurant	Forest Glen	41.725504	-87.616658

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

```
Fried Chicken Joint     2
Fast Food Restaurant    1
Name: 1st Most Common Venue, dtype: int64
--------------------------------------------
Fast Food Restaurant    2
Fried Chicken Joint     1
Name: 2nd Most Common Venue, dtype: int64
--------------------------------------------
Chicago Lawn     1
South Chicago    1
Forest Glen      1
Name: Borough, dtype: int64
--------------------------------------------
```

Cluster 6

Cluster 6

```
cluster_6 = chi_merged.loc[chi_merged['Cluster Labels'] == 6, chi_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
43	North Park,River's Edge,Hollywood Park	American Restaurant	Fast Food Restaurant	Mexican Restaurant	Whisky Bar	Fried Chicken Joint	North Park	42.097329	-87.891911
69	Washington Park	American Restaurant	Mexican Restaurant	Caribbean Restaurant	African Restaurant	Czech Restaurant	Washington Park	41.828125	-87.832676

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

```
American Restaurant    2
Name: 1st Most Common Venue, dtype: int64
--------------------------------------------
Fast Food Restaurant    1
Mexican Restaurant      1
Name: 2nd Most Common Venue, dtype: int64
--------------------------------------------
Washington Park    1
North Park         1
Name: Borough, dtype: int64
--------------------------------------------
```

Cluster 7

cluster 7

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

1]:

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	Borough	Latitude	Longitude
58	South Austin,North Austin,The Island,Galewood	Mexican Restaurant	Pizza Place	Chinese Restaurant	Whisky Bar	French Restaurant	Austin	41.568075	-87.769531
61	South Lawndale,Little Village,Marshall Square	Mexican Restaurant	Chinese Restaurant	Seafood Restaurant	Italian Restaurant	Fast Food Restaurant	South Lawndale	41.768296	-87.715295
74	West Humboldt Park	Mexican Restaurant	Pizza Place	Chinese Restaurant	Whisky Bar	French Restaurant	Austin, Humboldt Park	41.568075	-87.769531

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

```
Mexican Restaurant    3
Name: 1st Most Common Venue, dtype: int64
---------------------------------------------
Pizza Place           2
Chinese Restaurant    1
Name: 2nd Most Common Venue, dtype: int64
---------------------------------------------
South Lawndale           1
Austin, Humboldt Park    1
Austin                   1
Name: Borough, dtype: int64
---------------------------------------------
```

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	**1st Most Common Venue**	**2nd Most Common Venue**	**Borough**
0	American Restaurant	Italian Restaurant	Ashburn, Belmont Cragin, Morgan Park, Bridgeport, Brighton Park
1	Mexican Restaurant	Fast Food Restaurant	Hyde Park, Irving Park, Lincoln Square, West Elsdon, West Ridge
2	Pizza Place	American Restaurant	Uptown, Auburn Gresham, Lakeview
3	Fast Food Restaurant	American Restaurant	West Englewood, Lake View, Garfiel d Ridge, West Garfield Park, East Gar field Park
4	Pizza Place	Fast Food Restaurant	Albany Park, Armour Square, East Side, Grand Boulevard, North Lawndale
5	Fried Chicken Joint	Fast Food Restaurant	Chicago Lawn, South Chicago, Forest Glen
6	American Restaurant	Fast Food Restaurant	North Park, Washington Park
7	Mexican Restaurant	Pizza Place	Austin, South Lawndale, Humboldt Park

Fast food, who does not like it. And it is obvious from the analysis that Fast Food Restaurant is the most common venue across all the clusters or neighborhoods.

So, as Fast food is a ready-to-go place for Chicago 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 - American Restaurant
- Cluster 1 - Mexican Restaurant
- Cluster 2 - Pizza Place
- Cluster 3 - Fast Food Restaurant
- Cluster 4 - Pizza Place
- Cluster 5 - Fried Chicken Joint
- Cluster 6 - American Restaurant
- Cluster 7 - Mexican Restaurant

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 Chicago 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 Chicago City and so is the diversity statistics.

The results of this project can be improved and made more inquisitive by using a current Chicago 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.