Coursera Capstone
The Battle of the Neighborhoods
Restaurant/Café Location Identifier

15 November 2020
Reem AlAbdouli

Business Problem

New York is a magnet for tourists from all around the globe. It allures international spotlight where it is one of the most sought after travel destination due to its cultural, ethnic and natural diversity, world best museums and art galleries, developed infrastructure and fine educational institutions. Also, New York is the heart of trade as economic growth as well as the best technological, medical and scientific minds in the world which makes it a strong competitor on the world map. This project will focus on Manhattan because the possibilities are endless where it has a dense population, beautiful skyscrapers, lavish shopping, tourist attractions, iconic historical structures, fine and performing arts, beautiful parks, recreational facilities and some of best restaurants in the world.

Since New York is host to culinary experts from all across the globe and has one of the most competitive and diverse restaurant scenes in the world, it will not be easy to casually predict if opening a certain restaurant/café in Manhattan will be successful or not. This is where this project makes a breakthrough in helping food business seekers to decide the best locations for their restaurant/café. So, the aim of this project is to use clustering techniques to group neighborhoods in Manhattan and support food business seekers to decide which neighborhood will be best suited for their business. This project helps food business seekers to analyze neighborhoods in Manhattan, which was rather hard to do without algorithm, and identify a place for their business after analysis. The user is prompted to input a cuisine they are interested in then the analysis starts based on this user's entry. As an example, in this project the user will enter Italian. After that, the data sources will be used accordingly to analyze neighborhoods, popular venues and food venues then cluster the neighborhoods accordingly. These clusters will help anyone who is seeking to open a restaurant/café to decide on the location of his/her business. In the example of the user entering Italian cuisine, the clusters will indicate which are the best Neighborhoods in Manhattan the user can open an Italian restaurant.

Data Sources

Since this project is data driven, it is important to select and include all the required data. However, in this project only two data sources were used which are listed below:

1. New York Neighborhood Dataset which includes all New York Boroughs, Neighborhoods and locations (latitude, longitude). This dataset will help in finding the required venues and information from Foursquare API. It will also support in clustering.
   https://cocl.us/new_york_dataset
2. Foursquare Places API which gives real-time access to Foursquare's global database of rich venue data and user content. This will help in analyzing the neighborhoods in terms of the popular venues and food venues. It provides features regarding venues such as rating and price which will be helpful in clustering.
   https://developer.foursquare.com/docs/places-api/

There are various data sources that may be used to enrich this project and improve the clustering and in return enhance the predictions. For example, Foursquare Places Database may be used to get some restaurant/café features that are not available in the Foursquare Places API such as service quality, whether the food venue is crowded, whether food is worth the price and whether the food venue is trendy. Also, obtaining data about the demographic, social and economic characteristics of the people in Manhattan would've helped to analyze and understand their interests and tendencies.

Methodology

In order to identify the Neighborhoods which are optimal for user cuisine of interest, in this case Italian, there are several tasks done as follows:

- Import all required libraries as seen in Figure 1.

```
import pandas as pd
import geocoder # to get latitude and longitude values
import numpy as np # library to handle data in a vectorized manner
import json # library to handle JSON files
import requests # library to handle requests
from pandas.io.json import json_normalize # tranform JSON file into a pandas dataframe
from sklearn.cluster import KMeans # import k-means from clustering stage
!conda install -c conda-forge folium=0.5.0 --yes
import folium # map rendering library
# Matplotlib and associated plotting modules
import matplotlib.cm as cm
import matplotlib.colors as colors
!conda install -c conda-forge geopy --yes
from geopy.geocoders import Nominatim
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn import datasets
from yellowbrick.cluster import SilhouetteVisualizer
from sklearn.metrics import silhouette_score
```

```
Collecting package metadata (current_repodata.json): ...working... done
Solving environment: ...working... done

# All requested packages already installed.

Collecting package metadata (current_repodata.json): ...working... done
Solving environment: ...working... done

# All requested packages already installed.
```

Figure 1: Libraries Imported

- Extract the JSON file from data source 1 which includes information about New York Neighborhoods and Boroughs. The wget command was used to get the JSON file as shown in Figure 2 and then it was converted to a Dataframe which is presented in Figure 3.

```
!wget https://cocl.us/new_york_dataset
```

Load the data

```
with open('new_york_dataset') as json_data:
    newyork_data = json.load(json_data)
```

```
newyork_data
```

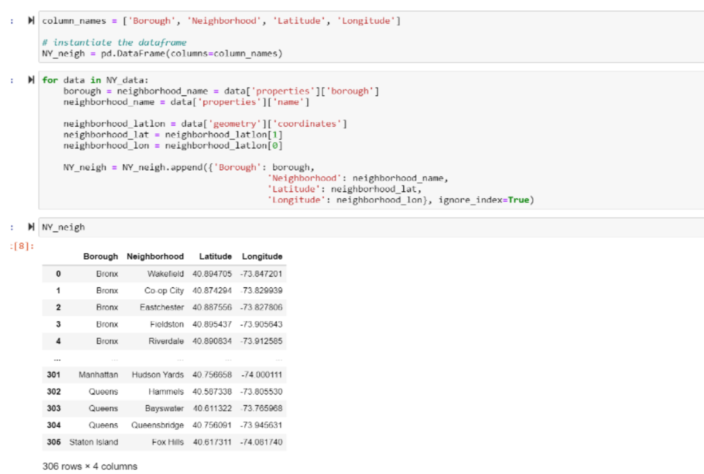
```
4]: {'type': 'FeatureCollection',
 'totalFeatures': 306,
 'features': [{'type': 'Feature',
   'id': 'nyu_2451_34572.1',
   'geometry': {'type': 'Point',
    'coordinates': [-73.84720052054902, 40.89470517661]},
   'geometry_name': 'geom',
   'properties': {'name': 'Wakefield',
    'stacked': 1,
    'annoline1': 'Wakefield',
    'annoline2': None,
    'annoline3': None,
    'annoangle': 0.0,
    'borough': 'Bronx',
    'bbox': [-73.84720052054902,
     40.89470517661,
     -73.84720052054902,
     40.89470517661]}},
  {'type': 'Feature',
   'id': 'nyu_2451_34572.2',
```

Get only relevant data which is in the features key

```
NY_data = newyork_data['features']
NY_data[0]
```

```
5]: {'type': 'Feature',
 'id': 'nyu_2451_34572.1',
 'geometry': {'type': 'Point',
  'coordinates': [-73.84720052054902, 40.89470517661]},
 'geometry_name': 'geom',
 'properties': {'name': 'Wakefield',
  'stacked': 1,
  'annoline1': 'Wakefield',
  'annoline2': None,
  'annoline3': None,
  'annoangle': 0.0,
  'borough': 'Bronx',
  'bbox': [-73.84720052054902,
```

Figure 2: Extract New York Dataset in JSON Format using wget command



```
column_names = ['Borough', 'Neighborhood', 'Latitude', 'Longitude']

# instantiate the dataframe
NY_neigh = pd.DataFrame(columns=column_names)
```

```
for data in NY_data:
    borough = neighborhood_name = data['properties']['borough']
    neighborhood_name = data['properties']['name']

    neighborhood_latlon = data['geometry']['coordinates']
    neighborhood_lat = neighborhood_latlon[1]
    neighborhood_lon = neighborhood_latlon[0]

    NY_neigh = NY_neigh.append({'Borough': borough,
                                'Neighborhood': neighborhood_name,
                                'Latitude': neighborhood_lat,
                                'Longitude': neighborhood_lon}, ignore_index=True)
```

```
NY_neigh
```

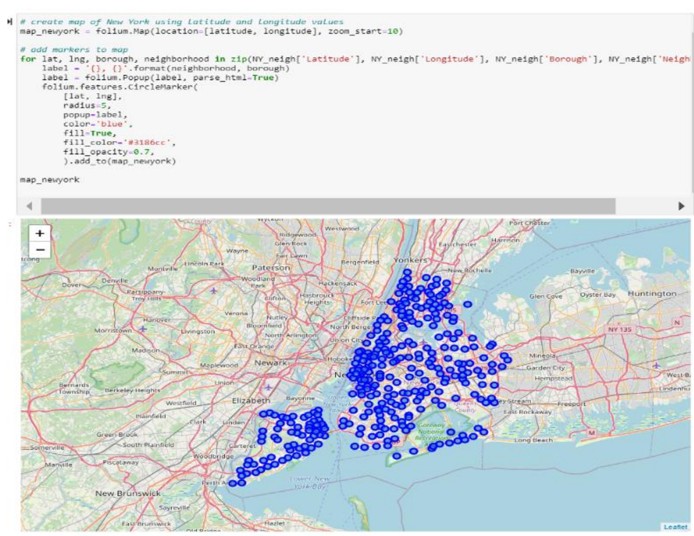
:[8]:

	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
...	...	...	...	...
301	Manhattan	Hudson Yards	40.756658	-74.000111
302	Queens	Hammels	40.587338	-73.805530
303	Queens	Bayswater	40.611322	-73.765968
304	Queens	Queensbridge	40.756091	-73.945631
305	Staten Island	Fox Hills	40.617311	-74.081740

306 rows × 4 columns

Figure 3: Convert New York Dataset from JSON to Dataframe

- Visualize the Neighborhoods of New York on Follium map which is helpful in understanding them as seen in Figure 4.



```
# create map of New York using latitude and longitude values
map_newyork = folium.Map(location=[latitude, longitude], zoom_start=10)

# add markers to map
for lat, lng, borough, neighborhood in zip(NY_neigh['Latitude'], NY_neigh['Longitude'], NY_neigh['Borough'], NY_neigh['Neigh
    label = '{}, {}'.format(neighborhood, borough)
    label = folium.Popup(label, parse_html=True)
    folium.features.CircleMarker(
        [lat, lng],
        radius=5,
        popup=label,
        color='blue',
        fill=True,
        fill_color='#3186cc',
        fill_opacity=0.7,
        ).add_to(map_newyork)

map_newyork
```


Figure 4: Show New York Neighborhoods on geo map

- Focus on Manhattan Borough only and visualize its neighborhoods on Follium Map which is shown in Figure 5 and 6.

Focus on Manhattan Borough only

```python
manhattan_dataset = NY_neigh[NY_neigh['Borough'] == 'Manhattan'].reset_index(drop=True)
manhattan_dataset
```

[12]:

	Borough	Neighborhood	Latitude	Longitude
0	Manhattan	Marble Hill	40.876551	-73.910660
1	Manhattan	Chinatown	40.715618	-73.994279
2	Manhattan	Washington Heights	40.851903	-73.936900
3	Manhattan	Inwood	40.867684	-73.921210
4	Manhattan	Hamilton Heights	40.823604	-73.949688
5	Manhattan	Manhattanville	40.816934	-73.957385
6	Manhattan	Central Harlem	40.815976	-73.943211
7	Manhattan	East Harlem	40.792249	-73.944182
8	Manhattan	Upper East Side	40.775639	-73.960508
9	Manhattan	Yorkville	40.775930	-73.947118
10	Manhattan	Lenox Hill	40.768113	-73.958860

Figure 5: Focus on Manhattan Borough only (Dataframe)

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

# add markers to map
for lat, lng, label in zip(manhattan_dataset['Latitude'], manhattan_dataset['Longitude'], manhattan_dataset['Neighborhood'])
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius=5,
        popup=label,
        color='blue',
        fill=True,
        fill_color='#3186cc',
        fill_opacity=0.7,
        parse_html=False).add_to(map_manhattan)

map_manhattan
```

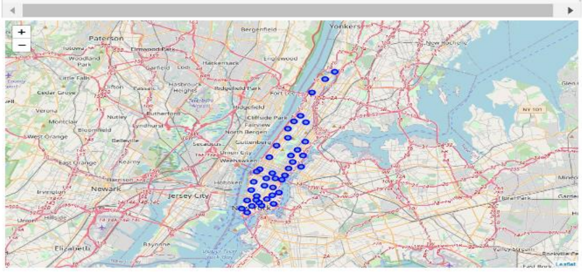


Figure 6: Map showing Manhattan Neighborhoods

- Connect to the Foursquare Places API (Data Source 2) using the client ID and client secret.
- Using the Foursquare Places API and the Manhattan Neighborhoods Dataframe, a call is made to the API to get all popular venues names, categories, latitude and longitude for each Manhattan Neighborhood. JSON file is returned with the call and this file is converted to a Dataframe. This is shown in Figure 7 and Figure 8.

```python
def getNearbyVenues(names, latitudes, longitudes, radius=500):

    venues_list=[]
    for name, lat, lng in zip(names, latitudes, longitudes):
        print(name)

        # create the API request URL
        url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={},{}&radius={}&limit={}'
            CLIENT_ID,
            CLIENT_SECRET,
            VERSION,
            lat,
            lng,
            radius,
            LIMIT)

        # make the GET request
        results = requests.get(url).json()["response"]['groups'][0]['items']


        # return only relevant information for each nearby venue
        venues_list.append([(
            name,
            lat,
            lng,
            v['venue']['name'],
            v['venue']['id'],
            v['venue']['location']['lat'],
            v['venue']['location']['lng'],
            v['venue']['categories'][0]['name']) for v in results])

    nearby_venues = pd.DataFrame([item for venue_list in venues_list for item in venue_list])
    nearby_venues.columns = ['Neighborhood',
                  'Neighborhood Latitude',
                  'Neighborhood Longitude',
                  'Nearby Venue Name',
                  'Nearby Venue ID',
                  'Nearby Venue Latitude',
                  'Nearby Venue Longitude',
                  'Nearby Venue Category']

    return(nearby_venues)
```

```python
LIMIT=100
manhattan_venues = getNearbyVenues(names=manhattan_dataset['Neighborhood'],
                                   latitudes=manhattan_dataset['Latitude'],
                                   longitudes=manhattan_dataset['Longitude']
                                  )
```

Figure 7: Places API call to get popular venues in each Neighborhoods

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Nearby Venue Name	Nearby Venue ID	Nearby Venue Latitude	Nearby Venue Longitude	Nearby Venue Category
0	Marble Hill	40.876551	-73.910660	Arturo's	4b4429abf964a52037f225e3	40.874412	-73.910271	Pizza Place
1	Marble Hill	40.876551	-73.910660	Bikram Yoga	4baf59e8f964a520a6f93be3	40.876844	-73.906204	Yoga Studio
2	Marble Hill	40.876551	-73.910660	Tibbett Diner	4b79cc46f964a520c5122fe3	40.880404	-73.908937	Diner
3	Marble Hill	40.876551	-73.910660	Dunkin'	4b5357adf964a520319827e3	40.877136	-73.906666	Donut Shop
4	Marble Hill	40.876551	-73.910660	Starbucks	55f81cd2498ee903149fcc64	40.877531	-73.905582	Coffee Shop
...	...	...	...	...	...	...	...	...
3199	Hudson Yards	40.756658	-74.000111	StarDust	512ff93be4b0ef4effd7560b	40.759869	-73.996460	Nightclub
3200	Hudson Yards	40.756658	-74.000111	Big George's Smokehouse	59da6a2c0a08ab3b81d58d8d	40.757954	-74.002296	BBQ Joint
3201	Hudson Yards	40.756658	-74.000111	NY Waterway 42nd St Bus	4d8bee412505a35df076a352	40.760050	-74.003379	Bus Station
3202	Hudson Yards	40.756658	-74.000111	Twilight Cruise By Citysightseeing	4e3dce6f1f6e844231eb1f23	40.759744	-74.004096	Boat or Ferry
3203	Hudson Yards	40.756658	-74.000111	City Lights Cruises	4e63fcfe8877954de8de286a	40.759804	-74.004025	Boat or Ferry

3204 rows × 8 columns

Figure 8: The resulting Dataframe containing the popular venues

- ○ The interest here is to get all hotspots in each Neighborhood so that we use them later to check each food venue is surrounded with what hotspots. This can help in figuring out if these hotspots will have an influence in visiting the user's cuisine of interest. For example, if in a certain Neighborhood there are various tourist attraction areas this might indicate that after the tourist visit these attractions they might want a place to eat. So, this Neighborhood might be a potential for a user's restaurant/café. So, the call made to the API above gets all popular venues for each Neighborhood; however, for the purpose stated above any food related venue will be omitted as they are not relevant for this task.

- In the Places API, there are 4 levels of categories where the first level is the Primary category which serves as an umbrella for the other levels. Categories provided by the call above are not the primary categories, so in order to match each category from the call above to its primary category, another call is made to the API to get a JSON file of all levels of categories. Then it is converted to a Dataframe which will be used to match the categories that were retrieved earlier with the popular venues. After getting all the primary category for the venues, the Food category was omitted from the Dataframe this because it is not required at this stage as mentioned previously. So now the Dataframe contains the Neighborhoods and the venue primary categories each contains. (Figure 9,10,11)

Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Nearby Venue Name	Nearby Venue ID	Nearby Venue Latitude	Nearby Venue Longitude	Nearby Venue Category	Nearby_Venue_Primary_Category
Marble Hill	40.876551	-73.910660	Arturo's	4b4429abf964a52037f225e3	40.874412	-73.910271	Pizza Place	Food
Marble Hill	40.876551	-73.910660	Bikram Yoga	4baf59e8f964a520a6f93be3	40.876844	-73.906204	Yoga Studio	Outdoors & Recreation
Marble Hill	40.876551	-73.910660	Tibbett Diner	4b79cc46f964a520c5122fe3	40.880404	-73.908937	Diner	Food
Marble Hill	40.876551	-73.910660	Dunkin'	4b5357adf964a520319827e3	40.877136	-73.906666	Donut Shop	Food
Marble Hill	40.876551	-73.910660	Starbucks	55f81cd2498ee903149fcc64	40.877531	-73.905582	Coffee Shop	Food
...	...	...	...	...	...	...	...	...
Hudson Yards	40.756658	-74.000111	StarDust	512ff93be4b0ef4effd7560b	40.759869	-73.996460	Nightclub	Nightlife Spot
Hudson Yards	40.756658	-74.000111	Big George's Smokehouse	59da6a2c0a08ab3b81d58d8d	40.757954	-74.002296	BBQ Joint	Food
Hudson Yards	40.756658	-74.000111	NY Waterway 42nd St Bus	4d8bee412505a35df076a352	40.760050	-74.003379	Bus Station	Travel & Transport
Hudson Yards	40.756658	-74.000111	Twilight Cruise By Citysightseeing	4e3dce6f1f6e844231eb1f23	40.759744	-74.004096	Boat or Ferry	Travel & Transport
Hudson Yards	40.756658	-74.000111	City Lights Cruises	4e63fcfe8877954de8de286a	40.759804	-74.004025	Boat or Ferry	Travel & Transport

Figure 9: Adding the primary category for each venue to the Dataframe

Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Nearby Venue Name	Nearby Venue ID	Nearby Venue Latitude	Nearby Venue Longitude	Nearby Venue Category	Nearby_Venue_Primary_Category
Marble Hill	40.876551	-73.910660	Bikram Yoga	4baf59e8f964a520a6f93be3	40.876844	-73.906204	Yoga Studio	Outdoors & Recreation
Marble Hill	40.876551	-73.910660	Rite Aid	4b88e053f964a5208a1132e3	40.875467	-73.908906	Pharmacy	Shop & Service
Marble Hill	40.876551	-73.910660	TCR The Club of Riverdale	4a725fa1f964a520f6da1fe3	40.878628	-73.914568	Tennis Stadium	Arts & Entertainment
Chinatown	40.715618	-73.994279	Hotel 50 Bowery	578692f4498e1054905dbde7	40.715936	-73.996789	Hotel	Travel & Transport
Chinatown	40.715618	-73.994279	Bar Belly	503fffabe4b05e5c0eace385	40.715135	-73.991802	Cocktail Bar	Nightlife Spot
...	...	...	...	...	...	...	...	...
Hudson Yards	40.756658	-74.000111	Brooklyn Fare	514ca6f57043482a0c84e7ab	40.756130	-73.996614	Supermarket	Shop & Service
Hudson Yards	40.756658	-74.000111	DiMenna Center for Classical Music	4daf05e343a1f97d91988078	40.756323	-73.997192	Music School	Professional & Other Places
Hudson Yards	40.756658	-74.000111	505W37	4b6c43bdf964a5208f2c2ce3	40.756909	-73.998007	Residential Building (Apartment / Condo)	Residence
Hudson Yards	40.756658	-74.000111	Equinox Hotel - Hudson Yards	5c9aaee431ac6c0039a49499	40.754768	-74.001986	Hotel	Travel & Transport
Hudson Yards	40.756658	-74.000111	Ada's Place	5b592caba42362002cf7b54f	40.757646	-73.997997	Cocktail Bar	Nightlife Spot

Figure 10: Removing any food category related venues

	Neighborhood	Nearby_Venue_Primary_Category
0	Battery Park City	Arts & Entertainment,Nightlife Spot,Outdoors &...
1	Carnegie Hill	Arts & Entertainment,Nightlife Spot,Outdoors &...
2	Central Harlem	Arts & Entertainment,Nightlife Spot,Outdoors &...
3	Chelsea	Arts & Entertainment,College & University,Nigh...
4	Chinatown	Arts & Entertainment,Nightlife Spot,Outdoors &...
5	Civic Center	Arts & Entertainment,Nightlife Spot,Outdoors &...
6	Clinton	Arts & Entertainment,Nightlife Spot,Outdoors &...
7	East Harlem	Arts & Entertainment,Nightlife Spot,Outdoors &...
8	East Village	Arts & Entertainment,Nightlife Spot,Outdoors &...
9	Financial District	Arts & Entertainment,Nightlife Spot,Outdoors &...
10	Flatiron	Arts & Entertainment,Nightlife Spot,Outdoors &...
11	Gramercy	Arts & Entertainment,Nightlife Spot,Outdoors &...
12	Greenwich Village	Arts & Entertainment,Nightlife Spot,Outdoors &...
13	Hamilton Heights	Arts & Entertainment,Nightlife Spot,Outdoors &...
14	Hudson Yards	Arts & Entertainment,Nightlife Spot,Outdoors &...
15	Inwood	Arts & Entertainment,Nightlife Spot,Outdoors &...
16	Lenox Hill	Arts & Entertainment,College & University,Nigh...
17	Lincoln Square	Arts & Entertainment,College & University,Nigh...
18	Little Italy	Arts & Entertainment,Nightlife Spot,Outdoors &...
19	Lower East Side	Arts & Entertainment,Nightlife Spot,Outdoors &...
20	Manhattan Valley	Nightlife Spot,Outdoors & Recreation,Shop & Se...
21	Manhattanville	Arts & Entertainment,Nightlife Spot,Outdoors &...
22	Marble Hill	Arts & Entertainment,Outdoors & Recreation,Sho...
23	Midtown	Arts & Entertainment,Nightlife Spot,Outdoors &...
24	Midtown South	Arts & Entertainment,Nightlife Spot,Outdoors &...
25	Morningside Heights	Arts & Entertainment,College & University,Nigh...
26	Murray Hill	Arts & Entertainment,Nightlife Spot,Outdoors &...
27	Noho	Arts & Entertainment,Nightlife Spot,Outdoors &...
28	Roosevelt Island	Outdoors & Recreation,Professional & Other Pla...
29	Soho	Arts & Entertainment,Nightlife Spot,Outdoors &...
30	Stuyvesant Town	Nightlife Spot,Outdoors & Recreation,Shop & Se...
31	Sutton Place	Nightlife Spot,Outdoors & Recreation,Shop & Se...
32	Tribeca	Arts & Entertainment,Nightlife Spot,Outdoors &...
33	Tudor City	Nightlife Spot,Outdoors & Recreation,Professio...
34	Turtle Bay	Arts & Entertainment,Nightlife Spot,Outdoors &...
35	Upper East Side	Arts & Entertainment,Nightlife Spot,Outdoors &...
36	Upper West Side	Arts & Entertainment,Nightlife Spot,Outdoors &...
37	Washington Heights	Nightlife Spot,Outdoors & Recreation,Shop & Se...
38	West Village	Arts & Entertainment,Nightlife Spot,Outdoors &...
39	Yorkville	Nightlife Spot,Outdoors & Recreation,Professio...

Figure 11:Dataframe showing the main primary categories for each Neighborhood

- Next, the user is asked to enter their cuisine/café of interest as shown in Figure 12. As an example, the user will enter Italian.

```
user=input('Enter your cuisine/cafe of interest: ')
```

```
Enter your cuisine/cafe of interest: Italian
```

Figure 12: Ask the User to input a cuisine/Cafe of interest

- Using the input from the user in this case Italian, the number of Italian food venue will be counted for each Neighborhood. This will help in showing whether a Neighborhood has no Italian cuisine at all or that a Neighborhood has a huge number of Italian cuisines. This will assess the user to identify which neighborhood could be a potential for his/her business. So, a call is made to the Foursquare Places API to search each Neighborhood in Manhattan for anything related to Italian. The return from the call was converted from JSON to Dataframe. Then, only venues under food category were kept and other categories were omitted this is because the interest is to count the number of Italian cuisine in each Neighborhood, any other non-food categories are irrelevant. This is shown in Figure 13, 14 and 15.

```
def searchvenue(names, latitudes, longitudes,search_query,radius=500):
    trending_list=[]
    for name, lat, lng in zip(names, latitudes, longitudes):
        print(name)
        # create the API request URL
        url = 'https://api.foursquare.com/v2/venues/search?client_id={}&client_secret={}&ll={},{}&v={}&query={}&radius={}&li
        # make the GET request
        results = requests.get(url).json()['response']['venues']
        if(len(results)!=0):
            l=[]
            for v in results:
                if (v['categories']==[]):
                    cat='---'
                else:
                    cat=v['categories'][0]['name']
                l.append((
                        name,
                        lat,
                        lng,
                        v['name'],
                        v['id'],
                        v['location']['lat'],
                        v['location']['lng'],
                        cat))
            trending_list.append(l)

    search = pd.DataFrame([item for trending_list in trending_list for item in trending_list])
    search.columns = ['Neighborhood',
                  'Neighborhood_Latitude',
                  'Neighborhood_Longitude',
                  'Venue_Name',
                  'Venue_ID',
                  'Venue_Latitude',
                  'Venue_Longitude',
                  'Venue_Category']


    return(search)
```

```
LIMIT=100
manhattan_search = searchvenue(names=manhattan_dataset['Neighborhood'],
                                 latitudes=manhattan_dataset['Latitude'],
                                 longitudes=manhattan_dataset['Longitude'],
                                 search_query=user
                                )
```

Figure 13: Call to Places API to search for Italian venues in each neighborhood

eighborhood	Neighborhood_Latitude	Neighborhood_Longitude	Venue_Name	Venue_ID	Venue_Latitude	Venue_Longitude	Venue_Category
Chinatown	40.715618	-73.994279	Italian American Museum Of New York	4b9d4c67f964a5206ca136e3	40.719191	-73.997376	History Museum
Chinatown	40.715618	-73.994279	Italian Food Center	51a63ef7498e9eedb94e0c5f	40.719589	-73.997339	Italian Restaurant
Chinatown	40.715618	-73.994279	Bones & Jones Italian Beef	53823e43498e17831b23b7aa	40.715538	-73.989596	Food Truck
Chinatown	40.715618	-73.994279	Little Italian Rooftop	4d97923e61a3a1cdb92ab942	40.718063	-73.998161	Scenic Lookout
Chinatown	40.715618	-73.994279	Grand Tuscany Ristorante Italiano	4e4c3b37bd413c4cc667d4a8	40.719269	-73.997765	Italian Restaurant
...	...	...	...	...	...	...	...
Flatiron	40.739673	-73.990947	Old Fashioned Hand Made Italian Ice	5014183ce4b0073bd04161fa	40.743568	-73.991700	Food Truck
Flatiron	40.739673	-73.990947	Cesar NYC Kitchens	574c24f9498e3a7d0efcd58c	40.742260	-73.991692	Furniture / Home Store
Hudson Yards	40.756658	-74.000111	Catria Modern Italian	5e53415c3e613c0008e067d8	40.754677	-73.998657	Italian Restaurant
Hudson Yards	40.756658	-74.000111	Villa Fresh Italian Kitchen	4e18ce19483b5cfa49cec3b6	40.758566	-73.999109	Pizza Place
Hudson Yards	40.756658	-74.000111	LI Sal's Italian Kitchen	55e48d4b498e0c90704b9382	40.753831	-73.999194	Food Truck

Figure 14: Resulting Dataframe with Italian venues for each neighborhood

	Neighborhood	Neighborhood_Latitude	Neighborhood_Longitude	Venue_Name	Venue_ID	Venue_Latitude	Venue_Longitude	Venue_Category	Primary_Venue_Category
0	Chinatown	40.715618	-73.994279	Italian American Museum Of New York	4b9d4c67f964a5206ca136e3	40.719191	-73.997376	History Museum	Arts & Entertainment
1	Chinatown	40.715618	-73.994279	Italian Food Center	51a63ef7498e9eedb94e0c5f	40.719589	-73.997339	Italian Restaurant	Food
2	Chinatown	40.715618	-73.994279	Bones & Jones Italian Beef	53823e43498e17831b23b7aa	40.715538	-73.989596	Food Truck	Food
3	Chinatown	40.715618	-73.994279	Little Italian Rooftop	4d97923e61a3a1cdb92ab942	40.718063	-73.998161	Scenic Lookout	Outdoors & Recreation
4	Chinatown	40.715618	-73.994279	La Nonna	4b96bd72f964a5207ce134e3	40.718590	-73.997685	Italian Restaurant	Food
...	...	...	...	...	...	...	...	...	...
171	Flatiron	40.739673	-73.990947	Old Fashioned Hand Made Italian Ice	5014183ce4b0073bd04161fa	40.743568	-73.991700	Food Truck	Food
172	Flatiron	40.739673	-73.990947	Cesar NYC Kitchens	574c24f9498e3a7d0efcd58c	40.742260	-73.991692	Furniture / Home Store	Shop & Service
173	Hudson Yards	40.756658	-74.000111	Villa Fresh Italian Kitchen	4e18ce19483b5cfa49cec3b6	40.758566	-73.999109	Pizza Place	Food
174	Hudson Yards	40.756658	-74.000111	Catria Modern Italian	5e53415c3e613c0008e067d8	40.754677	-73.998657	Italian Restaurant	Food
175	Hudson Yards	40.756658	-74.000111	LI Sal's Italian Kitchen	55e48d4b498e0c90704b9382	40.753831	-73.999194	Food Truck	Food

176 rows × 9 columns

Figure 15: Adding primary category for each venue

```
manhattan_search=manhattan_search[manhattan_search['Primary_Venue_Category']=='Food']
manhattan_search
```

	Neighborhood	Neighborhood_Latitude	Neighborhood_Longitude	Venue_Name	Venue_ID	Venue_Latitude	Venue_Longitude	Venue_Category	Primary_Venue_Category
1	Chinatown	40.715618	-73.994279	Italian Food Center	51a63ef7498e9eedb94e0c5f	40.719589	-73.997339	Italian Restaurant	Food
2	Chinatown	40.715618	-73.994279	Bones & Jones Italian Beef	53823e43498e17831b23b7aa	40.715538	-73.989596	Food Truck	Food
4	Chinatown	40.715618	-73.994279	La Nonna	4b96bd72f964a5207ce134e3	40.718590	-73.997685	Italian Restaurant	Food
5	Chinatown	40.715618	-73.994279	Grand Tuscany Ristorante Italiano	4e4c3b37bd413c4cc667d4a8	40.719269	-73.997765	Italian Restaurant	Food
6	Chinatown	40.715618	-73.994279	Ristorante Italiano	4d950c78647d8cfa5a2a973d	40.718273	-73.997938	Italian Restaurant	Food
...	...	...	...	...	...	...	...	...	...
170	Flatiron	40.739673	-73.990947	Tre Stelle Italian Restaurant&Bar	526324c111d238e68dea4efd	40.743427	-73.991045	Italian Restaurant	Food
171	Flatiron	40.739673	-73.990947	Old Fashioned Hand Made Italian Ice	5014183ce4b0073bd04161fa	40.743568	-73.991700	Food Truck	Food
173	Hudson Yards	40.756658	-74.000111	Villa Fresh Italian Kitchen	4e18ce19483b5cfa49cec3b6	40.758566	-73.999109	Pizza Place	Food
174	Hudson Yards	40.756658	-74.000111	Catria Modern Italian	5e53415c3e613c0008e067d8	40.754677	-73.998657	Italian Restaurant	Food
175	Hudson Yards	40.756658	-74.000111	LI Sal's Italian Kitchen	55e48d4b498e0c90704b9382	40.753831	-73.999194	Food Truck	Food

125 rows × 9 columns

Figure 16: Keeping only food category related venues

- After that, for each venue under the food category in each Neighborhood, the ratings and price range was retrieved from the Places. This is shown in Figure 17 and 18.

```
def details(names,vname, latitudes, longitudes,ids):
    detail_list=[]

    for name, vnames, lat, lng,venue_id in zip(names,vname,latitudes,longitudes,ids):
        print(name)
        # create the API request URL
        url = 'https://api.foursquare.com/v2/venues/{}?client_id={}&client_secret={}&v={}'.format(venue_id, CLIENT_ID,
        # make the GET request
        result = requests.get(url).json()['response']
        l=[]
        if(len(result)!=0):
            results=result['venue']
            if ("rating" in results):
                if (results['rating']==[]):
                    rat='---'
                else:
                    rat=results['rating']
            else:
                rat='---'

            if ("price" in results):
                if (results['price']==[]):
                    p='---'
                else:
                    p=results['price']['message']
            else:
                p='---'

        else:
            rat='---'
            p='---'
            #crowd='---'
            #ser='---'
            #trend='---'
            #vmoney='---'


        l.append(( name,
                  vnames,
                         lat,
                         lng,
                         rat,
                         p
                         #crowd,
                         #ser,
                        # trend,
                        # vmoney
                 ))
        detail_list.append(l)

    search = pd.DataFrame([item for detail_list in detail_list for item in detail_list])
    search.columns = ['Neighborhood',
                        'Venue_Name',
                      'Venue_Latitude',
                      'Venue_Longitude',
                      'Venue_Rating',
                      'Venue_Price'
                      #'Venue_Crowd',
                      #'Venue_Service',
                      #'Venue_Trendy',
                      #'Venue_ValueforMoney'
                         ]


    return(search)
```

```
manhattan_details = details(names=manhattan_search['Neighborhood'],
                          vname=manhattan_search['Venue_Name'],
                                latitudes=manhattan_search['Venue_Latitude'],
                                longitudes=manhattan_search['Venue_Longitude'],
```

Figure 17: Call Places API to get rating and prices for each Italian Food venue

	Neighborhood	Venue_Name	Venue_Latitude	Venue_Longitude	Venue_Rating	Venue_Price
0	Chinatown	Italian Food Center	40.719589	-73.997339	---	Moderate
1	Chinatown	Bones & Jones Italian Beef	40.715538	-73.989596	---	Cheap
2	Chinatown	La Nonna	40.718590	-73.997685	6.7	Moderate
3	Chinatown	Grand Tuscany Ristorante Italiano	40.719269	-73.997765	---	Moderate
4	Chinatown	Ristorante Italiano	40.718273	-73.997938	---	Moderate
...	...	...	...	...	...	...
120	Flatiron	Tre Stelle Italian Restaurant&Bar	40.743427	-73.991045	---	Very Expensive
121	Flatiron	Old Fashioned Hand Made Italian Ice	40.743568	-73.991700	---	Cheap
122	Hudson Yards	Villa Fresh Italian Kitchen	40.758566	-73.999109	---	Cheap
123	Hudson Yards	Catria Modern Italian	40.754677	-73.998657	---	Moderate
124	Hudson Yards	LI Sal's Italian Kitchen	40.753831	-73.999194	---	Cheap

125 rows × 6 columns

Figure 18: Resulting Dataframe containing the rating and price range for each venue

- Then the Dataframe was cleaned in terms of datatype and unknown values. Also, the price returned from the call is in the range of Cheap, Moderate, Expensive and Very Expensive. These string type ranges were mapped to integer type ranges where 1 indicates Cheap, 2 indicated Moderate, 3 indicates Expensive and 4 indicates Very Expensive. (Figure 19)

```
manhattan_detaills=manhattan_details.copy()
manhattan_detaills.drop(['Venue_Latitude','Venue_Longitude'],axis=1,inplace=True)
manhattan_detaills
```

35]:

	Neighborhood	Venue_Name	Venue_Rating	Venue_Price
0	Chinatown	Italian Food Center	---	Moderate
1	Chinatown	Bones & Jones Italian Beef	---	Cheap
2	Chinatown	La Nonna	6.7	Moderate
3	Chinatown	Grand Tuscany Ristorante Italiano	---	Moderate
4	Chinatown	Ristorante Italiano	---	Moderate
...	...	...	...	...
120	Flatiron	Tre Stelle Italian Restaurant&Bar	---	Very Expensive
121	Flatiron	Old Fashioned Hand Made Italian Ice	---	Cheap
122	Hudson Yards	Villa Fresh Italian Kitchen	---	Cheap
123	Hudson Yards	Catria Modern Italian	---	Moderate
124	Hudson Yards	LI Sal's Italian Kitchen	---	Cheap

125 rows × 4 columns

Replace unknown values of rating and price with 0

```
for index, row in manhattan_detaills.iterrows():
        if (row["Venue_Rating"]=="---"):
            manhattan_detaills.at[index,'Venue_Rating']='0'
        if (row["Venue_Price"]=="---"):
            manhattan_detaills.at[index,'Venue_Price']='0'
```

The price has 4 categories (Cheap, moderate, expensive, very expensive)

```
manhattan_detaills['Venue_Price'].unique()
```

```
37]: array(['Moderate', 'Cheap', 'Very Expensive', 'Expensive', '0'],
      dtype=object)
```

Map the 4 price categories with numbers (map Cheap with 1, Moderate with 2, Expensive with 3, Very expensive with 4)

```
manhattan_detaills['Venue_Price']=manhattan_detaills['Venue_Price'].map( {'Cheap':1, 'Moderate':2,'Expensive':3,'Very Expens
```

```
manhattan_detaills['Venue_Rating']=manhattan_detaills['Venue_Rating'].astype(float)
```

Make sure data types in each columns is as required

```
manhattan_detaills.dtypes
```

```
0]: Neighborhood      object
    Venue_Name        object
    Venue_Rating     float64
    Venue_Price        int64
    dtype: object
```

Figure 19: Cleaning the Dataframe

- After cleaning, the Dataframe was grouped by Neighborhood and the number of food venues were counted for each Neighborhood and the average rating and price for each Neighborhood food venues were found. This is seen in Figure 20. The final Dataframe after cleaning and grouping is shown in Figure 21.

Group by Neighborhoods and do the following:

- Count the number of food venues that match the user's cuisine of interest (in the example it's Italian) in each Neighborhood
- Get average rating for these venues above for each neighborhood
- Get the average price for these venues above for each neighborhood

```
manhattan_detaills=manhattan_detaills.groupby('Neighborhood').agg({'Venue_Name':'count',
                    'Venue_Rating':'mean',
                    'Venue_Price':'mean'
                    })
```

Rename the columns

```
manhattan_detail=manhattan_detaills.copy()
manhattan_detail.rename(columns={'Venue_Name':'Number_UserMatched_Venues','Venue_Rating': 'Avg_Venue_Rating','Venue_Price':',
```

Round the Rating and Price column to nearest integers

```
manhattan_detail['Avg_Venue_Rating']=manhattan_detail['Avg_Venue_Rating'].round()
manhattan_detail['Avg_Venue_Price']=manhattan_detail['Avg_Venue_Price'].round()
```

```
manhattan_detail['Avg_Venue_Rating']=manhattan_detail['Avg_Venue_Rating'].astype(int)
manhattan_detail['Avg_Venue_Price']=manhattan_detail['Avg_Venue_Price'].astype(int)
```

Figure 20: Grouped and aggregated Dataframe

```
manhattan_detail
```

Neighborhood	Number_UserMatched_Venues	Avg_Venue_Rating	Avg_Venue_Price
Battery Park City	1	7	2
Carnegie Hill	1	0	2
Chelsea	1	0	1
Chinatown	5	1	2
Civic Center	6	4	2
Clinton	7	3	2
East Village	7	1	1
Financial District	7	4	2
Flatiron	3	0	2
Gramercy	2	8	1
Greenwich Village	5	3	2
Hamilton Heights	1	0	2
Hudson Yards	3	0	1
Inwood	1	0	2
Lenox Hill	2	0	1
Little Italy	5	2	2
Lower East Side	2	4	2
Manhattan Valley	1	0	1
Manhattanville	1	0	2
Midtown	12	6	2
Midtown South	5	2	2
Morningside Heights	2	0	2
Murray Hill	5	2	2
Noho	5	5	2
Soho	7	2	2
Sutton Place	5	1	2
Tudor City	1	7	4
Turtle Bay	5	6	3
Upper East Side	1	8	4
Upper West Side	4	5	2
Washington Heights	1	8	2
West Village	4	6	2
Yorkville	7	3	2

Figure 21: Resulting Dataframe after grouping and aggregating

- The cleaned and grouped Dataframe is merged with the venues Dataframe that we got before which contains non-food related venues (hot spots) in each neighborhood. Resulting Dataframe is shown in Figure 22.

	Neighborhood	Nearby_Venue_Primary_Category	Number_UserMatched_Venues	Avg_Venue_Rating	Avg_Venue_Price
0	Battery Park City	Arts & Entertainment,Nightlife Spot,Outdoors &...	1	7	2
1	Carnegie Hill	Arts & Entertainment,Nightlife Spot,Outdoors &...	1	0	2
2	Chelsea	Arts & Entertainment,College & University,Nigh...	1	0	1
3	Chinatown	Arts & Entertainment,Nightlife Spot,Outdoors &...	5	1	2
4	Civic Center	Arts & Entertainment,Nightlife Spot,Outdoors &...	6	4	2
5	Clinton	Arts & Entertainment,Nightlife Spot,Outdoors &...	7	3	2
6	East Village	Arts & Entertainment,Nightlife Spot,Outdoors &...	7	1	1
7	Financial District	Arts & Entertainment,Nightlife Spot,Outdoors &...	7	4	2
8	Flatiron	Arts & Entertainment,Nightlife Spot,Outdoors &...	3	0	2
9	Gramercy	Arts & Entertainment,Nightlife Spot,Outdoors &...	2	8	1
10	Greenwich Village	Arts & Entertainment,Nightlife Spot,Outdoors &...	5	3	2
11	Hamilton Heights	Arts & Entertainment,Nightlife Spot,Outdoors &...	1	0	2
12	Hudson Yards	Arts & Entertainment,Nightlife Spot,Outdoors &...	3	0	1
13	Inwood	Arts & Entertainment,Nightlife Spot,Outdoors &...	1	0	2
14	Lenox Hill	Arts & Entertainment,College & University,Nigh...	2	0	1
15	Little Italy	Arts & Entertainment,Nightlife Spot,Outdoors &...	5	2	2
16	Lower East Side	Arts & Entertainment,Nightlife Spot,Outdoors &...	2	4	2
17	Manhattan Valley	Nightlife Spot,Outdoors & Recreation,Shop & Se...	1	0	1
18	Manhattanville	Arts & Entertainment,Nightlife Spot,Outdoors &...	1	0	2
19	Midtown	Arts & Entertainment,Nightlife Spot,Outdoors &...	12	6	2
20	Midtown South	Arts & Entertainment,Nightlife Spot,Outdoors &...	5	2	2
21	Morningside Heights	Arts & Entertainment,College & University,Nigh...	2	0	2
22	Murray Hill	Arts & Entertainment,Nightlife Spot,Outdoors &...	5	2	2
23	Noho	Arts & Entertainment,Nightlife Spot,Outdoors &...	5	5	2
24	Soho	Arts & Entertainment,Nightlife Spot,Outdoors &...	7	2	2
25	Sutton Place	Nightlife Spot,Outdoors & Recreation,Shop & Se...	5	2	2
26	Tudor City	Nightlife Spot,Outdoors & Recreation,Professio...	1	7	4
27	Turtle Bay	Arts & Entertainment,Nightlife Spot,Outdoors &...	5	6	3
28	Upper East Side	Arts & Entertainment,Nightlife Spot,Outdoors &...	1	8	4
29	Upper West Side	Arts & Entertainment,Nightlife Spot,Outdoors &...	4	5	2
30	Washington Heights	Nightlife Spot,Outdoors & Recreation,Shop & Se...	1	8	2
31	West Village	Arts & Entertainment,Nightlife Spot,Outdoors &...	4	6	2
32	Yorkville	Nightlife Spot,Outdoors & Recreation,Professio...	7	3	2

Figure 22: Italian food venue Dataframe merged with the popular venues Dataframe

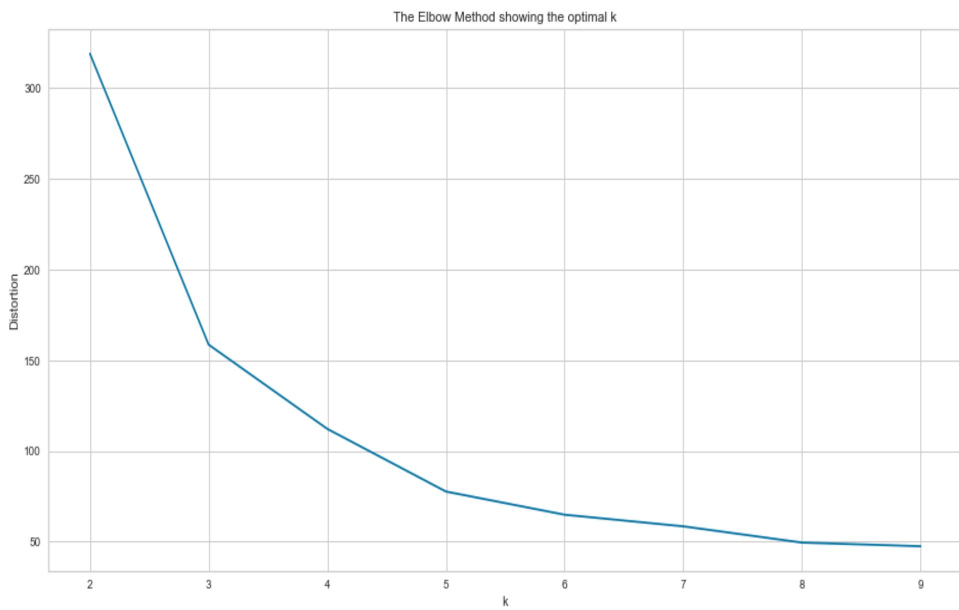
- Since the clustering algorithms do not support string type, one hot encoding is used to convert the popular venues categories from string to a type the clustering algorithm understands. The final resulting Dataframe before inputting it into the clustering algorithm is shown in Figure 23.

	Neighborhood	Number_UserMatched_Venues	Avg_Venue_Rating	Avg_Venue_Price	Nearby_Categroy_Arts & Entertainment,College & University,Nightlife Spot,Outdoors & Recreation,Professional & Other Places,Shop & Service	Nearby_Categroy_Arts & Entertainment,College & University,Nightlife Spot,Outdoors & Recreation,Professional & Other Places,Shop & Service,Travel & Transport	Nearby_Categroy_Arts & Entertainment,College & University,Nightlife Spot,Outdoors & Recreation,Shop & Service	Nearby_Categroy_Arts & Entertainment,Nightlife Spot,Outdoors & Recreation,Professional & Other Places,Residence,Shop & Service,Travel & Transport	Nearby_Categroy_Arts & Entertainment,Nightlife Spot,Outdoors & Recreation,Professional & Other Places,Shop & Service	Nearby_Categroy_Arts & Entertainment,Nightlife Spot,Outdoors & Recreation,Professional & Other Places,Shop & Service,Travel & Transport	Nearby_Categroy_Arts & Entertainment,Nightlife Spot,Outdoors & Recreation,Shop & Service	Nearby_Categroy_Arts & Entertainment,Nightlife Spot,Outdoors & Recreation,Shop & Service,Travel & Transport	Nearby_Categroy_Nightlife Spot,Outdoors & Recreation,Professional & Other Places,Shop & Service	Nearby_Categroy_Nightlife Spot,Outdoors & Recreation,Professional & Other Places,Shop & Service,Travel & Transport	Nearby_Categroy_Nightlife Spot,Outdoors & Recreation,Shop & Service,Travel & Transport
0	Battery Park City	1	7	2	0	0	0	0	0	1	0	0	0	0	0
1	Carnegie Hill	1	0	2	0	0	0	0	0	1	0	0	0	0	0
2	Chelsea	1	0	1	0	1	0	0	0	0	0	0	0	0	0
3	Chinatown	5	1	2	0	0	0	0	0	0	0	1	0	0	0
4	Civic Center	6	4	2	0	0	0	0	0	1	0	0	0	0	0
5	Clinton	7	3	2	0	0	0	1	0	0	0	0	0	0	0
6	East Village	7	1	1	0	0	0	0	0	0	1	0	0	0	0
7	Financial District	7	4	2	0	0	0	0	0	1	0	0	0	0	0
8	Flatiron	3	0	2	0	0	0	0	1	0	0	0	0	0	0
9	Gramercy	2	8	1	0	0	0	0	0	1	0	0	0	0	0
10	Greenwich Village	5	4	2	0	0	0	0	0	0	0	1	0	0	0
11	Hamilton Heights	1	0	2	0	0	0	0	1	0	0	0	0	0	0
12	Hudson Yards	3	0	1	0	0	0	1	0	0	0	0	0	0	0
13	Inwood	1	0	2	0	0	0	0	0	1	0	0	0	0	0
14	Lenox Hill	2	0	1	1	0	0	0	0	0	0	0	0	0	0
15	Little Italy	5	2	2	0	0	0	0	0	0	0	1	0	0	0
16	Lower East Side	2	4	2	0	0	0	0	0	0	1	0	0	0	0
17	Manhattan Valley	1	0	1	0	0	0	0	0	0	0	0	0	0	1
18	Manhattanville	1	0	2	0	0	0	0	0	0	0	1	0	0	0
19	Midtown	12	6	2	0	0	0	0	0	0	0	1	0	0	0
20	Midtown South	5	2	2	0	0	0	0	0	1	0	0	0	0	0
21	Morningside Heights	2	0	2	0	0	1	0	0	0	0	0	0	0	0
22	Murray Hill	5	2	2	0	0	0	0	0	1	0	0	0	0	0
23	Noho	5	5	2	0	0	0	0	0	1	0	0	0	0	0
24	Soho	7	2	2	0	0	0	0	0	0	0	1	0	0	0
25	Sutton Place	5	1	2	0	0	0	0	0	0	0	0	0	0	1
26	Tudor City	1	7	4	0	0	0	0	0	0	0	0	0	1	0
27	Turtle Bay	5	6	3	0	0	0	1	0	0	0	0	0	0	0
28	Upper East Side	1	8	4	0	0	0	0	0	0	0	1	0	0	0
29	Upper West Side	4	5	2	0	0	0	0	0	1	0	0	0	0	0
30	Washington Heights	1	8	2	0	0	0	0	0	0	0	0	0	0	1
31	West Village	4	6	2	0	0	0	0	0	0	0	1	0	0	0
32	Yorkville	7	3	2	0	0	0	0	0	0	0	0	1	0	0

Figure 23: Hot encoding to convert popular venues categories from categorical type to a type clustering algorithm understands

- Clustering algorithm chosen for this project is the K-means clustering algorithm. To know the optimal number of clusters, the elbow method and Silhouette score were used as shown in Figure 24,25 and 26. (For the Italian cuisine example)

```
plt.figure(figsize=(16,8))
plt.plot(K, distortions, 'bx-')
plt.xlabel('k')
plt.ylabel('Distortion')
plt.title('The Elbow Method showing the optimal k')
plt.show()
```




```
kl = KneeLocator( range(2, 10), distortions, curve="convex", direction="decreasing")
print("The optimal number of clusters using elbow method is",kl.elbow)
```

```
The optimal number of clusters using elbow method is 4
```

Figure 24: Elbow method to find the optimal number of clusters

```
for i in range(2,10):
    print("Silhouette Score for",i," Clusters is ",score[i-2])
n=score.index(max(score))
numcluster=n+2
print (numcluster," Clusters has the best Silhouette Score")
```

```
Silhouette Score for 2  Clusters is  0.35302333653231766
Silhouette Score for 3  Clusters is  0.45879696085900395
Silhouette Score for 4  Clusters is  0.4630761462248538
Silhouette Score for 5  Clusters is  0.4392770222756117
Silhouette Score for 6  Clusters is  0.3952706441895223
Silhouette Score for 7  Clusters is  0.31545225122648246
Silhouette Score for 8  Clusters is  0.2987217630023837
Silhouette Score for 9  Clusters is  0.2638115877219031
4  Clusters has the best Silhouette Score
```

Figure 25: Silhouette score to find the optimal number of clusters

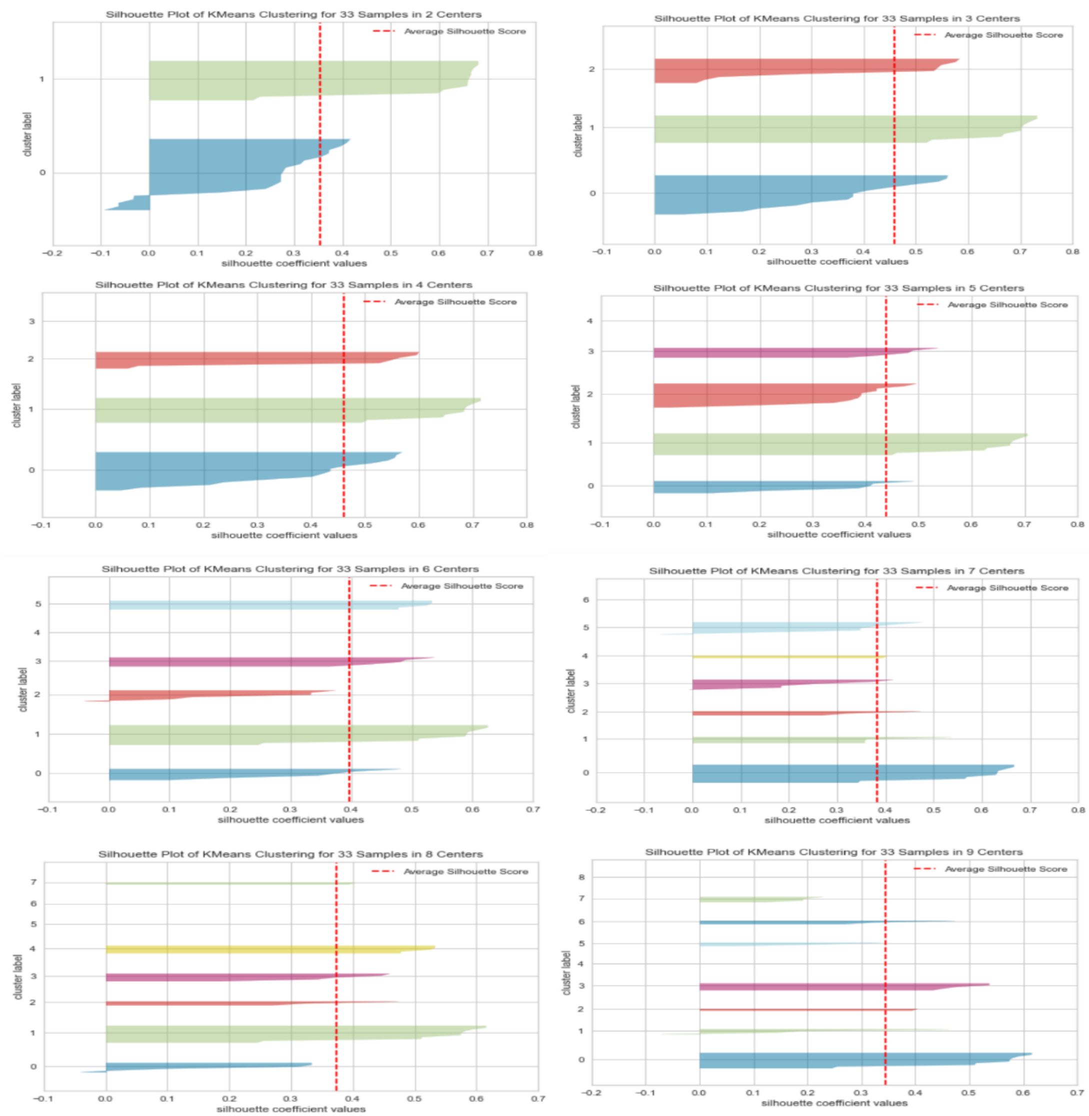


Figure 26: Comparing Silhouette score for different number of clusters

- As noticed from the graphs above, 2 Clusters, 3 Clusters, 4 Clusters, 5 Clusters and 7 Clusters all have Silhouette scores above average Silhouette score which make them candidate for the optimal number of clusters. However, fluctuation in size (thickness) of the silhouette plot representing each cluster also is a deciding point. So, 2 Clusters, 3 Clusters, 5 Clusters and 7 Clusters have more fluctuation in size as compared to 4 Clusters. For the plot with 4 Clusters, the thickness is more uniform than the plot with 2 Clusters, 3 Clusters, 5 Clusters and 7 Clusters. Thus, one can select the optimal number of Clusters as 4.
- 6 Clusters, 8 Clusters and 9 Clusters are not considered optimal number of Clusters because of the presence of Clusters with below-average silhouette scores and the fluctuations in the size of the silhouette plots

- After that, the K-means clustering algorithm was applied with 4 clusters for the Italian cuisine example. Output is shown in the results section below. (Figure 27)

```
kclusters = numcluster

Manhattan_grouped_clustering = df_manhattan.drop('Neighborhood', 1)

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

# check cluster labels generated for each row in the dataframe
kmeans.labels_

array([2, 1, 1, 0, 0, 0, 0, 0, 1, 2, 0, 1, 1, 1, 1, 0, 2, 1, 1, 3, 0, 1,
       0, 0, 0, 0, 2, 0, 2, 0, 2, 2, 0])
```

Figure 27: Applying k-mean clustering algorithm with 3 clusters

Results

- After applying the K-means algorithm, the cluster labels were added to the DataFrame. Also, the Dataframe was cleaned as follows:
    - Some Neighborhoods have 0 number of venues that are related to the user's cuisine of interest...in this case there are neighborhoods that do not have any italian cuisines. These Neighborhoods are shown in the table above as NaN and were added to a new cluster. (Cluster #5)
    - "Nearby_Venue_Ptimary_Category" column unique entries were mapped to numbers so that it becomes easier to analyze
    - Avg Rating and Avg Price type were changed to integers
    - Venues with Avg Rating and Avg Price of 0, have no rating and no price range. So 0 values were changed to "No rating" and "No Price range"
- The final Dataframe showing different clusters is seen in Figure 28. Then the clusters are visualized on Folium Map with different colors as shown in Figure 29 where if you pressed a certain mark it will give you the neighborhood and cluster number. Also, each cluster can be analyzed using separate Dataframes and charts. (Figures 30 and 31)

Neighborhood	Latitude	Longitude	Cluster Labels	Nearby_Venue_Primary_Category	Number_UserMatched_Venues	Avg_Venue_Rating	Avg_Venue_Price
Marble Hill	40.876551	-73.910660	4	0	0	0	0
Chinatown	40.715618	-73.994279	0	1	5	1	2
Washington Heights	40.851903	-73.936900	2	2	1	8	2
Inwood	40.867684	-73.921210	1	3	1	0	2
Hamilton Heights	40.823604	-73.949688	1	4	1	0	2
Manhattanville	40.816934	-73.957385	1	1	1	0	2
Central Harlem	40.815976	-73.943211	4	0	0	0	0
East Harlem	40.792249	-73.944182	4	0	0	0	0
Upper East Side	40.775639	-73.960508	2	1	1	8	4
Yorkville	40.775930	-73.947118	0	5	7	3	2
Lenox Hill	40.768113	-73.958860	1	6	2	0	1
Roosevelt Island	40.762160	-73.949168	4	0	0	0	0
Upper West Side	40.787658	-73.977059	0	3	4	5	2
Lincoln Square	40.773529	-73.985338	4	0	0	0	0
Clinton	40.759101	-73.996119	0	7	7	3	2
Midtown	40.754691	-73.981669	3	1	12	6	2
Murray Hill	40.748303	-73.978332	0	3	5	2	2
Chelsea	40.744035	-74.003116	1	8	1	0	1
Greenwich Village	40.726933	-73.999914	0	1	5	4	2
East Village	40.727847	-73.982226	0	9	7	1	1
Lower East Side	40.717807	-73.980890	2	9	2	4	2
Tribeca	40.721522	-74.010683	4	0	0	0	0
Little Italy	40.719324	-73.997305	0	1	5	2	2
Soho	40.722184	-74.000657	0	1	7	2	2
West Village	40.734434	-74.006180	2	1	4	6	2
Manhattan Valley	40.797307	-73.964286	1	2	1	0	1
Morningside Heights	40.808000	-73.963896	1	10	2	0	2
Gramercy	40.737210	-73.981376	2	3	2	8	1
Battery Park City	40.711932	-74.016869	2	3	1	7	2
Financial District	40.707107	-74.010665	0	3	7	4	2
Carnegie Hill	40.782683	-73.953256	1	3	1	0	2
Noho	40.723259	-73.988434	0	3	5	5	2
Civic Center	40.715229	-74.005415	0	3	6	4	2
Midtown South	40.748510	-73.988713	0	3	5	2	2
Sutton Place	40.760280	-73.963556	0	2	5	1	2
Turtle Bay	40.752042	-73.967708	0	7	5	6	3
Tudor City	40.746917	-73.971219	2	11	1	7	4
Stuyvesant Town	40.731000	-73.974052	4	0	0	0	0
Flatiron	40.739673	-73.990947	1	4	3	0	2
Hudson Yards	40.756658	-74.000111	1	7	3	0	1

Figure 28: Resulting Dataframe

```python
map_clusters = folium.Map(location=[latitude, longitude], zoom_start=11)

# set color scheme for the clusters
x = np.arange(kclusters+1)
ys = [i + x + (i*x)**2 for i in range(kclusters+1)]
colors_array = cm.rainbow(np.linspace(0, 1, len(ys)))

rainbow = [colors.rgb2hex(i) for i in colors_array]

# add markers to the map
markers_colors = []
for lat, lon, poi, cluster in zip(manhattan_merged['Latitude'], manhattan_merged['Longitude'], manhattan_merged['Neighborhoo
    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
```

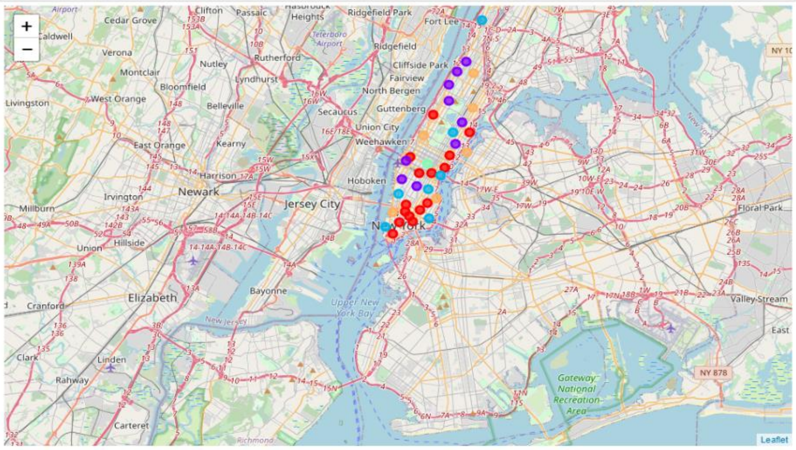


Figure 29: Show different clusters on map

Cluster 1

	Neighborhood	Nearby_Venue_Primary_Category	Number_UserMatched_Venues	Avg_Venue_Rating	Avg_Venue_Price
1	Chinatown	1	5	1	2
9	Yorkville	5	7	3	2
12	Upper West Side	3	4	5	2
14	Clinton	7	7	3	2
16	Murray Hill	3	5	2	2
18	Greenwich Village	1	5	4	2
19	East Village	9	7	1	1
22	Little Italy	1	5	2	2
23	Soho	1	7	2	2
29	Financial District	3	7	4	2
31	Noho	3	5	5	2
32	Civic Center	3	6	4	2
33	Midtown South	3	5	2	2
34	Sutton Place	2	5	1	2
35	Turtle Bay	7	5	6	3

Cluster 2

	Neighborhood	Nearby_Venue_Primary_Category	Number_UserMatched_Venues	Avg_Venue_Rating	Avg_Venue_Price
3	Inwood	3	1	0	2
4	Hamilton Heights	4	1	0	2
5	Manhattanville	1	1	0	2
10	Lenox Hill	6	2	0	1
17	Chelsea	8	1	0	1
25	Manhattan Valley	2	1	0	1
26	Morningside Heights	10	2	0	2
30	Carnegie Hill	3	1	0	2
38	Flatiron	4	3	0	2
39	Hudson Yards	7	3	0	1

Cluster 3

	Neighborhood	Nearby_Venue_Primary_Category	Number_UserMatched_Venues	Avg_Venue_Rating	Avg_Venue_Price
2	Washington Heights	2	1	8	2
8	Upper East Side	1	1	8	4
20	Lower East Side	9	2	4	2
24	West Village	1	4	6	2
27	Gramercy	3	2	8	1
28	Battery Park City	3	1	7	2
36	Tudor City	11	1	7	4

Cluster 4

	Neighborhood	Nearby_Venue_Primary_Category	Number_UserMatched_Venues	Avg_Venue_Rating	Avg_Venue_Price
15	Midtown	1	12	6	2

Cluster 5

	Neighborhood	Nearby_Venue_Primary_Category	Number_UserMatched_Venues	Avg_Venue_Rating	Avg_Venue_Price
0	Marble Hill	0	0	0	0
6	Central Harlem	0	0	0	0
7	East Harlem	0	0	0	0
11	Roosevelt Island	0	0	0	0
13	Lincoln Square	0	0	0	0
21	Tribeca	0	0	0	0
37	Stuyvesant Town	0	0	0	0

Figure 30: Dataframe for each cluster (going from top left to bottom right)

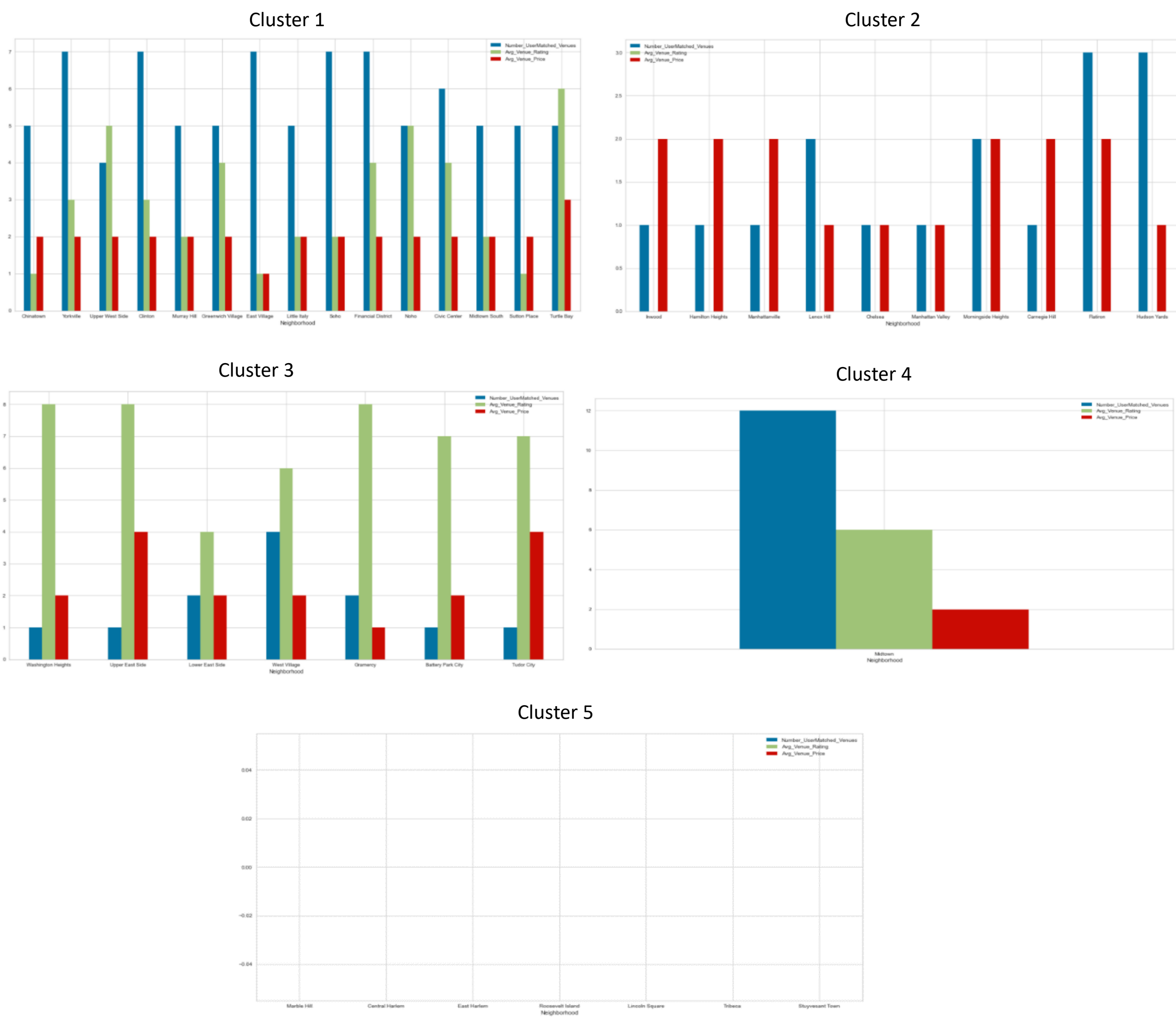


Figure 31: Compare different clusters visually

Discussion

- As noticed from the map, cluster tables and charts, Neighborhoods ['Marble Hill', 'Central Harlem', 'East Harlem', 'Roosevelt Island', 'Lincoln Square', 'Tribeca', 'Stuyvesant Town'] have 0 number of Italian cuisines. These Neighborhoods were grouped into a new cluster which is Cluster #5.
- Clusters 1-4 are identified according to the number venues matched to the user's entry in the neighborhood, their average rating, average price and their surrounding popular spots.
- However, it is noticed that the popular spots in each Neighborhoods are kind of similar in each cluster.

- As seen in charts above:
    - Cluster #1 grouped Neighborhoods that have high number of Italian cuisine with Moderate price range and average rating
    - Cluster #2 grouped Neighborhoods that have low number of Italian cuisine with Moderate price and no rating
    - Cluster #3 grouped Neighborhoods that have low number of Italian cuisine with above Moderate price and high rating
    - Cluster #4 grouped Neighborhoods that have very high number of Italian cuisines with Moderate price and Moderate rating
    - Cluster #5 grouped Neighborhoods that have 0 number of Italian cuisines
- The choice of which cluster to open up the Italian cuisine depends on the user's target for the Italian restaurant. So if the user is thinking of opening up a luxurious Italian restaurant then the price range will be above Moderate and so Cluster #2 or #5 might be good candidates.
- However, there should be more information on the neighborhoods such as demographics, social and economic characteristics of the people which will help to know people's interests and tendencies. So, there will be more insight and in this way it will be known, for example, if there is demand for Italian cuisine in Cluster #5 or not.

Conclusion

New York is the core of the best restaurants globally with various cuisines and international culinary experts. So, opening up a food business in Manhattan may be a hard decision as there is a strong of competition. However, with the help of an algorithm, identifying the best location for food business will be much simpler. This project is aimed to help food business seekers to decide which Neighborhood in Manhattan is best suited for their business. However, only two data sources were used and in my opinion there is room for improvement with the use of additional data sources. For example, the use of Foursquare Places Databases will add more features regarding the venues that are not available in the Foursquare Places AP such as service quality, whether the food venue is crowded, whether food is worth the price and whether the food venue is trendy. Also, another data source that will enrich this project is data about the demographic, social and economic characteristics of the people in Manhattan which would've helped to analyze and understand people's interests and tendencies. Adding relevant data will improve the clustering of Neighborhoods and enhance the identification of the best Neighborhood for the food business seeker.