CapStone1

October 11, 2019

1 The Battle of Neighborhoods Part II

1.0.1 IBM Capstone Project

1.0.2 By: Benyam Tadesse

1.0.3 Table of Contents

1.1 1. Introduction

An investor wants to expand his Bakery-Coffee shop in New York City but the investor isn't from New York City and doesn't know much about the different boroughs and/or neighborhoods in the city. In knowing this we will provide the relevant data for the investor, which will mostly be neighborhood venue data that lists the least amount of coffee shops and/or bakeries. We will also want to figure out which borough has the least amount of crime data reported in that specific borough which will likely have an affect on the demographics, this information will also likely help with real estate pricing and also give potential customers a better vibe to make them feel safer but also keep potential competitors at bay.

in this report we're trying to give a simplistic understanding of the different boroughs crime rates in New York City, so that the investor can get a better understanding of the structures (demographics) of the boroughs. We will also explore certain neighborhoods from that preferred borough, we will leverage the Foursquare API for this part of the task. In layman terms we will answer which borough is better or worse in terms of crime data, after which figure out which neighborhoods are preferred in that specific borough and a list of most common venues in these neighborhoods using the Foursquare API.

1.2 2. Data Description

We will be using three different datasets & the Foursquare API in this report. The first dataset we will be analyzing is the New York City crime data. This data is comprised of crime reports from all 5 boroughs in fiscal year ending in 2018. The dataset contains longitude and latitude of where the crime occurred, it also includes various other information but we will only be using the location data. This data was accessible from the City of New York public API found here: - https://data.cityofnewyork.us/resource/qgea-i56i.json.

```
df1 =  df['Borough'].value_counts()
df1
```

- Borough – Count
- Brooklyn – 293
- Manhattan – 251
- Bronx – 224
- Queens – 191
- Staten Island – 39 Name: Borough, dtype: int64

The second dataset we will be looking at is New York City borough border map Geojson dataset, this map will be used to create a Choropleth map. A Choropleth map is a thematic map in which areas are shaded in proportion to the measurement of the statistical variable being displayed on the map, the variable data will be the crime data collected. This dataset can be downloaded from the City of New York site found here: - https://data.cityofnewyork.us/widgets/tqmj-j8zm

The third Dataset we will be using is the New York City neighborhoods geographical coordinates, we will learn that New York City has 5 boroughs and 306 neighborhoods. Thus the reason for us to figure out which borough we want to focus our analyses on. The New York City neighborhoods geographical coordinates data will be utilized using Foursquare API. this Dataset was provided by NYU and can be downloaded from this site: - https://geo.nyu.edu/catalog/nyu_2451_34572.

Finally we will be accessing the Foursquare call API to to get venue location data.

1.3 3. Methodology Section

The main component of this report will consist of performing a exploratory data analysis (EDA) on the New York City crime data while including the Geojson data of the 5 boroughs to superimpose the combined data into a Choropleth Folium map. After which we will utilize the foursquare API to get venue data on the borough selected after performing EDA, this will help us figure out which neighborhood has what type of venues in the area for potential investors.

We will begin with the New York City crime data by finding unique values of the number of times a crime is committed in each borough, by using the **.value_count()** pandas function. Doing this will give use a clear picture of the 5 boroughs, after which we will utilize this informations in our *Choropleth* map. A *Choropleth* map is a thematic map in which areas are shaded in proportion to the measurement of the statistical variable being displayed on the map, such as the **.value_count()** data we gathered. The *Choropleth* map will provide an essential way to visualize how the measurement varies across the 5 boroughs.

after accomplishing the *Choropleth* map wee will begin to segment & cluster the neighborhoods in Queens, we will also get the location data by using *Geopy* library. after gathering the location data we utilization the Foursquare API to explore and segment the neighborhoods in Queens. segmenting is division into separate parts or sections there are 4 main types of segmentation Geographic, Demographic, Psychographic, & Behavioral we will be using geographic segmentation in this report. Extracting venues category data from all the neighborhoods in Queens is the next method we will execute, also we will print out the top 5 most common venues in that neighborhood. Finally we reach the *k*-means algorithm part of the process, *k*-means algorithm is an unsupervised/ partitioning clustering algorithm that: - 1. Cluster the data into k groups where k is predefined - 2. Select k points at random as cluster centers. - 3. Assign objects to their closed cluster center according to the *Euclidean distance* function. - 4. Calculate the centroid or mean of all objects in each cluster. - 5. Final step is to repeat the steps until the same points are assigned to each cluster in consecutive rounds.

k-means divides the data into non-overlapping subsets (clusters) without any cluster-internal structure, the objective of k-means is to form clusters in such a way that similar samples go into a cluster, in our case common venue categories are put in the same cluster. We can reach this by following these steps: - Cluster Neighborhoods - We run k-means to cluster the neighborhood into 5 clusters. - create a new dataframe that includes the cluster as well as the top 10 venues for each neighborhood. - visualize the resulting clusters - Examine Clusters - Now we examine each cluster and determine the discriminating venue categories that distinguish each cluster.

1.4 4. Results

We begin by importing the libraries we'll use

[48]:
```
import numpy as np # library to handle data in a vectorized manner
import pandas as pd # library for data analsysis
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
import json # library to handle JSON files
#!conda install -c conda-forge geopy --yes
from geopy.geocoders import Nominatim # convert an address into latitude and␣
 ↪longitude values
import requests # library to handle requests
from pandas.io.json import json_normalize # tranform JSON file into a pandas␣
 ↪dataframe
# Matplotlib and associated plotting modules
import matplotlib.cm as cm
import matplotlib.colors as colors
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
print('Libraries imported.')
```

```
Libraries imported.
```

Now we load the NY Crime data as a *json* data file

[49]:
```
with open('ny_crime.json') as json_data:
    ny_crime = json.load(json_data)
```

next step is to convert the *json* data into a *pandas* Data Frame

[50]:
```
df=pd.DataFrame(ny_crime)
```

now we get the number of crimes committed in each Borough using *.value_counts*.

[51]:
```
df.rename(columns={'boro_nm':'Borough'}, inplace=True)
df1 =  df['Borough'].value_counts()
df1
```

[51]:
```
BROOKLYN         293
MANHATTAN        251
BRONX            224
QUEENS           191
STATEN ISLAND     39
Name: Borough, dtype: int64
```

now we create a new *pandas DataFrame* with the data we gathered.

[52]:
```
data= {'Borough': ['Brooklyn', 'Manhattan', 'Bronx', 'Queens', 'Staten Island'],
      'Count': [293, 251, 224, 191, 39]}
df2=pd.DataFrame(data)
df2.head()
```

[52]:

	Borough	Count
0	Brooklyn	293
1	Manhattan	251
2	Bronx	224
3	Queens	191
4	Staten Island	39

Here we use geopy library to get the latitude and longitude values of New York City.

[54]:
```
address = 'New York City, NY'
geolocator = Nominatim(user_agent="ny_explorer")
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude
print('The geograpical coordinate of New York City are {}, {}.'.
 ↪format(latitude, longitude))
```

```
The geograpical coordinate of New York City are 40.7127281, -74.0060152.
```

let's create a choropleth map of the New York City Boroughs, centered around [**40.7127281, -74.0060152**] *latitude* and *longitude* values, with an intial zoom level of 11, and using *Mapbox Bright* style.

[55]:
```
# download the cities geojson file
ny_geo = r'NY_Borough_geo.json'

ny_map = folium.Map(location=[40.7127281, -74.0060152], zoom_start=11,␣
 ↪tiles='Mapbox Bright')

print('GeoJSON file downloaded!')
```

```
GeoJSON file downloaded!
```

we will use the *choropleth* method with the following main parameters: 1. geo_data, which is the GeoJSON file. 2. data, which is the dataframe containing the data. 3. columns, which represents the columns in the dataframe that will be used to create the `Choropleth` map. 4. key_on, which is the key or variable in the GeoJSON file that contains the name of the variable of interest. To determine that, you will need to open the GeoJSON file using any text editor and note the name of the key or variable that contains the name of the countries, since the countries are our variable of interest. In this case, **name** is the key in the GeoJSON file that contains the name of the countries. Note that this key is case_sensitive, so you need to pass exactly as it exists in the GeoJSON file. 5. Last we'll define our own thresholds and starting & ending with Count min & max

[56]:
```
threshold_scale = np.linspace(df2['Count'].min(),
                              df2['Count'].max(),
                              6, dtype=int)
threshold_scale = threshold_scale.tolist() # change the numpy array to a list
threshold_scale[-1] = threshold_scale[-1] + 1
# make sure that the last value of the list is greater than the maximum␣
 ↪immigration
ny_map = folium.Map(location=[40.7127281, -74.0060152], zoom_start=10)
ny_map.choropleth(
    geo_data=ny_geo,
    data=df2,
    columns=['Borough', 'Count'],
    key_on='feature.properties.boro_name',
    threshold_scale=threshold_scale,
    fill_color='YlOrRd',
    fill_opacity=0.7,
    line_opacity=0.3,
    legend_name='Crime in New York City ',
    reset=True
)
# display map
ny_map
```

[56]:
```
<folium.folium.Map at 0x7f5ffa55c160>
```

1.5 FourSquare API utilization/ Clustering

Load and explore the data

[11]:
```
with open('newyork_data.json') as json_data:
    newyork_data = json.load(json_data)
```

[12]:
```
neighborhoods_data = newyork_data['features']
```

Tranform the data into a *pandas* Dataframe The next task is essentially transforming this data of nested Python dictionaries into a *pandas* dataframe.

```
[13]: # define the dataframe columns
column_names = ['Neighborhood', 'Latitude', 'Longitude']
# instantiate the dataframe
neighborhoods = pd.DataFrame(columns=column_names)
```

let's now loop through the data and fill the dataframe one row at a time.

```
[14]: for data in neighborhoods_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]
    neighborhoods = neighborhoods.append({'Boro': borough,
                                          'Neighborhood': neighborhood_name,
                                          'Latitude': neighborhood_lat,
                                          'Longitude': neighborhood_lon},␣
 ↪ignore_index=True)
```

let's segment and cluster only the neighborhoods in Queens. So let's slice the original dataframe and create a new dataframe of the Queens data.

```
[15]: Queens_data = neighborhoods[neighborhoods['Boro'] == 'Queens'].
 ↪reset_index(drop=True)
Queens_data.head()
```

[15]:

	Neighborhood	Latitude	Longitude	Boro
0	Astoria	40.768509	-73.915654	Queens
1	Woodside	40.746349	-73.901842	Queens
2	Jackson Heights	40.751981	-73.882821	Queens
3	Elmhurst	40.744049	-73.881656	Queens
4	Howard Beach	40.654225	-73.838138	Queens

Here we use geopy library to get the latitude and longitude values of Queens.

```
[16]: address = 'Queens, NY'

geolocator = Nominatim(user_agent="ny_explorer")
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude
print('The geograpical coordinate of Queens are {}, {}.'.format(latitude,␣
 ↪longitude))
```

```
The geograpical coordinate of Queens are 40.6524927, -73.7914214158161.
```

Now we create map of Queens, neighborhoods using latitude and longitude values

```
[17]: map_queens = folium.Map(location=[latitude, longitude], zoom_start=11)
# add markers to map
for lat, lng, label in zip(Queens_data['Latitude'], Queens_data['Longitude'],␣
↪Queens_data['Neighborhood']):
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius=5,
        popup=label,
        color='red',
        fill=True,
        fill_color='#3186cc',
        fill_opacity=0.7,
        parse_html=False).add_to(map_queens)
map_queens
```

```
[17]: <folium.folium.Map at 0x7f600a29b6d8>
```

1.5.1 Define Foursquare Credentials and Version

we are going to start utilizing the Foursquare API to explore the neighborhoods and segment them.

```
[18]: CLIENT_ID = '2RB3EBPU40AQE5YLHJSQZMLTJTTTZRT0XXK1RRSIJQ4XW0A1' # your␣
↪Foursquare ID
CLIENT_SECRET = 'Z0NAH1VT2W5QTZIONK03TZ5YOUPSIH2VJPZ5ARS105YWZZVW' # your␣
↪Foursquare Secret
VERSION = '20180605' # Foursquare API version
```

let's get the queens neighborhood's latitude and longitude values.

```
[19]: neighborhood_latitude = Queens_data.loc[0, 'Latitude'] # neighborhood latitude␣
↪value
neighborhood_longitude = Queens_data.loc[0, 'Longitude'] # neighborhood␣
↪longitude value
neighborhood_name = Queens_data.loc[0, 'Neighborhood'] # neighborhood name

print('Latitude and longitude values of {} are {}, {}.'.
↪format(neighborhood_name,
                                                                   ␣
↪neighborhood_latitude,
                                                                   ␣
↪neighborhood_longitude))
```

```
Latitude and longitude values of Astoria are 40.76850859335492,
-73.91565374304234.
```

Here we show the top 100 venues that are in Queens within a radius of 500 meters. let's create the GET request URL. Name your URL **url**.

[20]:
```
LIMIT = 100 # limit of number of venues returned by Foursquare API
radius = 500 # define radius
# create URL
url = 'https://api.foursquare.com/v2/venues/explore?
↪&client_id={}&client_secret={}&v={}&ll={},{}&radius={}&limit={}'.format(
    CLIENT_ID,
    CLIENT_SECRET,
    VERSION,
    neighborhood_latitude,
    neighborhood_longitude,
    radius,
    LIMIT)
```

Send the GET request and examine the resutls

[21]:
```
results = requests.get(url).json()
```

function that extracts the category of the venue

[22]:
```
def get_category_type(row):
    try:
        categories_list = row['categories']
    except:
        categories_list = row['venue.categories']

    if len(categories_list) == 0:
        return None
    else:
        return categories_list[0]['name']
```

Now we are ready to clean the json and structure it into a *pandas* dataframe.

[23]:
```
venues = results['response']['groups'][0]['items']

nearby_venues = json_normalize(venues) # flatten JSON

# filter columns
filtered_columns = ['venue.name', 'venue.categories', 'venue.location.lat',␣
↪'venue.location.lng']
nearby_venues =nearby_venues.loc[:, filtered_columns]

# filter the category for each row
nearby_venues['venue.categories'] = nearby_venues.apply(get_category_type,␣
↪axis=1)
```

```
# clean columns
nearby_venues.columns = [col.split(".")[-1] for col in nearby_venues.columns]

nearby_venues.head()
```

[23]:

	name	categories	lat	lng
0	Favela Grill	Brazilian Restaurant	40.767348	-73.917897
1	Orange Blossom	Gourmet Shop	40.769856	-73.917012
2	Titan Foods Inc.	Gourmet Shop	40.769198	-73.919253
3	CrossFit Queens	Gym	40.769404	-73.918977
4	Off The Hook	Seafood Restaurant	40.767200	-73.918104

1.5.2 Explore Neighborhoods in Queens

now we'll write a function to repeat the same process to all the neighborhoods in Queens.

[24]:
```
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={}'.format(
            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']['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',
                  'Venue',
```

```
                 'Venue Latitude',
                 'Venue Longitude',
                 'Venue Category']
    return(nearby_venues)
```

Now write the code to run the above function on each neighborhood and create a new dataframe called *Queens_venues*.

[ ]:
```
Queens_venues = getNearbyVenues(names=Queens_data['Neighborhood'],
                                   latitudes=Queens_data['Latitude'],
                                   longitudes=Queens_data['Longitude'])
```

[26]:
```
#Now we checj the size of the DataFrame
Queens_venues.shape
```

[26]: (2138, 7)

[28]:
```
print('There are {} uniques categories.'.format(len(Queens_venues['Venue␣
↪Category'].unique())))
```

```
There are 268 uniques categories.
```

1.6 Analyze Each Neighborhood

[30]:
```
# one hot encoding
onehot = pd.get_dummies(Queens_venues[['Venue Category']], prefix="",␣
↪prefix_sep="")
# add neighborhood column back to dataframe
onehot['Neighborhood'] = Queens_venues['Neighborhood']
# move neighborhood column to the first column
fixed_columns = [onehot.columns[-1]] + list(onehot.columns[:-1])
onehot = onehot[fixed_columns]
```

Let's see the size of the new DataFrame

[31]:
```
onehot.shape
```

[31]: (2138, 268)

Next, let's group rows by neighborhood and by taking the mean of the frequency of occurrence of each category

[32]:
```
grouped = onehot.groupby('Neighborhood').mean().reset_index()
grouped.shape
```

[32]: (81, 268)

Let's print each neighborhood along with the top 5 most common venues

[ ]:
```
num_top_venues = 5
for hood in grouped['Neighborhood']:
    print("----"+hood+"----")
    temp = grouped[grouped['Neighborhood'] == hood].T.reset_index()
    temp.columns = ['venue','freq']
    temp = temp.iloc[1:]
    temp['freq'] = temp['freq'].astype(float)
    temp = temp.round({'freq': 2})
     print(temp.sort_values('freq', ascending=False).reset_index(drop=True).
 ↪head(num_top_venues))
    print('\n')
```

Let's put that into a *pandas* dataframe let's write a function to sort the venues in descending order.

[34]:
```
def return_most_common_venues(row, num_top_venues):
    row_categories = row.iloc[1:]
    row_categories_sorted = row_categories.sort_values(ascending=False)

    return row_categories_sorted.index.values[0:num_top_venues]
```

Now let's create the new dataframe and display the top 10 venues for each neighborhood.

[57]:
```
num_top_venues = 10
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'] = grouped['Neighborhood']
for ind in np.arange(grouped.shape[0]):
    neighborhoods_venues_sorted.iloc[ind, 1:] =␣
 ↪return_most_common_venues(grouped.iloc[ind, :], num_top_venues)
neighborhoods_venues_sorted.head(1)
```

[57]:
```
  Neighborhood 1st Most Common Venue 2nd Most Common Venue  \
0      Arverne             Surf Spot         Metro Station

  3rd Most Common Venue 4th Most Common Venue 5th Most Common Venue  \
0        Sandwich Place              Bus Stop            Donut Shop

  6th Most Common Venue 7th Most Common Venue 8th Most Common Venue  \
```

```
0       Bed & Breakfast       Thai Restaurant                    Café

  9th Most Common Venue 10th Most Common Venue
0           Coffee Shop             Board Shop
```

1.6.1 Cluster Neighborhoods

We run *k*-means to cluster the neighborhood into 5 clusters.

[37]:
```
# set number of clusters
kclusters = 5

Queens_clustering = grouped.drop('Neighborhood', 1)

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

# check cluster labels generated for each row in the dataframe
kmeans.labels_[0:10]
```

[37]:
```
array([3, 3, 3, 3, 3, 3, 1, 3, 3, 0], dtype=int32)
```

Let's create a new dataframe that includes the cluster as well as the top 10 venues for each neighborhood.

[58]:
```
# add clustering labels
neighborhoods_venues_sorted.insert(0, 'Cluster Labels', kmeans.labels_)

Queens_merged = Queens_data

# merge toronto_grouped with toronto_data to add latitude/longitude for each␣
 ↪neighborhood
Queens_merged = Queens_merged.join(neighborhoods_venues_sorted.
 ↪set_index('Neighborhood'), on='Neighborhood')

Queens_merged.head(1) # check the last columns!
```

[58]:
```
  Neighborhood   Latitude  Longitude    Boro  Cluster Labels  \
0      Astoria  40.768509 -73.915654  Queens               3

  1st Most Common Venue 2nd Most Common Venue      3rd Most Common Venue  \
0                   Bar      Greek Restaurant  Middle Eastern Restaurant

  4th Most Common Venue 5th Most Common Venue 6th Most Common Venue  \
0            Hookah Bar    Seafood Restaurant                Bakery

      7th Most Common Venue 8th Most Common Venue 9th Most Common Venue  \
0  Mediterranean Restaurant                  Café    Japanese Restaurant
```

```
  10th Most Common Venue
0          Ice Cream Shop
```

let's visualize the resulting clusters

[40]:
```
# create map
map_clusters = folium.Map(location=[latitude, longitude], zoom_start=10)

# set color scheme for the clusters
x = np.arange(kclusters)
ys = [i + x + (i*x)**2 for i in range(kclusters)]
colors_array = cm.rainbow(np.linspace(0, 1, len(ys)))
rainbow = [colors.rgb2hex(i) for i in colors_array]

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

[40]: `<folium.folium.Map at 0x7f60001d2a58>`

1.6.2 Examine Clusters

Now we examine each cluster and determine the discriminating venue categories that distinguish each cluster.

Cluster 1

[41]:
```
Queens_merged.loc[Queens_merged['Cluster Labels'] == 0, Queens_merged.
↪columns[[0] + list(range(1, Queens_merged.shape[1]))]]
```

[41]:
```
      Neighborhood   Latitude  Longitude    Boro  Cluster Labels  \
43    Breezy Point  40.557401 -73.925512  Queens               0
49  Rockaway Beach  40.582802 -73.822361  Queens               0
```

```
61    Belle Harbor  40.576156 -73.854018  Queens               0
62   Rockaway Park  40.580343 -73.841534  Queens               0
75         Roxbury  40.567376 -73.892138  Queens               0
78         Hammels  40.587338 -73.805530  Queens               0

   1st Most Common Venue            2nd Most Common Venue  \
43                 Beach                       Board Shop
49                 Beach                   Ice Cream Shop
61                 Beach                              Spa
62                 Beach                       Donut Shop
75                 Beach                              Bar
78                 Beach  Southern / Soul Food Restaurant

        3rd Most Common Venue 4th Most Common Venue 5th Most Common Venue  \
43          Monument / Landmark                 Trail         Women's Store
49   Latin American Restaurant          Deli / Bodega             BBQ Joint
61                Deli / Bodega    Mexican Restaurant    Chinese Restaurant
62                  Pizza Place                  Bank            Bagel Shop
75         Fast Food Restaurant                   Pub         Deli / Bodega
78                        Diner              Bus Stop           Bus Station

   6th Most Common Venue        7th Most Common Venue 8th Most Common Venue  \
43           Event Space  Eastern European Restaurant   Egyptian Restaurant
49     Seafood Restaurant                  Food Truck      Arepa Restaurant
61            Donut Shop                      Bakery            Bagel Shop
62            Smoke Shop           Seafood Restaurant              Bus Stop
75        Baseball Field                   Irish Pub                 Trail
78               Dog Run                  Shoe Store  Gym / Fitness Center

   9th Most Common Venue 10th Most Common Venue
43      Electronics Store    Empanada Restaurant
49            Pizza Place          Moving Target
61     Italian Restaurant              Boutique
62             Board Shop      French Restaurant
75      Electronics Store            Dry Cleaner
78                   Café             Food Truck
```

Cluster 2

[42]:
```python
Queens_merged.loc[Queens_merged['Cluster Labels'] == 1, Queens_merged.
 ↪columns[[0] + list(range(1, Queens_merged.shape[1]))]]
```

[42]:
```
   Neighborhood   Latitude  Longitude    Boro  Cluster Labels  \
63   Somerville  40.597711 -73.796648  Queens               1
79    Bayswater  40.611322 -73.765968  Queens               1

   1st Most Common Venue 2nd Most Common Venue 3rd Most Common Venue  \
```

```
63                   Park         Women's Store   Empanada Restaurant
79              Playground                  Park         Women's Store

   4th Most Common Venue 5th Most Common Venue        6th Most Common Venue  \
63           Dry Cleaner  Dumpling Restaurant  Eastern European Restaurant
79   Empanada Restaurant           Dry Cleaner          Dumpling Restaurant

          7th Most Common Venue 8th Most Common Venue 9th Most Common Venue  \
63           Egyptian Restaurant     Electronics Store           Event Space
79  Eastern European Restaurant   Egyptian Restaurant     Electronics Store

   10th Most Common Venue
63             Donut Shop
79            Event Space
```

Cluster 3

[43]:
```
Queens_merged.loc[Queens_merged['Cluster Labels'] == 2, Queens_merged.
 ↪columns[[0] + list(range(1, Queens_merged.shape[1]))]]
```

[43]:
```
   Neighborhood   Latitude  Longitude    Boro  Cluster Labels  \
50     Neponsit  40.572037 -73.857547  Queens               2

   1st Most Common Venue 2nd Most Common Venue 3rd Most Common Venue  \
50                 Beach         Women's Store           Event Space

   4th Most Common Venue        5th Most Common Venue 6th Most Common Venue  \
50   Dumpling Restaurant  Eastern European Restaurant   Egyptian Restaurant

   7th Most Common Venue 8th Most Common Venue 9th Most Common Venue  \
50     Electronics Store   Empanada Restaurant    Falafel Restaurant

   10th Most Common Venue
50            Flower Shop
```

Cluster 4 Cluster 4 has over 40 results so we wont show it on this report. I have included the commands needed to get the results.

[ ]:
```
Queens_merged.loc[Queens_merged['Cluster Labels'] == 3, Queens_merged.
 ↪columns[[0] + list(range(1, Queens_merged.shape[1]))]]
```

Cluster 5

[45]:
```
Queens_merged.loc[Queens_merged['Cluster Labels'] == 4, Queens_merged.
 ↪columns[[0] + list(range(1, Queens_merged.shape[1]))]]
```

[45]:
```
    Neighborhood   Latitude  Longitude    Boro  Cluster Labels  \
64   Brookville  40.660003 -73.751753  Queens               4

   1st Most Common Venue 2nd Most Common Venue 3rd Most Common Venue  \
64         Deli / Bodega         Women's Store    Falafel Restaurant

   4th Most Common Venue        5th Most Common Venue 6th Most Common Venue  \
64   Dumpling Restaurant  Eastern European Restaurant   Egyptian Restaurant

   7th Most Common Venue 8th Most Common Venue 9th Most Common Venue  \
64     Electronics Store   Empanada Restaurant           Event Space

   10th Most Common Venue
64                   Farm
```

1.7 5. Discussion, Conclusion, & Disclaimer

:- discuss any observations noted and any recommendations based on the results.

This report we observed the crime data of the 5 boroughs, we saw that Brooklyn, Manhattan, & The Bronx had high crime rates with Brooklyn as the highest. This information made it easier to choose which borough we'd put our focus into which will be Queens, Queens is an optimal choose mainly due to the fact that Staten Island had such low crime rate that real estate prices would be astronomical and/or rent prices so Queens was the best choice for financial reasons.

After segmenting & clustering Queens neighborhood venue data from the FourSquare API, we can see that the neighborhoods venues give use an understanding of the demographics of the neighborhoods or at the least what the people in that neighborhoods prefer. By looking at the top 10 most common venues we can get a sense of the neighborhood, by doing this we can pick a preferred neighborhood and/or neighborhoods for the potential investor. Prime example of this is Jackson Heights, Jackson Heights has 5 of its top 10 most common venues as spanish restaurants, this tells use that its mostly a spanish community. Another thing we will notice is that Queens has a lot of Dei/Bodega it's the number 1 most common venue in a number of neighborhoods. Now for us to pick which neighborhoods is adequate for our investor we will look for neighborhoods with venues that require high foot traffic such as metro stations, gyms, parks, & malls. One neighborhood that meets these criteria is Beechhurst, Beechhurst top 3 most common venues are Chinese Restaurant, Yoga Studio, Shopping Mall which are not considered competitor to the market the potential investor wants to get into, not to mention 2 of its top 10 most common venues are Gym, & Gym/Fitness Center. With all these high foot traffic venues opening a Bakery-Coffee shop would be optimal for the potential investor.

In conclusion, we see that that there are 3 potential neighborhoods that we can open a Bakery-Coffee shop in, these 3 neighborhoods with high foot traffic are: - Beechhurst: Chinese Restaurant, Yoga Studio, Shopping Mall, Donut Shop, Supermarket, Gym, Gym/Fitness Center, Dessert Shop, Italian Restaurant, Deli/Bodega - Forest Hills: Gym/Fitness Center, Gym, Yoga Studio, Pharmacy, Pizza Place, Park, Thai Restaurant, Convenience Store, Farmers Market, Food Truck - Hollis: Park, Shopping Mall, Sandwich Place, Fried Chicken Joint, Baseball Field, Discount Store, Asian Restaurant, Bakery, Electronics Store, Grocery from what wee can see these 3 neighborhoods have one thing in common and that is the communities are highly active whether it's gym's or baseball fields to shopping malls, this indicates that these neighborhoods are high foot traffic areas. High foot traffic helps our potential investors investment with exposure to the market without having to put out adds or chase potential customers.

In comparison here are a couple of neighborhoods with coffee shops or bakeries as one of their top 5 most common venues: - Arverne: Surf Spot, Metro Station, Sandwich Place, Coffee Shop, Pizza Place, Board Shop, Bus Stop, Bed & Breakfast, Beach, Donut Shop - Lefrak City: Cosmetics Shop, Department Store, Bakery, Pharmacy, Supplement Shop, Restaurant, Mexican Restaurant, Dry Cleaner, Furniture/Home Store, Fruit & Vegetable Store

We see that Arverne second most common venue is a Metro Stations, this will cause that neighborhood to have high foot traffic due to commuters so its optimal location for coffee shops. Lefrak City shopping venues such as cosmetics and department stores, this will also have modest foot traffic so having a bakery is understandable (Note this will depend on their menus options).

From what we've gather Beechhurst is the optimal neighborhood, mainly due to the fact that Forest Hills tenth most common venue are Food Truck and that might take potential customers away. Hollis has Bakeries as the eighth most common venue this is in direct competition with the investor business.