CAPSTONE PROJECT- BATTLE OF THE NEIGHBORHOODS

City comparison and Restaurant Venue Recommendation

Purpose:

This document provides the details of my final peer reviewed assignment for the IBM Data Science Professional Certificate program.

INTRODUCTION:

New York City is the biggest city in United States and Toronto is the capital city of the province of Ontario in Canada. Both cities have some similarities and also differences. Both cities are quite safe. Both are the economic, cultural, and fashion capitals of their countries.
New York City's architecture is far more impressive. Toronto's a lovely city, but its streetscapes are made beautiful by the people who occupy it--the cafes and such. The architecture itself leaves much to be desired. New York is brilliantly planned. If anyone in New York for a three-day weekend, get a hotel in Midtown Manhattan and nearly everything able to see and visit within walking distance. Toronto is much more sprawled out and need to use the subway lots. (3 of TO's biggest attractions: the CN Tower, the Royal Ontario Museum, the Ontario Science Centre and Casa Loma, are nowhere near each other).

New York has AMAZING pizza and the best Italian restaurants in North America; Toronto has quite possibly the worst pizza on the planet. With its diverse culture, comes diverse food items. There are many restaurants in New York City, each belonging to different categories like Italian, Chinese, Indian, French etc.

So, as part of this project, I will list and visualize all recommended place of New York City where someone can open restaurant and perfect location to setup the office.

The sample recommender in this notebook will provide the following use case scenario:

- Populated area like shopping complex, school, college, office is placed, there will restaurant suitable.
- Area wise target customer.
- Customer wise food variant.
- Based on peoples earning define food cost.

DATA ACQUISITION:

New York data set will retrieve from this link https://cocl.us/new_york_dataset . Then will explore Neighborhoods in New York City and analyze each neighborhood.

I will be using the FourSquare API (FourSquare website: www.foursquare.com) to explore neighborhoods in selected area. The Foursquare explore function will be used to get the most common venue categories in each neighborhood, and then use this feature to group the neighborhoods into clusters. The following information are retrieved on the first query:

- Venue ID
- Venue Name
- Coordinates: Latitude and Longitude

Another venue query will be performed to retrieve venue ratings (based on selected parameters) for each location.

I will use GeoSpace data from https://data.cityofnewyork.us/City-Government/Borough-Boundaries/tqmj-j8zm to get New York Borough boundaries that will help us visualize choropleth map.

Approach:

- Collect the New York City data from https://cocl.us/new_york_dataset
- Using FourSquare API, we will find all venues for each neighborhood.
- Find out all venues for restaurants.
- Find rating, tips and like count and preferable place for restaurants using FourSquare API.
- Using rating for each restaurant, we will sort that data.
- Visualize the Ranking of neighborhoods using folium library(python)

Questions based on datasets

- What is best location in New York City for restaurant business?
- Which areas have potential Restaurant market?
- Which all areas lack of restaurants?
- Which is the best place to setup the office?

METHODOLOGY

Restaurant Venue Recommendation:

The source data contains recommended neighborhood in New York City. I will retrieve the most recommended neighborhood from New York City dataset with some parameter like populated area. For this demonstration, I will simplify the analysis by using conditional function.

Data download and convert:

Data collect from https://geo.nyu.edu/catalog/nyu_2451_34572. I downloaded the files and open as jason format. The retrieved dataset contains jason data. I will convert it in pandas dataframe.
By the following code I just explore the dataset. Then Define the data with the 'features'.

```
In [2]: with open('newyork_data.json') as json_data:
            newyork_data = json.load(json_data)
```

```
In [3]: newyork_data
```

```
Out[3]: {'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',
```

```
In [4]: neighborhoods_data = newyork_data['features']
        #Let's take a look at the first item in this list.
        neighborhoods_data[0]
```

```
Out[4]: {'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]}}
```

Convert data in 'Pandas Dataframe':

The next task is essentially transforming this data of nested Python dictionaries into a *pandas* dataframe. So, let's start by creating an empty dataframe. At first define columns then transform data into dataframe.

In [5]:
```
# define the dataframe columns
column_names = ['Borough', 'Neighborhood', 'Latitude', 'Longitude']

# instantiate the dataframe
neighborhoods = pd.DataFrame(columns=column_names)
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({'Borough': borough,
                                          'Neighborhood': neighborhood_name,
                                          'Latitude': neighborhood_lat,
                                          'Longitude': neighborhood_lon}, ignore_index=True)
```

The resultant dataframe is,

In [6]:
```
neighborhoods
```

Out[6]:

	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

Use geopy library to get the latitude and longitude values of New York City:

In order to define an instance of the geocoder, have to define user_agent. I will name this agent *ny_explorer*, as shown below.

In [8]:
```
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.
```

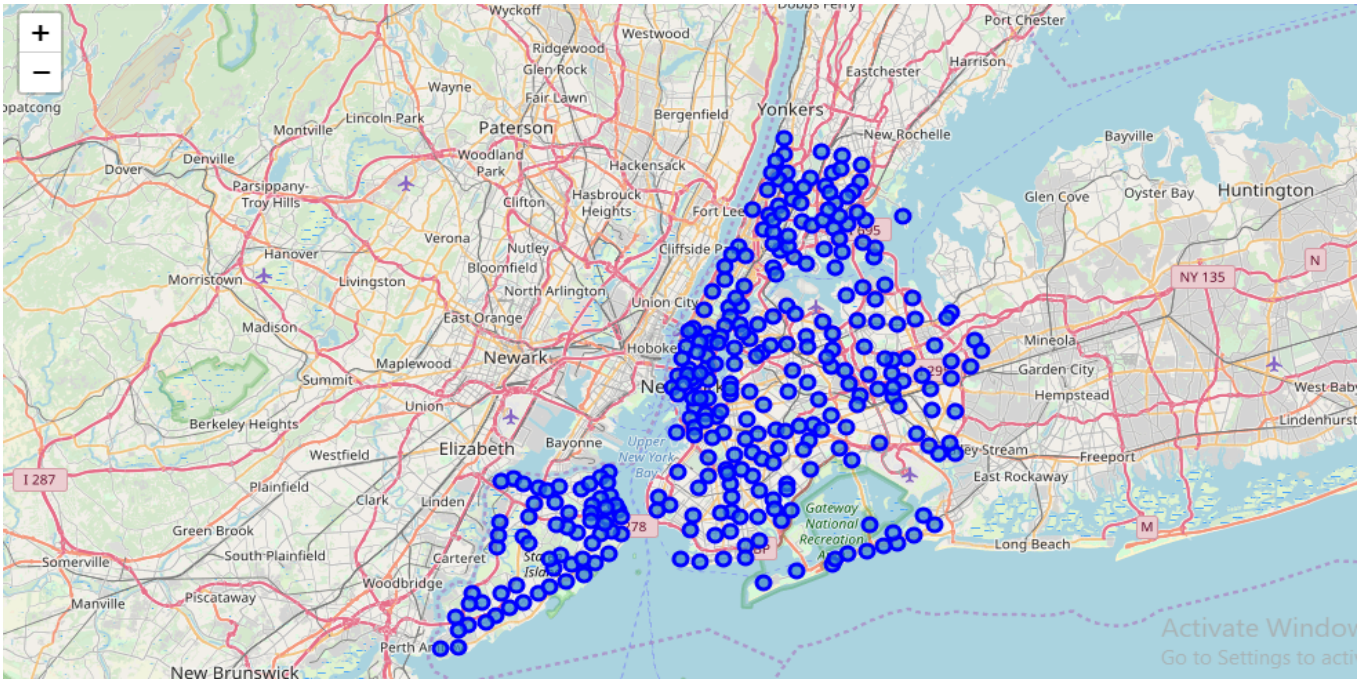
Create a map of New York with neighborhoods superimposed on top.

```
In [9]: # 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(neighborhoods['Latitude'], neighborhoods['Longitude'], neighborhoods['Borough'], neigl
    label = '{}, {}'.format(neighborhood, borough)
    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_newyork)

display(map_newyork)
```

The output of the code



Retrieving FourSquare Places of interest:

Using the Foursquare API, the **explore** API function was be used to get the most common venue categories in each neighborhood, and then used this feature to group the neighborhoods into clusters. Here by this API I will get the information about New York neighborhoods then get preferable places for restaurant business. To get data from Foursquare process will describe below.

Define Foursquare Credentials and Version:

```
In [10]: CLIENT_ID = '1QEORTYPVXPDCE2DFVI0ZOYGIZRLOVBILKIWE1QSOG3TVQI1' # my Foursquare ID
         CLIENT_SECRET = 'LCVRO2WKPX0ASZK4VJKEXOE201IMJSV3I01EJLLS4D3Y5SWZ' # my Foursquare Secret
         VERSION = '20180605' # Foursquare API version

         print('Your credentails:')
         print('CLIENT_ID: ' + CLIENT_ID)
         print('CLIENT_SECRET:' + CLIENT_SECRET)

         Your credentails:
         CLIENT_ID: 1QEORTYPVXPDCE2DFVI0ZOYGIZRLOVBILKIWE1QSOG3TVQI1
         CLIENT_SECRET:LCVRO2WKPX0ASZK4VJKEXOE201IMJSV3I01EJLLS4D3Y5SWZ
```

Let's explore top 100 venue within a radius of 500 meters:

```
In [11]: LIMIT = 100
         radius = 500
         url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={},{}&radius={}&limit={}'.format(
             CLIENT_ID,
             CLIENT_SECRET,
             VERSION,
             latitude,
             longitude,
             radius,
             LIMIT)
         url

Out[11]: 'https://api.foursquare.com/v2/venues/explore?&client_id=1QEORTYPVXPDCE2DFVI0ZOYGIZRLOVBILKIWE1QSOG3TVQI1&client_secret=LCVRO2W
         KPX0ASZK4VJKEXOE201IMJSV3I01EJLLS4D3Y5SWZ&v=20180605&ll=40.7127281,-74.0060152&radius=500&limit=100'
```

Send the GET request and examine the results:

```
In [12]: results = requests.get(url).json()
         results

Out[12]: {'meta': {'code': 200, 'requestId': '5dc9610dc58ed7002cfdbbc5'},
          'response': {'suggestedFilters': {'header': 'Tap to show:',
            'filters': [{'name': 'Open now', 'key': 'openNow'},
             {'name': '$-$$$$', 'key': 'price'}]},
           'headerLocation': 'Downtown Manhattan',
           'headerFullLocation': 'Downtown Manhattan, New York',
           'headerLocationGranularity': 'neighborhood',
           'totalResults': 161,
           'suggestedBounds': {'ne': {'lat': 40.7172281045, 'lng': -74.00008952063419},
            'sw': {'lat': 40.7082280955, 'lng': -74.0119408793658}},
           'groups': [{'type': 'Recommended Places',
             'name': 'recommended',
             'items': [{'reasons': {'count': 0,
                'items': [{'summary': 'This spot is popular',
                  'type': 'general',
                  'reasonName': 'globalInteractionReason'}]},
               'venue': {'id': '57f0689d498e7d49d9189369',
                'name': 'The Bar Room at Temple Court',
                'location': {'address': '123 Nassau St',
```

Here is main work. I will define venue of crowded area without restaurant and create dataframe. By this I will get listed venue where suitable for restaurant.

In [17]:
```
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={}'.for
            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 if 'Restaurant' not in v['venue']['categories'][0]['name']])

    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)
nyc_venues = getNearbyVenues(names=neighborhoods['Neighborhood'],
                             latitudes=neighborhoods['Latitude'],
                             longitudes=neighborhoods['Longitude']
                            )
```

Sample of result:

```
Wakefield
Co-op City
Eastchester
Fieldston
Riverdale
Kingsbridge
Marble Hill
Woodlawn
Norwood
Williamsbridge
Baychester
Pelham Parkway
City Island
Bedford Park
University Heights
Morris Heights
Fordham
East Tremont
West Farms
```

Here showing resulted neighborhood where restaurant preferable. I select preferable place which are most surrounding with venue and define this whose listed value bigger then 75.

```
In [26]: res_venues = nyc_venues.groupby(['Neighborhood','Neighborhood Latitude','Neighborhood Longitude'])
                      .count().sort_values('Venue Category', ascending=False).reset_index()
         res_venues = res_venues[res_venues['Venue Category']>=75]
         res_venues[['Neighborhood','Venue Category']]
```

Out[26]:

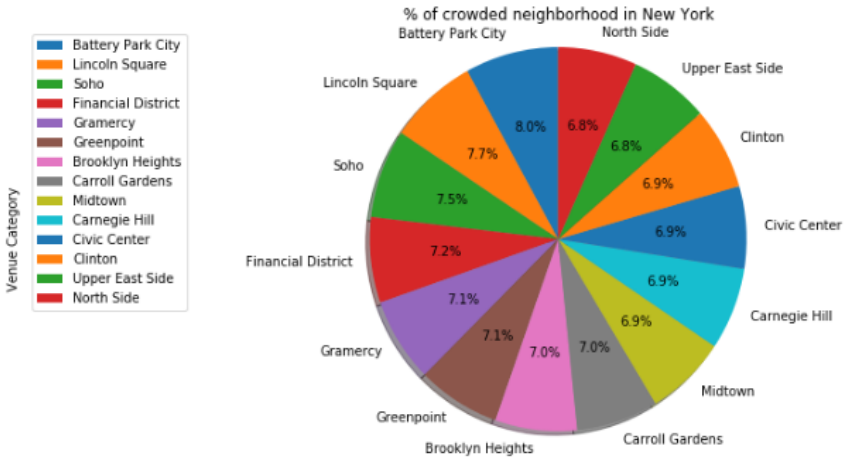
	Neighborhood	Venue Category
0	Battery Park City	90
1	Lincoln Square	86
2	Soho	82
3	Civic Center	79
4	Financial District	79
5	Gramercy	79
6	Greenpoint	79
7	Clinton	78
8	Brooklyn Heights	78
9	Midtown	77
10	Carroll Gardens	77
11	Upper East Side	76
12	Carnegie Hill	76
13	North Side	75
14	Cobble Hill	75
15	Chelsea	75

Let's show the result with the visualization:

Here I am showing 2 type graphical view for the result. One is pie and another one is bar chart. Its help us to get decision that which places is suitable for the restaurant business.

Lets visualize the result

In [80]:
```
df_res_venues = res_venues.set_index('Neighborhood')
df_res_venues[['Venue Category']].plot(kind='pie',
                                       figsize=(15, 6),
                                       autopct='%1.1f%%',
                                       startangle=90,
                                       shadow=True,
                                       subplots=True,
                                       )
plt.title('% of crowded neighborhood in New York')
plt.axis('equal')
plt.legend(labels=df_res_venues.index, loc='upper left')
plt.show()
```

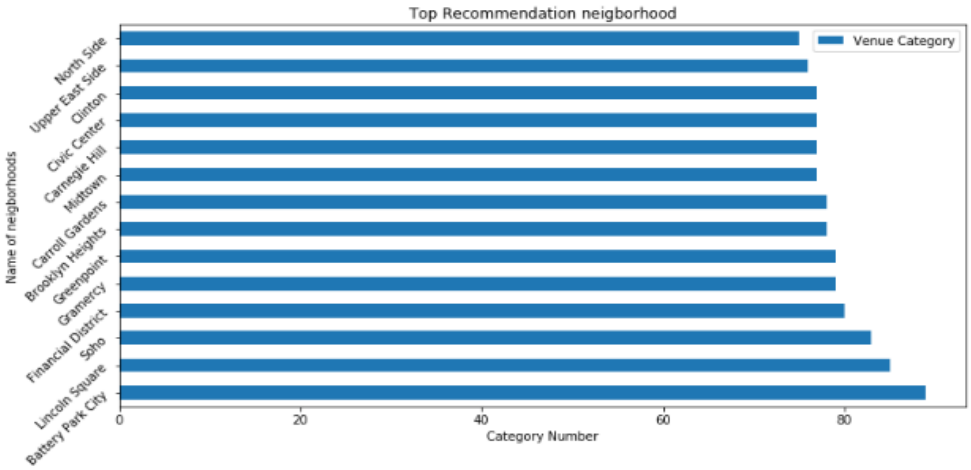


Activate Wi
Go to Settings

Lets check in another graph

In [99]:
```
df_res_venues[['Venue Category']].plot(kind='barh', figsize=(12, 6), rot=90) # rotate the bars by 90 degrees

plt.xlabel('Category Number')
plt.ylabel('Name of neigborhoods')
plt.title('Top Recommendation neigborhood')
plt.yticks(
    rotation=45,
    horizontalalignment='right',
    fontweight='light',
    fontsize='medium',
)
plt.show()
```



Activ
Go to

Discussion and Conclusion

On this notebook, Analysis of best town venue recommendations based on restaurant venue category has been presented. Recommendations based on people go there mostly. As New York city is a big city with a whole host of interesting venues scattered around the city, the information extracted in this notebook present on the city areas, will be a good supplement to web based recommendations for visitors to find out nearby venues of interest and be a useful aid in deciding a place to stay or where to go during their visits. Using Foursquare API, we have collected a good amount of venue recommendations in New York City. Sourcing from the venue recommendations from FourSquare has its limitation; The list of venues is not exhaustive list of all the available venues is the area. Furthermore, not all the venues found in the area has crowded. For this reason, the number of analyzed venues has bigger than 75 number categories are available venues initially collected. The results therefore may significantly change, when more information is collected on those with missing data. The generated neighborhood from our results very good and interesting places located in areas. This kind of results may be very interesting restaurant business. Our results also yielded some interesting findings. For instance, the initial assumption among websites providing recommendations is that the Central Area. I will be providing another supplementary Inferential Statics in the future about on these data collected and also update in a new notebook using other categories. For now, this completes the requirements for this task. Thank you.

Shah Alam Sumon
email: sacsesumon@gmail.com
Created For: COURSERA **IBM Applied Data Science Capstone Project**