Predicting the most popular area to open restaurants in NY City

Manasa S K

September 21, 2019

1. Introduction

1.1 Background

New York City comprises 5 boroughs sitting where the Hudson River meets the Atlantic Ocean. At its core is Manhattan, a densely populated borough that's among the world's major commercial, financial and cultural centers. Its iconic sites include skyscrapers such as the Empire State Building and sprawling Central Park. Broadway Theater is staged in neon-lit Times Square. Since New York has more visitors from different part of world, they prefer different kind of foods because it is popular city, hence it need more restaurants intern produces more profit in opening a restaurants.

1.2 Problem

Company ABC decided to open specific kind of restaurants in New York, so the company called a data scientist team to find the most popular area in New York City, the data scientist as to find the major commercial, financial and cultural center so the company can open restaurants in nearest popular areas and gains more profit.

1.3 Interest

Obviously, The Company would call the data scientists to predict most accurate popular places of New York .This is also used to predict popular places of New York for tourism.

2. Data acquisition and cleaning

2.1 Data sources

The dataset newyork_data is already given by course era .Dataset is in json format In the dataset we considered many towns of NY city , each with its features is provided in the dataset ,Each features have 13 different values .From 13 values 4 are considered they are borough , neighborhood ,latitude ,longitude

2.2 Data cleaning

Data downloaded or scraped from multiple sources were combined into one table. There are lot of values in the data, among that values which are under interest is choosen.The following snapshot shows how the data is cleaned.

```
In [3]:  with open('Newyork_data.json') as json_data:
             newyork_data = json.load(json_data)
```

```
In [4]:  newyork_data
```


```
Out[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',
```


Pre-processing the data

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

Transform the dataset into dataframes

```
column_names = ['Borough', 'Neighborhood', 'Latitude', 'Longitude'] #selecting only 4 columns

# 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)
```

```
In [7]:  neighborhoods#data
```

Out[7]:

	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
5	Bronx	Kingsbridge	40.881687	-73.902818
6	Manhattan	Marble Hill	40.876551	-73.910660
7	Bronx	Woodlawn	40.898273	-73.867315
8	Bronx	Norwood	40.877224	-73.879391
9	Bronx	Williamsbridge	40.881039	-73.857446
10	Bronx	Baychester	40.866858	-73.835798

3. Exploratory Data Analysis

3.1 Creating New York map

To create a map of New York City for analysis we use geopy library to find the latitude and longitude of the city, then we get the visualization of New York City .This is shown in below code.

Using geopy library to find the latitude and longitude of Newyork city

```
In [9]:  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))
```

Create the map of newyork city

```
In [10]: # 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'], r
             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)

         map_newyork
```

3.2 Retrieving top 500 venues in Manhattan

We consider a city Manhattan then find the top 500 venues in that. Similarly we can also consider other cities to find the venues. This is shown below.

Getting the top venues within 500 meters radius in Manhattan

```
In [11]: manh_data = neighborhoods[neighborhoods['Borough'] == 'Manhattan'].reset_index(drop=True)
         manh_data.loc[0, 'Neighborhood']
         neighborhood_latitude = manh_data.loc[0, 'Latitude'] # neighborhood latitude value
         neighborhood_longitude = manh_data.loc[0, 'Longitude'] # neighborhood longitude value

         neighborhood_name = manh_data.loc[0, 'Neighborhood'] # neighborhood name


         # Now, let's get the top 100 venues that are in Marble Hill within a radius of 500 meters.

         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,
             neighborhood_latitude,
             neighborhood_longitude,
             radius,
             LIMIT)

         results = requests.get(url).json()
```

```
In [12]:  # function that extracts the category of the venue
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']
```

```
In [13]:  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()
```

3.3 Retrieve nearby venues

Among top 500 venues nearby venues are retrieved through a function. This is shown as below.

This below function is used to get the near by venues

```
In [14]:  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']['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)
```

```
In [15]:  manh_venues = getNearbyVenues(names=manh_data['Neighborhood'],
                                   latitudes=manh_data['Latitude'],
                                   longitudes=manh_data['Longitude']
                                  )
```

In [16]:
```
print(manh_venues.shape)
manh_venues.head()
```

(3331, 7)

Out[16]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Marble Hill	40.876551	-73.91066	Arturo's	40.874412	-73.910271	Pizza Place
1	Marble Hill	40.876551	-73.91066	Bikram Yoga	40.876844	-73.906204	Yoga Studio
2	Marble Hill	40.876551	-73.91066	Tibbett Diner	40.880404	-73.908937	Diner
3	Marble Hill	40.876551	-73.91066	Starbucks	40.877531	-73.905582	Coffee Shop
4	Marble Hill	40.876551	-73.91066	Dunkin'	40.877136	-73.906666	Donut Shop

3.4 Hot encoding the data

Since the data values has to be turned into zeroes and once we use hot encoding it is shown has below.

hot encoding the data

In [17]:
```
# one hot encoding
manh_onehot = pd.get_dummies(manh_venues[['Venue Category']], prefix="", prefix_sep="")

# add neighborhood column back to dataframe
manh_onehot['Neighborhood'] = manh_venues['Neighborhood']

# move neighborhood column to the first column
fixed_columns = [manh_onehot.columns[-1]] + list(manh_onehot.columns[:-1])
manh_onehot = manh_onehot[fixed_columns]

manh_onehot.head()
```

3.4 Finding top 5 venues

Among several venues top 5 venues are retrieved and there frequency is counted as below.

Finding the top 5 venues in Manhattan

In [19]:
```
num_top_venues = 5

for hood in manh_grouped['Neighborhood']:
    print("hood")
    temp = manh_grouped[manh_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')
```

This gives the result as:

```
hood
            venue  freq
0     Coffee Shop  0.07
1            Park  0.07
2           Hotel  0.05
3  Memorial Site  0.04
4             Gym  0.04


hood
          venue  freq
0   Pizza Place  0.06
1   Coffee Shop  0.06
2          Café  0.05
3           Bar  0.04
4   Yoga Studio  0.03


hood
```

Among them top 10 restaurant are selected, this identifies the restaurant kind that are present in that area.

To get top 10 venue numbers

In [21]:
```
num_of_top_venues = 10

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

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

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

for ind in np.arange(manh_grouped.shape[0]):
    neighborhoods_venues_sorted.iloc[i, 1:] = return_most_common_venues(manh_grouped.iloc[i, :], num_of_top_venues)

neighborhoods_venues_sorted.head()
```

3.5 Clustering of restaurants

Once the top 10 are retrieved then they are subjected to clustering, I considered 8 clusters this is shown below.

clustering

In [22]:
```
kclusters = 8 #considering 8 clusters

manh_grouped_clustering = manh_grouped.drop('Neighborhood', 1)

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

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

Out[22]:
```
array([7, 0, 0, 7, 0, 7, 7, 3, 0, 7])
```

In [23]:
```
neighborhoods_venues_sorted.insert(0, 'Cluster Labels', kmeans.labels_)

manh_merged = manh_data

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

manh_merged.head() # check the last column
```

The below cell is for cluster 0 likewise 8 clusters are made

In [25]:
```
manh_merged.loc[manh_merged['Cluster Labels'] == 0, manh_merged.columns[[1] + list(range(5, manh_merged.shape[1]))]]
```

4. Conclusion

It will be a good idea to setup Chinese, African, Canadian restaurants here because these neighborhoods are known for restaurants and have no above mentioned Restaurants that could stand as competitors. However, it could be helpful to check the demographics of people in these