This notebook will be used for the Assignment of week 5 of capstone course for Applied Data Science on Coursera

Author: Rajiv Ranjan Singh

1. Introduction

In this project for the capstone course for Applied Data Science, we shall cluster and compare neighbourhoods in Toronto and New York City on the basis of popular venues, major crime indicators and area of the neighbourhood. At the end of the exercise, a desirable output shall be a table of similar neighbourhoods across New York and Toronto. This information shall be useful for anyone who is doing business in any of these cities and wants to expand to the other city. It shall also be useful for professionals who are looking to change jobs within New York or Toronto or from one city to another.

1.1 Background

The City of New York, usually called either New York City (NYC) or simply New York (NY), is the most populous city in the United States. With an estimated 2018 population of 8,398,748 distributed over a land area of about 302.6 square miles (784 km2), New York is also the most densely populated major city in the United States.Located at the southern tip of the state of New York, the city is the center of the New York metropolitan area, the largest metropolitan area in the world by urban landmass and one of the world's most populous megacities. A global power city, New York City has been described as the cultural,financial and media capital of the world, and exerts a significant impact upon commerce, entertainment, research, technology, education, politics, tourism, art, fashion, and sports.

Similarly, Toronto is the provincial capital of Ontario and the most populous city in Canada, with a population of 2,731,571 in 2016. The diverse population of Toronto reflects its current and historical role as an important destination for immigrants to Canada. Its varied cultural institutions, which include numerous museums and galleries, festivals and public events, entertainment districts, national historic sites, and sports activities,attract over 43 million tourists each year. Toronto is an international centre of business, finance, arts, and culture, and is recognized as one of the most multicultural and cosmopolitan cities in the world.

1.2 Problem Description

Both cities have a large and diverse population of both Toronto and New York, including . Every year hundreds of thousands of immigrants, businessmen and professionals visit, migrate to or settle in these cities for work, livelihood and tourism. Due to the large area, several neighbourhoods, income differences, and variations in quality of life from one neighbourhood to another, it is often a tedious task to find neighbouhoods suitable to one's preferences. Businessmen often need information on neighbourhood to reloacte or to open new businesses. Further, anyone moving from New York to Toronto or vie versa would want to move to more or less similar neighbourhood. Also many people prefer to avoid crime-prone areas of a city whether for residence or business. Therefore the crime data for each neighbourhood is also relevant to categorization of similar neighbourhoods.

Therefore the problem is to group neighbourhoods within and across New York and Toronto and categorize them based on popular venues, businesses, area and crime rate.

1.3 Target Audience

1. Businesses looking for expansion in New York and Toronto
2. Professional looking for relocation
3. Students looking for relocation
4. House buyers

1.4 Success criteria

A good categorization of neighbourhoods between Toronto and New York and their prominent features.

2. Data

For our project we will need the following data for both Toronto and New York:

1. Major Crime Indicators for each neighbourhood/precinct
2. Area, Latitude and Longitutde for each neighbourhood/precinct
3. List of popular venues for each neighbourhood/precinct

2.1 Data for Toronto

1. For crime statistics of Toronto we shall use the data provided [here. (https://services.arcgis.com/S9th0jAJ7bqgIRjw/arcgis/rest/services/Neighbourhood_MCI/FeatureServer/0/query?where=1%3D1&outFields=*&returnGeometry=false&outSR=4326&f=json)](https://services.arcgis.com/S9th0jAJ7bqgIRjw/arcgis/rest/services/Neighbourhood_MCI/FeatureServer/0/query?where=1%3D1&outFields=*&returnGeometry=false&outSR=4326&f=json) This dataset provides a number of features including, neighbourhood name, neighbouhood id, major crime indicators from 2014-2018, average of major crimes for last 5 years, area and length of boundaries of each neighbourhood and population. For our analysis, we shall retain the data on

    - Neighbourhood
    - Assault_AVG
    - AutoTheft_AVG
    - BreakandEnter_AVG
    - Robbery_AVG
    - TheftOver_AVG
    - Homicide_AVG
    - Shape__Area

2. For latitude and longitude of each neighbourhood, we shall fetch the data for each neighbourhood using **'Here'** geocoder from geopy library in Python.

3. The data of venues and venue categories for each neighbourhood will be fetched using **Foursquare API**.

2.2 Data for New York

1. For crime statistics of New York we shall use the data provided [here. (https://www1.nyc.gov/assets/nypd/downloads/excel/analysis_and_planning/historical-crime-data/seven-major-felony-offenses-by-precinct-2000-2018.xls)](https://www1.nyc.gov/assets/nypd/downloads/excel/analysis_and_planning/historical-crime-data/seven-major-felony-offenses-by-precinct-2000-2018.xls) This dataset provides a number of features including precinct id, major crimes from 2001-2018. For our analysis, we shall retain the data on

    - Precinct id
    - Average of Crime figures from 2014 to 2018

Using **dataset cleaning and feature engineering**, we shall compute the average of major crimes for each precinct and rename the columns to align the dataset with toronto dataset. For further analysis, each precinct shall be treated as a neighbourhood.

2. For location data of New York Precincts, open data available at [this link (https://data.cityofnewyork.us/api/views/kmub-vria/rows.csv?accessType=DOWNLOAD)](https://data.cityofnewyork.us/api/views/kmub-vria/rows.csv?accessType=DOWNLOAD) will be used. This dataset provides the boundary data and shape_area for each precinct. The centroid (latitude and longitude) of each precinct will be calculated by extracting the boundary data from above dataset and used as neighbourhood latitude and longitude. Shape_area feature will be used as it is.

During coding, the performance of several free geocoders in fetching location data for precincts was observed to be poor as many precincts could not be located by free geocoders available in geopy.

3. The data of venues and venue categories for each neighbourhood will be fetched using **Foursquare API**.

3. Methodology

3.1 Data Retrieval,Cleaning and Feature Engineering

3.1.1 Toronto

- First of all crime data of Toronto was retrieved from [here. (https://services.arcgis.com/S9th0jAJ7bqgIRjw/arcgis/rest/services/Neighbourhood_MCI/FeatureServer/0/query?where=1%3D1&outFields=*&returnGeometry=false&outSR=4326&f=json)](https://services.arcgis.com/S9th0jAJ7bqgIRjw/arcgis/rest/services/Neighbourhood_MCI/FeatureServer/0/query?where=1%3D1&outFields=*&returnGeometry=false&outSR=4326&f=json) in json format. First, json dataset was flattened and saved as pandas Dataframe.

Create URL for fetching crime records from Toronto police data portal

In [2]:
```
url_crime = "https://services.arcgis.com/S9th0jAJ7bqgIRjw/arcgis/rest/services/Neighbourhood_MCI/FeatureServer/0/query?whe
crime_toronto_json = requests.get(url_crime).json()
```

convert to pandas Dataframe and flatten json

In [3]:
```
crime_toronto_df = json_normalize(crime_toronto_json["features"])
crime_toronto_df.head()
```

Out[3]:

	attributes.OBJECTID	attributes.Neighbourhood_Crime_Rates_Neigh	attributes.Neighbourhood_Crime_Rates_Hood_	attributes.Hood_ID	attributes.Neighbourho
0	1	Yonge-St.Clair	097	97	Yonge-St.Cl
1	2	York University Heights	027	27	York University Heigh
2	3	Lansing-Westgate	038	38	Lansing-Westga

- Then column names were cleaned and on following 8 columns were retained for further analysis.
  - Neighbourhood
  - Assault_AVG
  - AutoTheft_AVG
  - BreakandEnter_AVG
  - Robbery_AVG
  - TheftOver_AVG
  - Homicide_AVG
  - Shape__Area

In [4]:
```
# clean columns names
crime_toronto_df.columns = [col.split(".")[-1] for col in crime_toronto_df.columns]
```

In [5]:
```
crime_toronto_df.head()
```

Out[5]:

	OBJECTID	Neighbourhood_Crime_Rates_Neigh	Neighbourhood_Crime_Rates_Hood_	Hood_ID	Neighbourhood	Assault_2014	Assault_2015	Assault_20
0	1	Yonge-St.Clair	097	97	Yonge-St.Clair	58	38	
1	2	York University Heights	027	27	York University Heights	78	101	1
2	3	Lansing-Westgate	038	38	Lansing-Westgate	216	203	2
3	4	Yorkdale-Glen Park	031	31	Yorkdale-Glen Park	121	141	1
4	5	Stonegate-Queensway	016	16	Stonegate-Queensway	109	140	1

In [6]:
```
# filter columns
filtered_columns_crime_toronto = ['Neighbourhood', 'Assault_AVG', 'AutoTheft_AVG','BreakandEnter_AVG','Robber
                                  'TheftOver_AVG','Homicide_AVG','Shape__Area']
crime_toronto_df = crime_toronto_df.loc[:, filtered_columns_crime_toronto]

print(crime_toronto_df.head())
print(crime_toronto_df.shape)
```

- It was observed that Homicide averages were 'NA' for some neighbourhoods where 0 homocides were recorded. hence 35 such values were accordingly replaced with 0 (int).

Homicide AVG as N/A can be replaced with 0 based on analysis.

```
In [7]: crime_toronto_df[crime_toronto_df == 'N/A'].count()

/home/sp-pasighat/.local/lib/python3.6/site-packages/pandas/core/ops/__init__.py:1115: FutureWarning: elemen
twise comparison failed; returning scalar instead, but in the future will perform elementwise comparison
  result = method(y)

Out[7]: Neighbourhood        0
        Assault_AVG          0
        AutoTheft_AVG        0
        BreakandEnter_AVG    0
        Robbery_AVG          0
        TheftOver_AVG        0
        Homicide_AVG        35
        Shape__Area          0
        dtype: int64

In [8]: crime_toronto_df.replace(value=0,to_replace={'Homicide_AVG':'N/A'},inplace=True)
```

The cleaned dataframe had 140 rows and 8 columns.

- Then latitudes and longitudes of each neighbourhood was fetched using **Geocoder** from **Geopy** package. It was observed that Nominatim geocoder was not able to give location data for all the neighbourhoods in tha dataset. After trials , it was observed that **Here** geocoder had best results among all the free geocoders enabled in Geopy. Further, in order to prevent blocking by "Here" api due to repeated calls **RateLimiter** library from Geopy was used. Using **'tqdm'** package and progress_apply function, a progress bar was used to monitor the geocoding of each neighbourhood.

Use the Geocoder package to create the dataframe with lat and long of each neighbourhood:

```
#from geopy.geocoders import Nominatim
#set timeout to 5 sec for slow interent connection and restrict search to country Canada ("CA")
#geolocator = Nominatim(user_agent="toronto_crime_neighbourhood",timeout = 5,country_bias='CA')

from geopy.geocoders import Here
#set timeout to 5 sec for slow interent connection and restrict search to country Canada ("CA")
geolocator = Here(user_agent="toronto_crime_neighbourhood",
                  timeout = 5,
                  #country_codes='CA',
                  app_id='dGiVRxoqi7Hhs95261Vk',
                  app_code='JOYxudJfw6OPtI7DmPLGLQ')

# using tqdm library for progress bar
from tqdm import tqdm
tqdm.pandas()

#use Rate limiter for delayed fetching of geocodes to avoid errors
from geopy.extra.rate_limiter import RateLimiter
geocode = RateLimiter(geolocator.geocode, min_delay_seconds=0.5,max_retries=5)
crime_toronto_df['location'] = crime_toronto_df['Neighbourhood'].progress_apply(geocode)


#extract point information from location
crime_toronto_df['point'] = crime_toronto_df['location'].progress_apply(lambda loc: tuple(loc.point) if loc e

100%|██████████| 140/140 [02:37<00:00,  1.12s/it]
100%|██████████| 140/140 [00:00<00:00, 41155.21it/s]
```

- Finally, rows with any possible non retrived locations were dropped and cleaned dtaframe had shape of 140 X 10.

3.1.2 New York

- The crime data for New York was much harder to process. We used the data provided [here. (https://www1.nyc.gov/assets/nypd/downloads/excel/analysis_and_planning/historical-crime-data/seven-major-felony-offenses-by-precinct-2000-2018.xls)](https://www1.nyc.gov/assets/nypd/downloads/excel/analysis_and_planning/historical-crime-data/seven-major-felony-offenses-by-precinct-2000-2018.xls) This dataset provides a number of features including precinct id, major crimes from 2001-2018 IN EXCEL FILE FORMAT. As can be seen, columns contain yearwise crime figures ffrom 2000 to 2018 while crime heads are mentioned in successive rows for each precinct. Further, only every 7th row has valid precinct ID data and intermediate cells are 'NaN' due to inability to parse merged cells from xls file.

Part 3: Fetching and cleaning crime data of New York

```
nyc_crime_url = 'https://www1.nyc.gov/assets/nypd/downloads/excel/analysis_and_planning/historical-crime-data
crime_nyc_xls = pd.read_excel(nyc_crime_url,skiprows=2)
crime_nyc_xls.head()
```

	PCT	CRIME	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018
0	1	MURDER & NON NEGL. MANSLAUGHTER ...	3.0	1.0	2.0	2.0	2.0	0.0	1.0	0.0	1.0	2.0	2.0	0.0	1.0	0.0	0.0	1.0	0.0	9.0	1.0
1	NaN	RAPE	12.0	5.0	10.0	11.0	11.0	5.0	4.0	7.0	12.0	4.0	6.0	13.0	10.0	12.0	8.0	7.0	9.0	18.0	23.0
2	NaN	ROBBERY	252.0	188.0	210.0	164.0	154.0	172.0	119.0	128.0	108.0	106.0	96.0	102.0	81.0	68.0	50.0	79.0	60.0	69.0	80.0
3	NaN	FELONY ASSAULT	139.0	164.0	147.0	134.0	129.0	121.0	94.0	90.0	83.0	83.0	68.0	94.0	110.0	87.0	76.0	86.0	78.0	91.0	61.0
4	NaN	BURGLARY	475.0	437.0	297.0	403.0	246.0	293.0	255.0	231.0	189.0	207.0	170.0	155.0	188.0	171.0	145.0	138.0	117.0	65.0	85.0

- **Data Cleaning**: A lot of data cleaning was used to remove garbage data and get AVG of major crime heads for 2014-2018 for each precinct.

```
# drop Total rows for each precinct
crime_nyc_xls.drop(index=list(range(7,616,8)),inplace=True)

# calculate average of each crime head from 2014-2018, same as for Toronto crime data
crime_nyc_xls['AVG'] = crime_nyc_xls.iloc[:,[-5,-4,-3,-2,-1]].apply(np.mean,axis=1)

#dropping all columns except precint id, crime head and AVG
crime_nyc_xls = crime_nyc_xls.iloc[:,[0,1,crime_nyc_xls.shape[1]-1]]

#dropping NA values from AVG column
crime_nyc_xls.dropna(subset = ['AVG'],inplace= True)

#dropping junk rows at the end of file
crime_nyc_xls.drop(index=list(range(616,624,1)),inplace= True)

#only NaN values are in precint id column, using forward fill to repeat precint id for each row.
crime_nyc_xls.fillna(method='ffill',inplace=True)
```

```
crime_nyc_df = crime_nyc_xls.pivot(index='PCT',columns='CRIME',values='AVG')
crime_nyc_df.head()
```

CRIME	BURGLARY	FELONY ASSAULT	GRAND LARCENY	GRAND LARCENY OF MOTOR VEHICLE	MURDER & NON NEGL. MANSLAUGHTER	RAPE	ROBBERY
PCT							
1	110.0	78.4	1048.8	28.4	2.2	13.0	67.6
5	96.8	134.6	546.8	17.8	0.8	7.2	90.4
6	142.2	116.8	1057.0	32.4	0.6	12.6	125.0

- Finally Dataframe.pivot and renaming of columns was used to get the dataset in same order as Toronto crime data.

In [13]:
```
crime_nyc_df = crime_nyc_xls.pivot(index='PCT',columns='CRIME',values='AVG')
crime_nyc_df.head()
```

Out[13]:

CRIME	BURGLARY	FELONY ASSAULT	GRAND LARCENY	GRAND LARCENY OF MOTOR VEHICLE	MURDER & NON NEGL. MANSLAUGHTER	RAPE	ROBBERY
PCT							
1	110.0	78.4	1048.8	28.4	2.2	13.0	67.6
5	96.8	134.6	546.8	17.8	0.8	7.2	90.4
6	142.2	116.8	1057.0	32.4	0.6	12.6	125.0
7	70.8	155.2	440.0	25.6	0.4	11.4	120.6
9	135.4	151.0	878.2	38.2	2.2	21.4	120.2

In [14]:
```
newnames = dict(zip(crime_nyc_df.columns,['BreakandEnter_AVG','Assault_AVG','TheftOver_AVG',
                                          'AutoTheft_AVG','Homicide_AVG','Rape','Robbery_AVG']))
```

In [15]:
```
crime_nyc_df.rename(columns=newnames,inplace=True,errors='raise')

crime_nyc_df.drop(columns='Rape',inplace=True)

crime_nyc_df.reset_index(inplace=True)
```

- Since, neighbourhood wise crime data for new York was not available freely, precincts were treated as neighbourhoods. Retreving the location (lat,long) for each precinct turned out to be difficult. The open dataset available as csv file [here. (https://data.cityofnewyork.us/api/views/kmub-vria/rows.csv?accessType=DOWNLOAD)](https://data.cityofnewyork.us/api/views/kmub-vria/rows.csv?accessType=DOWNLOAD) It featured precinct ID, Shape_area, Shape_length and the all the locations forming the boundary geometry of each precinct. A custom function was defined to extract all the locations from boundary geometry and return the mean of latitudes and longitudes of all boundary points as the central latitude and longitude for precinct.

Fetch geosptial Data for New York Precincts from public data on NYC portal and create final Dataframe with crime and latitude and longtitude

In [26]:
```
nyc_pct_url = https://data.cityofnewyork.us/api/views/kmub-vria/rows.csv?accessType=DOWNLOAD
nyc_pct_df = pd.read_csv(nyc_pct_url)
nyc_pct_df.head()
```

Out[26]:

	Shape_Area	Precinct	Shape_Leng	the_geom
0	4.731472e+07	1	80797.248793	MULTIPOLYGON (((-74.0438776157395 40.690187676...
1	1.808880e+07	5	18676.124259	MULTIPOLYGON (((-73.98863862848766 40.72293372...
2	2.209819e+07	6	26402.900669	MULTIPOLYGON (((-73.99968392160721 40.73855224...
3	4.533179e+07	71	29978.094261	MULTIPOLYGON (((-73.92854313809303 40.66457328...
4	1.046213e+08	72	87968.194520	MULTIPOLYGON (((-73.99840899113158 40.67186872...

In [27]:
```
import re
def extract_lat_long_from_polygon(polygon):
    x = polygon[16:-3]
    series = x.split(', ')
    new = pd.DataFrame(series)
    new = new[0].str.split(" ", n = 1, expand = True)
    new[0] = new[0].apply(lambda loc: re.sub(r'[\(\)\{\}<>]', "", loc))
    new[1] = new[1].apply(lambda loc: re.sub(r'[\(\)\{\}<>]', "", loc))
    new = new.astype(float)

    return new.mean()[1],new.mean()[0]
```

- Finally some columns renaming and reorientation and merging with crime data of New York to get both Toronto and new York Datasets in same feature name and size.

In [29]:
```
nyc_pct_df.drop(columns=['Shape_Leng','the_geom'],inplace = True)
nyc_pct_df.rename(columns={'Precinct':'Neighbourhood','Shape_Area':'Shape__Area'},inplace=True)
nyc_pct_df = nyc_pct_df[['Neighbourhood','Shape__Area','Latitude','Longitude']]
nyc_pct_df.head()
```

Out[29]:

	Neighbourhood	Shape__Area	Latitude	Longitude
0	1	4.731472e+07	40.701709	-74.015542
1	5	1.808880e+07	40.713725	-73.997222
2	6	2.209819e+07	40.735407	-74.007312
3	71	4.533179e+07	40.662261	-73.946734
4	72	1.046213e+08	40.654756	-74.007489

In [30]:
```
crime_nyc_df = crime_nyc_df.merge(nyc_pct_df)
```

In [31]:
```
print(crime_toronto_df.columns)
print(crime_nyc_df.columns)
```

```
Index(['Neighbourhood', 'Assault_AVG', 'AutoTheft_AVG', 'BreakandEnter_AVG',
       'Robbery_AVG', 'TheftOver_AVG', 'Homicide_AVG', 'Shape__Area',
       'Latitude', 'Longitude'],
      dtype='object')
Index(['Neighbourhood', 'BreakandEnter_AVG', 'Assault_AVG', 'TheftOver_AVG',
       'AutoTheft_AVG', 'Homicide_AVG', 'Robbery_AVG', 'Shape__Area',
       'Latitude', 'Longitude'],
      dtype='object')
```

- Finally, the central latitude and longitude for each precinct area were converted to custom neighbourhood names by using **reverse geocoding** from **Here** geocoder in geopy. The custom neighbourhood names were cleaned using regex substition.

```
from geopy.geocoders import Here
#set timeout to 5 sec for slow interent connection and restrict search to country Canada ("CA")
geolocator_reverse = Here(user_agent="NYC_precinct_address",
                 timeout = 5,
                 #country_codes='CA',
                 app_id='dGiVRxoqi7Hhs95261Vk',
                 app_code='JOYxudJfw6OPtI7DmPLGLQ')

# using tqdm library for progress bar
from tqdm import tqdm
tqdm.pandas()

#use Rate limiter for delayed fetching of geocodes to avoid errors
from geopy.extra.rate_limiter import RateLimiter
geocode_reverse = RateLimiter(geolocator_reverse.reverse, min_delay_seconds=0.5,max_retries=5)

crime_nyc_df['Neighbourhood'] = crime_nyc_df.progress_apply(lambda x: geocode_reverse((x.Latitude,x.Longitude
```

```
100%|██████████| 77/77 [05:28<00:00,  4.27s/it]
```

```
In [33]: crime_nyc_df['Neighbourhood']  = crime_nyc_df['Neighbourhood'].apply(lambda loc: loc.address)
```

```
In [34]: crime_nyc_df.replace(to_replace=r", United States, (\w.).+, NY [0-9]+, USA",value="",regex=True,inplace=True)
```

```
In [35]: crime_nyc_df.shape
Out[35]: (77, 10)
```

3.2 Exploratory Data Analysis

3.2.1 Toronto

- Plot Toronto neighbourhoods on map using Nominatim and Folium

Part 6: Plot Toronto neighbourhoods on Map

```
In [36]: print('The dataframe has {} neighbourhoods.'.format(crime_toronto_df.shape[0]))
```

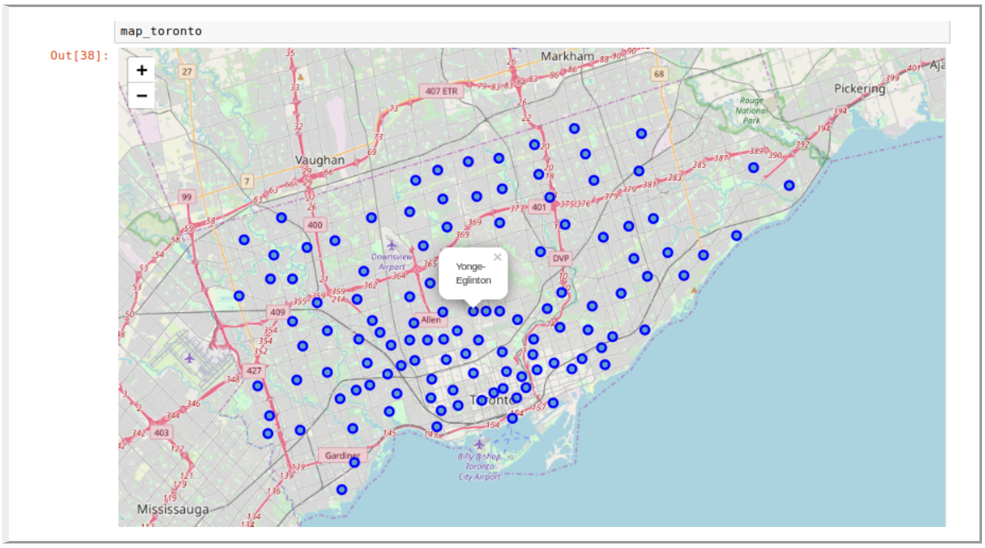
```
The dataframe has 140 neighbourhoods.
```

```
In [37]: from geopy.geocoders import Nominatim

address = 'Toronto'

geolocator = Nominatim(user_agent="toronto_explorer")
location_toronto = geolocator.geocode(address)
latitude_toronto = location_toronto.latitude
longitude_toronto = location_toronto.longitude
print('The geograpical coordinate of Toronto are {}, {}.'.format(latitude_toronto, longitude_toronto))
```

```
The geograpical coordinate of Toronto are 43.653963, -79.387207.
```

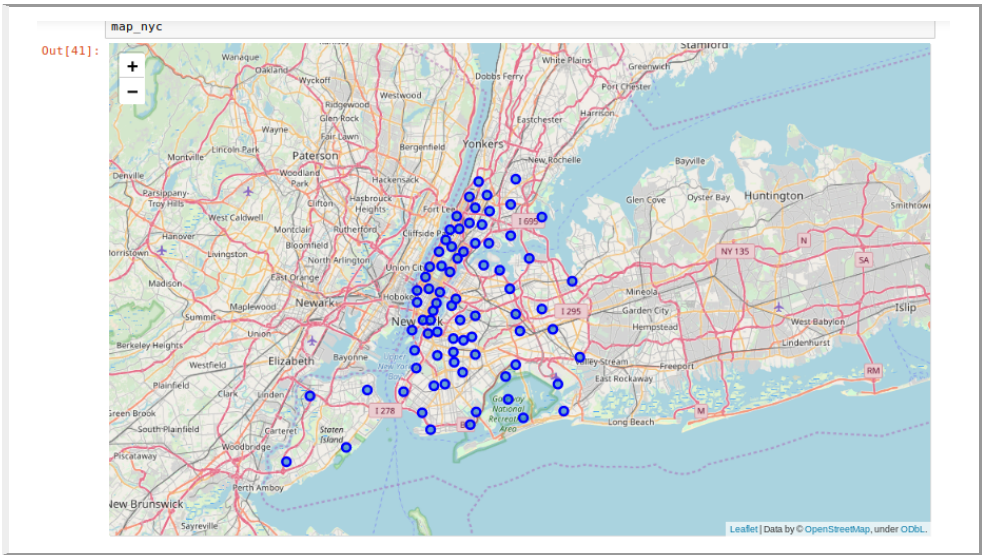
```
In [38]: map_toronto = folium.Map(location=[latitude_toronto, longitude_toronto], zoom_start=10)

# add markers to map
for lat, long, neighbourhood in zip(crime_toronto_df['Latitude'], crime_toronto_df['Longitude'],
                                    crime_toronto_df['Neighbourhood']):
    #label = '{}, {}'.format(neighbourhood, borough)
    label = '{}'.format(neighbourhood)
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, long],
        radius=5,
        popup=label,
        color='blue',
        fill=True,
        fill_color='#3186cc',
        fill_opacity=0.7,
```

The Toronto Neighbourhood Map



- Similarly plot new York neighbourhoods using Folium and Nominatim

The New York Neighbourhood Map



- Explore First neighbourhood in Toronto by fetching 100 top venues within a radius of 500 m using **Foursquare** api.

```
In [45]: LIMIT = 100 # limit of number of venues returned by Foursquare API

         radius = 500 # define radius
         url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={},{}&radius={}&limit
             CLIENT_ID,
             CLIENT_SECRET,
             VERSION,
             neighbourhood_latitude,
             neighbourhood_longitude,
             radius,
             LIMIT)

In [46]: results = requests.get(url).json()
         results
```

Out[46]:


```
{'meta': {'code': 200, 'requestId': '5da92e53d03360002cdf1520'},
 'response': {'suggestedFilters': {'header': 'Tap to show:',
   'filters': [{'name': 'Open now', 'key': 'openNow'}]},
  'headerLocation': 'Deer Park',
  'headerFullLocation': 'Deer Park, Toronto',
  'headerLocationGranularity': 'neighborhood',
  'totalResults': 45,
  'suggestedBounds': {'ne': {'lat': 43.693330004500005,
    'lng': -79.39265842323408},
   'sw': {'lat': 43.6843299955, 'lng': -79.40508157676592}},
  'groups': [{'type': 'Recommended Places',
    'name': 'recommended',
    'items': [{'reasons': {'count': 0,
       'items': [{'summary': 'This spot is popular',
         'type': 'general',
         'reasonName': 'globalInteractionReason'}]},
      'venue': {'id': '55c78cef498ec4095e9fba41',
       'name': 'LCBO',
```


- We can see that only 45 popular venues were returned by Foursquare for Yonge-St.Clair

```
In [48]: 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()
```

Out[48]:

	name	categories	lat	lng
0	LCBO	Liquor Store	43.686991	-79.399238
1	The Market By Longo's	Supermarket	43.686711	-79.399536
2	Cava Restaurant	Tapas Restaurant	43.689809	-79.394932
3	DAVIDsTEA	Tea Room	43.688376	-79.394158
4	The Bagel House	Bagel Shop	43.687374	-79.393696

```
In [49]: print('{} venues were returned by Foursquare.'.format(nearby_venues.shape[0]))

         45 venues were returned by Foursquare.
```

- Define a custom function get_nearby_venues to get Venue, Venue Latitude, Venue Longitude, Venue Category for each neighbourhood in Toronto Dataset. Apply the function to all rows of toronto dataset.

```
In [51]: crime_toronto_df_venues = getNearbyVenues(names=crime_toronto_df['Neighbourhood'],
                                                  latitudes=crime_toronto_df['Latitude'],
                                                  longitudes=crime_toronto_df['Longitude'],
                                                  radius=radius)

         0 : Yonge-St.Clair
         1 : York University Heights
         2 : Lansing-Westgate
         3 : Yorkdale-Glen Park
         4 : Stonegate-Queensway
         5 : Tam O'Shanter-Sullivan
         6 : The Beaches
         7 : Thistletown-Beaumond Heights
         8 : Thorncliffe Park
         9 : Danforth East York
         10 : Humewood-Cedarvale
         11 : Islington-City Centre West
         12 : Danforth
         13 : Rustic
         14 : Scarborough Village
         15 : South Parkdale
         16 : South Riverdale
         17 : St.Andrew-Windfields
         18 : Taylor-Massey
```

- Check the number of venues returned for each neighbourhood and count the total number of categories of all venues in Toronto dataset. We can see that total 280 categories were returned for Toronto neighbourhood popular venues.

```
In [53]: crime_toronto_df_venues.groupby('Neighbourhood',sort=False).count()
```

Out[53]:

Neighbourhood	Neighbourhood Latitude	Neighbourhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
Yonge-St.Clair	45	45	45	45	45	45
York University Heights	8	8	8	8	8	8
Lansing-Westgate	2	2	2	2	2	2
Yorkdale-Glen Park	14	14	14	14	14	14
Stonegate-Queensway	1	1	1	1	1	1
Tam O'Shanter-Sullivan	7	7	7	7	7	7
The Beaches	23	23	23	23	23	23
Thistletown-Beaumond Heights	11	11	11	11	11	11
Thorncliffe Park	20	20	20	20	20	20
Danforth East York	37	37	37	37	37	37

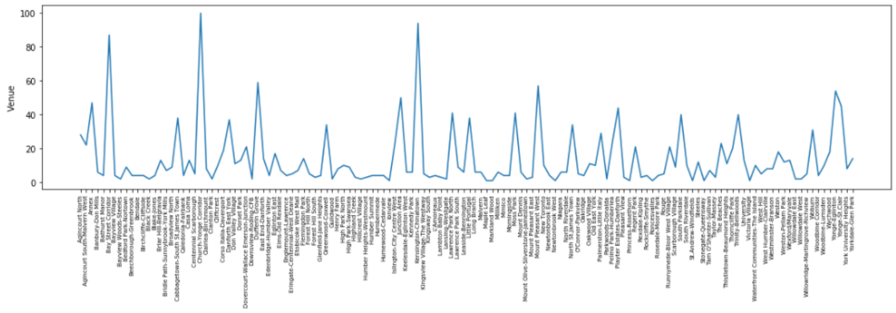
Let's find out how many unique categories can be curated from all the returned venues

```
In [54]: print('There are {} unique categories.'.format(len(crime_toronto_df_venues['Venue Category'].unique())))

There are 280 unique categories.
```

- Plot the number of venues for each neighbourhood. We can that their is huge variation in number of venues for each neighbourhood of Toronto.

```
import seaborn as sns
count_toronto = crime_toronto_df_venues.groupby('Neighbourhood',sort=False).count()
plt.figure(figsize=(18,4))
plt.xticks(rotation=90,**{'fontsize':'x-small'})
sns.lineplot(x=count_toronto.index.values,y=count_toronto['Venue'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f814ee9ea58>
```



- Converting venue category in numeric variable using One Hot Encoding(pandas dummy variables) and then grouping cateogry-wise total venues for each neighbourhood. Then this venue dataframe would be merged with crime dataset to get final dataframe ready for clustering and analysis.

	Neighbourhood	Accessories Store	Afghan Restaurant	African Restaurant	Airport Service	American Restaurant	Animal Shelter	Antique Shop	Arcade	Argentinian Restaurant	Art Gallery	Art Museum	& Crafts Store	Asian Restaurant
0	Yonge-St.Clair	0	0	0	0	1	0	0	0	0	0	0	0	0
1	York University Heights	0	0	0	0	0	0	0	0	0	0	0	0	0
2	Lansing-Westgate	0	0	0	0	0	0	0	0	0	0	0	0	0
3	Yorkdale-Glen Park	0	0	0	0	0	0	0	0	0	0	0	0	0
4	Stonegate-Queensway	0	0	0	0	0	0	0	0	0	0	0	0	0

Let's confirm the new size

```
In [155]: crime_toronto_df_grouped.shape
Out[155]: (136, 281)
```

Join crime data from crime_toronto_df to venues data

```
In [156]: crime_toronto_df_grouped = crime_toronto_df_grouped.merge(crime_toronto_df)
```

3.2.2 New York

- Entire process from steps 1 to 8 is repeated for new York dataframe too beginning with getting venues using Foursquare.

Part 9 : Retreive and append Venues data for New York neighbourhoods

```
In [61]: crime_nyc_df_venues = getNearbyVenues(names=crime_nyc_df['Neighbourhood'],
                                    latitudes=crime_nyc_df['Latitude'],
                                    longitudes=crime_nyc_df['Longitude']
                                   )

0 : Hugh L Carey Tunnel, New York, NY 10004
1 : 7 Catherine St, New York, NY 10038
2 : 744 Greenwich St, New York, NY 10014
3 : 271 Henry St, New York, NY 10002
4 : 119 Avenue B, New York, NY 10009
5 : 643 W 24th St, New York, NY 10011
6 : 7 Peter Cooper Rd, New York, NY 10010
7 : 107 W 37th St, New York, NY 10018
8 : 489 E 41st St, New York, NY 10017
9 : 548 W 53rd St, New York, NY 10019
10 : 338 E 80th St. New York. NY 10075
```

- Final grouped dataset of crime and category wise venues for New York Neighbourhoods

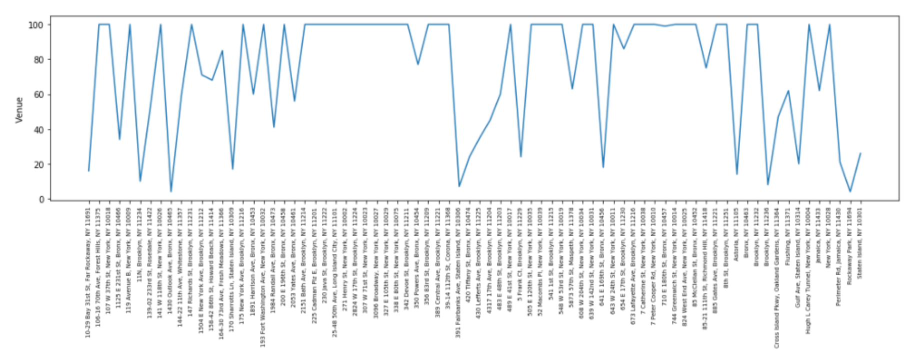
Let's confirm the new size

```
In [158]: crime_nyc_df_grouped.shape
Out[158]: (76, 392)
```

Join crime data from crime_nyc_df to venues data

```
In [159]: crime_nyc_df_grouped = crime_nyc_df_grouped.merge(crime_nyc_df)

In [160]: crime_nyc_df_grouped.shape
Out[160]: (76, 401)
```

- Plot the number of venues for each neighbourhood. We can that several neighbourhoods in New York have hit the ceiling of 100 popular venues per neighbourhood.



3.2.3 Dropping Sparse Columns of venue cateogory which are present in 2 or less than 2 neighbourhoods

Two many features in a dataset can affect machine learning and present difficulties in proper clustering. Hence features ('venue categories') which are present (non-zero) for two or less than two neighbourhoods were dropped to condense the dataset.

```
In [160]: crime_nyc_df_grouped.shape
Out[160]: (76, 401)

In [161]: crime_toronto_df_grouped.shape
Out[161]: (136, 290)

In [162]: sparse_cols=crime_nyc_df_grouped.columns[crime_nyc_df_grouped[crime_nyc_df_grouped==0].count()>crime_nyc_df_grouped.shape
          crime_nyc_df_grouped.drop(columns=sparse_cols,inplace=True)
          crime_nyc_df_grouped.shape
Out[162]: (76, 300)

In [163]: sparse_cols=crime_toronto_df_grouped.columns[crime_toronto_df_grouped[crime_toronto_df_grouped==0].count()==crime_toronto
          crime_toronto_df_grouped.drop(columns=sparse_cols,inplace=True)
          crime_toronto_df_grouped.shape
Out[163]: (136, 240)
```

We can see that 101(401 - 300) columns were dropped from NYC dataset and 50 (290-240) columns were dropped from Toronto dataset.

3.3 Inferential Statistical Testing

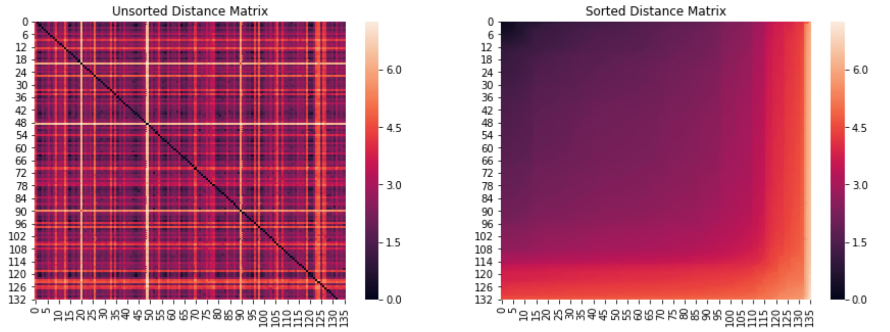
3.3.1 Plotting distance matrices for Toronto and New York

- We define a custom function to plot distance matrix of Neighbourhood Observations using cdist() function in scipy.spatial.distance library and plot the resulting distance matrix using seaborn.heatmap(). Anohter subplot is plotted that shows the heatmap of distance matrix sorted along rows and columns

```
: def plot_dist_matrix(df):

      from scipy.spatial.distance import cdist
      import seaborn as sns
      matrix = MinMaxScaler().fit_transform(df)
      Y = cdist(matrix,matrix)

      fig, (ax1, ax2) = plt.subplots(1,2)
      fig.set_figwidth(15)
      fig.set_figheight(5)

      sns.heatmap(Y,ax=ax1)
      sns.heatmap(np.sort(np.sort(Y,axis=0),axis=1),ax=ax2)

      ax1.set_title('Unsorted Distance Matrix')
      ax2.set_title('Sorted Distance Matrix')

      plt.show
```

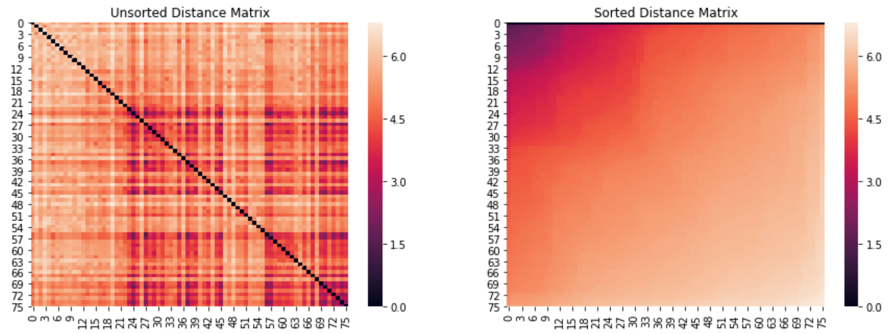
- Plotting Toronto dataset using plot_dist_matrix

```python
crime_toronto_df_grouped_clustering = crime_toronto_df_grouped.drop('Neighbourhood', 1)
plot_dist_matrix(crime_toronto_df_grouped_clustering)
```



- Plotting similar distance matrices for New York data

```python
crime_nyc_df_grouped_clustering = crime_nyc_df_grouped.drop('Neighbourhood', 1)
plot_dist_matrix(crime_nyc_df_grouped_clustering)
```



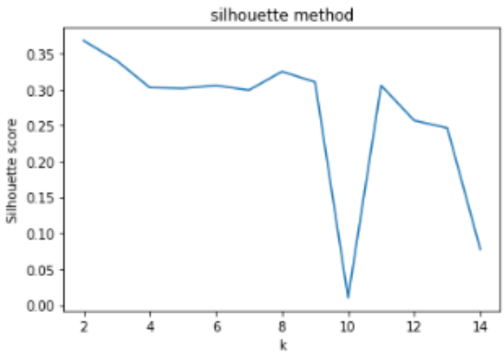
On comparison of distance matrix plots of New York and Toronto we can clearly observe that pairwise distances between neighbourhoods in Toronto are in much smaller range (almost 70% in 0-3) compared to pairwise distances of New York neighbourhoods which are in a larger range (almost 70 % in 4.5 and above). We can infer that similarity between neighbourhoods in Toronto is higher than similarity between those in New York.

3.3.2 Finding optimum value of n_cluster for kmeans algorithm

- Before using kmeans for clustering of similar neighbourhoods, we needs to find optimum number of clusters to get the best results. In order to find ideal number of clusters we need to run kmeans for varying range of clusters. For the purpose of this exercise, we use range of 2-15. We then define a custom function to plot silhouette scores for each test value of K. Note that features are scaled prior to clustering fit so that no features biases the distance matrix.

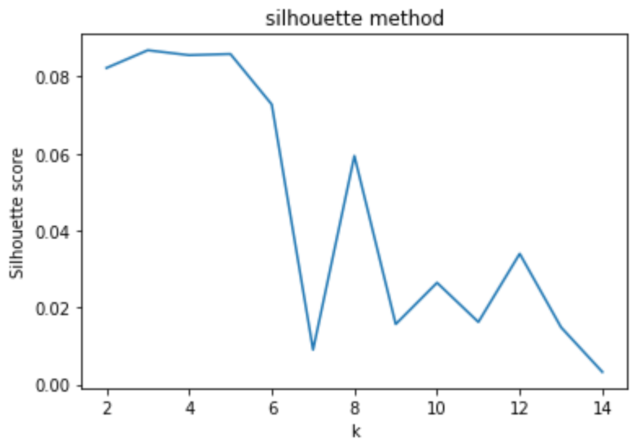
```
def plot_silhouette(df,K=range(2,15)):
    from sklearn.metrics import silhouette_score
    silhouette = []
    for k in K:
        mms=MinMaxScaler()
        mms.fit(df)
        X = mms.transform(df)
        km=KMeans(n_clusters=k,n_init=200)
        km.fit(X)
        silhouette.append(silhouette_score(X,km.labels_))
    plt.plot(K, silhouette,)
    plt.xlabel('k')
    plt.ylabel('Silhouette score')
    plt.title('silhouette method')
    plt.show()
```

```
plot_silhouette(crime_toronto_df_grouped_clustering)
```



As per the graph of silhouette scores, any values cluster(K) in the range of 2-8 will give a good score. We choose kclusters to be 6 for clsuering of Toronto Neighbourhoods.

- Same process of plotting silhouette scores was followed for finding optimum value of n_clusters for Kmeans for New York neighbourhoods. The resulting plot is



We can see that best silhouette scores are obtained for k in the range of 2 to 5. However, the best silhouette scores are not more than 0.09 which is way too low than corresponding scores for Toronto neighobourhoods (0.30 - 0.35). This shows that with the presently generated features , clustering for New York neighbourhoods may not be optimum. This can also be understood from the heatmap of distance matrix in section 3.3.1 (1) where we can see that very few neighbourhoods are closely located (~ few dark shades) and most of the neihgbourhoods are distant in feature space. However, for the purpose of this exercise,, we will use value of 5 for n_clusters.

3.4 Machine Learning

3.4.1 Clustering Toronto neighbourhoods

First of all, we cluster the neighbourhoods within the city of Toronto using kmeans. Initially, kmeans was run with parameters values:-

1. n_clusters=kclusters (6)
2. random_state=0,
3. n_init = N_INIT(20)
4. algorithm='auto'

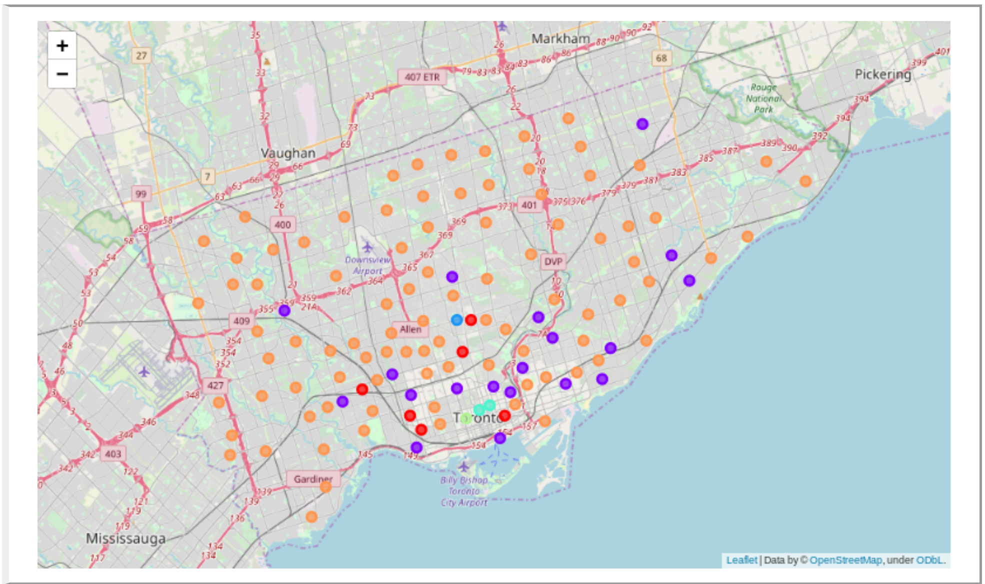
During code execution it was observed that clusters labels generated by kmeans were changing from run to run. This is probably because neighbourhoods are tightly packed in the feature-space as observed in the heatmap plot of distance matrix (section 3.3.1). In order to get consistent cluster labels, we increased the n_init parameter to 200 to enable maximum possible exploration through random initialization. Also algorithm used was forced as 'full'. The documentation for sklearn.cluster.KMeans says the following:- _**algorithm“auto”, “full” or “elkan”, default=”auto”** _K-means algorithm to use. The classical EM-style algorithm is “full”. The “elkan” variation is more efficient by using the triangle inequality, but currently doesn’t support sparse data. “auto” chooses “elkan” for dense data and “full” for sparse data.used the follwing parameters for final run of KMeans.

Since our dataset is highly sparse, hence algorithm used was forced as 'full'.

The final parameters used were as follows:-

1. n_clusters=kclusters (6)
2. random_state=0,
3. **n_init = N_INIT(200)**
4. **algorithm='full'**

After clustering, the cluster labels were merged in Toronto neighbourhood dataset and neighbourhoods were plotted using Folium with color-coding of markers as per cluster labels.



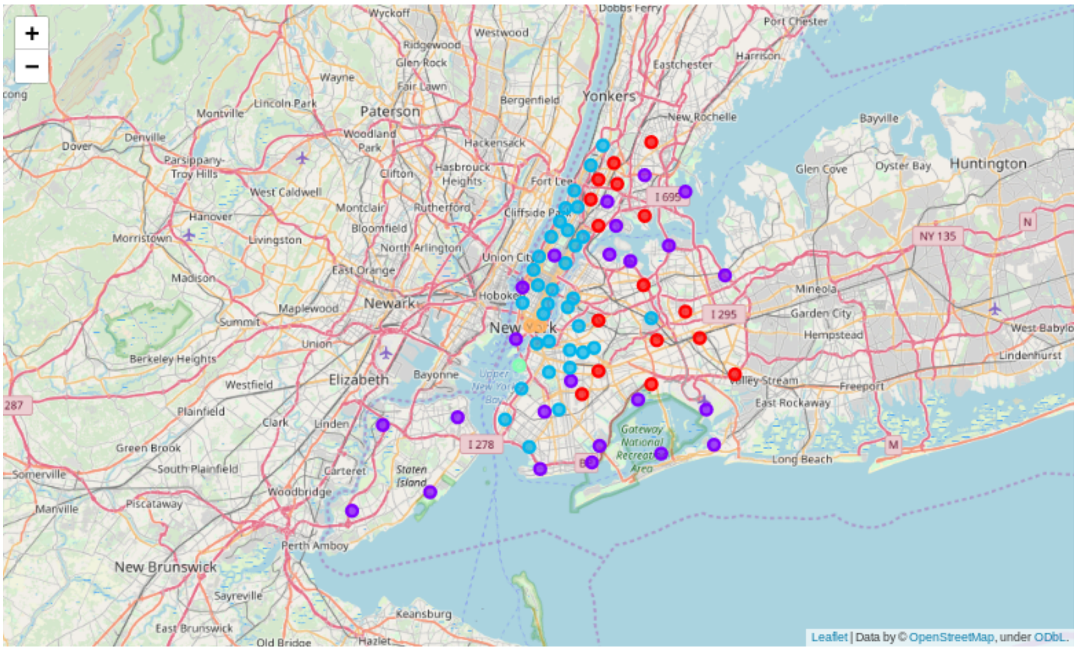
3.4.2 Clustering New York neighbourhoods

The Kmeans clustering of New York neighbourhoods was run with the same parameters as for Toronto neighbourhoods clustering except for n_cluster parameter. n_cluster parameter was set to 5 as determined during silhouette analysis.

The final parameters used were as follows:-

1. n_clusters=kclusters (5)
2. random_state=0,
3. n_init = N_INIT(200)
4. algorithm='full'

After clustering, the cluster labels were merged in New York neighbourhood dataset and neighbourhoods were plotted using Folium with color-coding of markers as per cluster labels.



3.4.3 Compare Neighbourhoods between New York and Toronto

- In order to compare and find similar neighbourhoods between New York and Toronto, we ensure that both datasets have same column names ("Venue Category") so that the two dataframes can be joined together. Therefore we find the columns (venue categories) in present in Toronto dataframe but not present in New York dataframe (using numpy.setdiff1d function) and add the missing columns to new York dataframe. The new columns are initialized to 0(zero).

```
In [299]: #Find columns present in Toronto but not in New York
          missing_in_nyc = np.setdiff1d(crime_toronto_df_grouped.columns.values,crime_nyc_df_grouped.columns.values)
          missing_in_nyc

Out[299]: array(['Afghan Restaurant', 'Animal Shelter', 'Auto Dealership',
                 'Automotive Shop', 'Baby Store', 'Beach Bar', 'Belgian Restaurant',
                 'Bike Rental / Bike Share', 'Bike Shop', 'Bridal Shop', 'Buffet',
                 'Business Service', 'Castle', 'Chocolate Shop', 'Church',
                 'Churrascaria', 'College Football Field', 'College Quad',
                 'College Rec Center', 'College Stadium', 'Costume Shop',
                 'Coworking Space', 'Creperie', 'Curling Ice',
                 'Egyptian Restaurant', 'Field', 'Food Service', 'Football Stadium',
                 'Golf Driving Range', 'Hockey Arena', 'Hostel',
                 'Hotpot Restaurant', 'Indian Chinese Restaurant', 'Irish Pub',
                 'Lake', 'Library', 'Locksmith', 'Moving Target', 'Music Store',
                 'Nightlife Spot', 'North Indian Restaurant', 'Other Repair Shop',
                 'Pakistani Restaurant', 'Pastry Shop', 'Persian Restaurant',
                 'Platform', 'Poke Place', 'Pool Hall', 'Portuguese Restaurant',
                 'Poutine Place', 'Rock Club', 'Soccer Stadium', 'Social Club',
                 'South Indian Restaurant', 'Student Center', 'Swiss Restaurant',
                 'Syrian Restaurant', 'Transportation Service', 'University',
                 'Volleyball Court'], dtype=object)

In [300]: crime_nyc_df_grouped = pd.concat([crime_nyc_df_grouped,pd.DataFrame(columns=list(missing_in_nyc))],sort = Fal
          crime_nyc_df_grouped.fillna(0,inplace=True)
```

- Same process was repeated for Toronto neighbourhood dataset too.

```
In [301]: #Find columns present in New York but not in Toronto
          missing_in_toronto = np.setdiff1d(crime_nyc_df_grouped.columns.values,crime_toronto_df_grouped.columns.values
          missing_in_toronto

Out[301]: array(['African Restaurant', 'Airport Lounge', 'Arepa Restaurant',
                 'Australian Restaurant', 'Austrian Restaurant', 'Baseball Stadium',
                 'Basketball Court', 'Beach', 'Beer Bar', 'Beer Garden',
                 'Big Box Store', 'Bike Trail', 'Boat or Ferry', 'Boxing Gym',
                 'Bridge', 'Building', 'Burrito Place', 'Bus Station',
                 'Cajun / Creole Restaurant', 'Cantonese Restaurant',
                 'Caucasian Restaurant', 'Check Cashing Service', 'Climbing Gym',
                 'Comic Shop', 'Community Center', 'Construction & Landscaping',
                 'Cuban Restaurant', 'Cycle Studio', 'Distillery', 'Dive Bar',
                 'Eastern European Restaurant', 'Empanada Restaurant',
                 'English Restaurant', 'Ethiopian Restaurant', 'Event Space',
                 'Exhibit', 'Factory', 'Filipino Restaurant',
                 'Financial or Legal Service', 'Fish & Chips Shop', 'Food',
                 'Food Court', 'Fountain', 'Frozen Yogurt Shop',
                 'Fruit & Vegetable Store', 'Garden', 'Garden Center', 'Gay Bar',
                 'German Restaurant', 'Gift Shop', 'Gourmet Shop', 'Gymnastics Gym',
                 'Halal Restaurant', 'Harbor / Marina', 'Hardware Store',
                 'Health & Beauty Service', 'Health Food Store', 'Heliport',
                 'Historic Site', 'Hobby Shop', 'Hookah Bar', 'Hot Dog Joint',

In [302]: crime_toronto_df_grouped = pd.concat([crime_toronto_df_grouped,pd.DataFrame(columns=list(missing_in_toronto))
          crime_toronto_df_grouped.fillna(0,inplace=True)
          crime_toronto_df_grouped
```

- Some feature engineering was done. A column of city labels was added to each dtatset before conactenation. Also since latitudes and longitudes of both New York and Toronto are different ranges, centering of latitudes and longitudes of neighbourhoods in each dataset was done prior to concatenation. Centering was done by subtracting the lat/long of respective city from lat and long of each neighbourhood. Susequently sparse columns with less than two non-zero values were dropped.

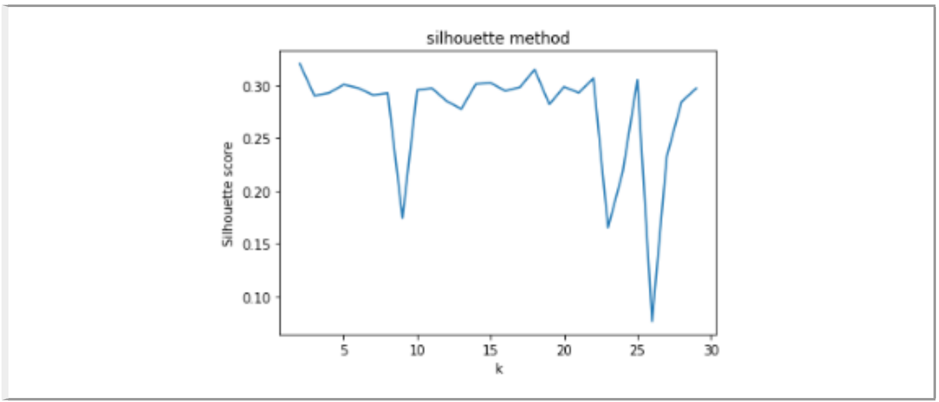
```
In [305]: crime_nyc_df_grouped['City'] = 'NYC'
          crime_toronto_df_grouped['City'] = 'Toronto'

In [306]: crime_nyc_df_grouped['Latitude'] = crime_nyc_df_grouped['Latitude'] - latitude_nyc
          crime_nyc_df_grouped['Longitude'] = crime_nyc_df_grouped['Longitude'] - longitude_nyc

In [307]: crime_toronto_df_grouped['Latitude'] = crime_toronto_df_grouped['Latitude'] - latitude_toronto
          crime_toronto_df_grouped['Longitude'] = crime_toronto_df_grouped['Longitude'] - longitude_toronto

In [308]: crime_nyc_df_grouped.shape

Out[308]: (76, 362)

In [309]: crime_toronto_df_grouped.shape

Out[309]: (136, 362)

In [310]: combined_df = pd.concat([crime_nyc_df_grouped,crime_toronto_df_grouped],sort=False)
          print(combined_df.shape)

          ...

In [311]: sparse_cols=combined_df.columns[combined_df[combined_df==0].count()==combined_df.shape[0]-2].values
          combined_df.drop(columns=sparse_cols,inplace=True)
          combined_df.shape
```

- The final combined dataframe has a 212 rows and 326 columns.

```
In [311]: sparse_cols=combined_df.columns[combined_df[combined_df==0].count()==combined_df.shape[0]-2].values
          combined_df.drop(columns=sparse_cols,inplace=True)
          combined_df.shape

Out[311]: (212, 326)
```

We can see that 36 sparse columns (362 -326) were dropped from the combined dataset.

- Silhouette analysis for combined dataframe was done using plot_silhouette custom function with test values of n_clusters from 2-30.



We can see from the silhoutte score plot that for k ranging between 2-20 silhouette score remain roughly between 0.28 to 0.32. However, there is a sharp drop at k values of 8. We also know that best values of K for New York dataset ranged from 2-5 only. Hence, we decide to run KMeans clustering for combined dataframe with n_cluster value of 5 only.

- After KMeans clustering, cluster labels so obtained were merged with the combined dataframe and neighbourhoods were grouped according to cluster labels and city labels in to the follwing pivot table. IPython.html and Dataframe.style.set_properties() were used for pretty rendering of pivot table.

City Cluster Labels	NYC	Toronto
0	Hugh L Carey Tunnel, New York, NY 10004 744 Greenwich St, New York, NY 10014 119 Avenue B, New York, NY 10009 7 Peter Cooper Rd, New York, NY 10010 107 W 37th St, New York, NY 10018 489 E 41st St, New York, NY 10017 548 W 53rd St, New York, NY 10019 338 E 80th St, New York, NY 10075 307 W 71st St, New York, NY 10023 327 E 105th St, New York, NY 10029 824 West End Ave, New York, NY 10025 3096 Broadway, New York, NY 10027 141 W 118th St, New York, NY 10026 193 Fort Washington Ave, New York, NY 10032 608 W 204th St, New York, NY 10034 356 83rd St, Brooklyn, NY 11209 654 E 17th St, Brooklyn, NY 11230 Brooklyn, NY 11232 147 Richards St, Brooklyn, NY 11231 175 New York Ave, Brooklyn, NY 11216 541 1st St, Brooklyn, NY 11215 673 Lafayette Ave, Brooklyn, NY 11216 885 Gates Ave, Brooklyn, NY 11221 389 Central Ave, Brooklyn, NY 11221 225 Cadman Plz E, Brooklyn, NY 11201 8th St, Brooklyn, NY 11251 342 Devoe St, Brooklyn, NY 11211 230 Java St, Brooklyn, NY 11222 25-48 50th Ave, Long Island City, NY 11101 106-16 70th Ave, Forest Hills, NY 11375	Church-Yonge Corridor Kensington-Chinatown Bay Street Corridor
1	505 E 120th St, New York, NY 10035 350 Powers Ave, Bronx, NY 10454 1984 Randall Ave, Bronx, NY 10473 85 McClellan St, Bronx, NY 10452 1891 Harrison Ave, Bronx, NY 10453 1125 E 231st St, Bronx, NY 10466 710 E 180th St, Bronx, NY 10457 2053 Yates Ave, Bronx, NY 10461 Bronx, NY 10463 200 E 196th St, Bronx, NY 10458 2824 W 17th St, Brooklyn, NY 11224 2151 Bath Ave, Brooklyn, NY 11214 483 E 48th St, Brooklyn, NY 11203 1504 E New York Ave, Brooklyn, NY 11212 85-21 111th St, Richmond Hill, NY 11418 Jamaica, NY 11433 5873 57th St, Maspeth, NY 11378 139-02 233rd St, Rosedale, NY 11422 158-42 86th St, Howard Beach, NY 11414 164-30 73rd Ave, Fresh Meadows, NY 11366 144-22 11th Ave, Whitestone, NY 11357 39-14 112th St, Corona, NY 11368	nan
2	639 W 142nd St, New York, NY 10031 52 Macombs Pl, New York,	nan

- We can see from the table above that clusters 2 and 3 have only one neighbourhood from New York and none from Toronto. In fact all neighbourhoods of Toronto except one have been grouped in cluster 4. This could be because neighbourhoods of Toronto were much tighly packed in n-dimensional feature space compared to neighbourhood distances (in feature space) of New York.

3.4.4 Comparing various clustering algorithms on combined dataframe

We can see in previous section that performance of KMeans algorithm in finding similar neighbourhoods between New York and Toronto was not satisfactory. Hence, it was decided to a test run of all available clustering algorithms in sklearn.cluster library on the combined and compare their performance in clustering similar neighbourhoods between the two cities.

The following algorithms were tried:-

1. MiniBatchKMeans
2. AffinityPropagation
3. MeanShift
4. SpectralClustering
5. Ward
6. AgglomerativeClustering
7. DBSCAN
8. OPTICS
9. Birch
10. GaussianMixture
11. SpectralBiclustering
12. SpectralCoclustering

The final pivot table output containing number of neighbourhoods in each cluster for two cities for each algorithm is displayed below.

```
RunningMiniBatchKMeans(batch_size=100, compute_labels=True, ini
t='k-means++',
                init_size=None, max_iter=100, max_no_improvemen
t=10,
                n_clusters=5, n_init=3, random_state=None,
                reassignment_ratio=0.01, tol=0.0, verbose=0)
```

City	NYC	Toronto
Cluster Labels		
0	9	nan
1	4	nan
2	2	nan
3	19	133
4	42	3

```
RunningAffinityPropagation(affinity='euclidean', convergence_it
er=15, copy=True,
                    damping=0.9, max_iter=200, preference=-120
0, verbose=False)
```

City	NYC	Toronto
Cluster Labels		
0	1	nan
1	1	nan
2	1	nan
3	nan	1
4	73	135

```
RunningMeanShift(bandwidth=16.97339757396255, bin_seeding=True,
cluster_all=True,
          min_bin_freq=1, n_jobs=None, seeds=None)
```

City	NYC	Toronto
Cluster Labels		
0	76	119
1	nan	1
2	nan	1
3	nan	1
4	nan	1
5	nan	1
6	nan	1
7	nan	1
8	nan	1
9	nan	1
10	nan	1
11	nan	1
12	nan	1

On comparing the summary tables of clustering done by various algorithms, we can see that almost all the clustering algorithms cluster most of the neighbourhoods in one or two clusters with 99% of Toronto neighbourhoods being clubbed in one cluster. DBSCAN algorithm which does not take any initial value of n_clusters and tries to find inherent cluster structure classifies all neighbourhoods of both New York and Toronto in one cluster.

However, SpectralClustering produces the most granular clustering with all the clusters containing some neighbourhoods of both cities New York and Toronto. The results of SpectralClustering are repoduced below.

Cluster Labels	NYC	Toronto
0	7	9
1	15	7
2	24	59
3	28	52
4	2	9

3.4.5 Inductive Clustering

1. When the result of clustering naturally induces functions for classification on the whole space of interest, the method is called that of inductive clustering. In contrast, a method is called non-inductive, if it does not induce such a function. Many clustering algorithms are not inductive and so cannot be directly applied to new data samples without recomputing the clustering, which may be intractable. Instead, we can use clustering to then learn an inductive model with a classifier.
2. For this exercise, we shall use Inductive Clustering of Agglomerative clustering and KneighboursClassifier. First step was to use AgglomerativeClustering model on New York dataframe to generate cluster labels for New York data. Then a sequence of Aggomerative clustering and KneighboursClassifier was used to predict the cluster labels for Toronto neighbourhoods.
3. The parameters finalised for Agglomerative Clustering after fine-tuning the models are as follows:-
    - n_clusters=5
    - affinity ='cosine'
    - connectivity= kneighbors_graph
    - linkage = 'complete'
    - compute_full_tree=True
4. The significance of above parameter is as follows:-
    - **complete** linkage minimizes the maximum distance between observations of pairs of clusters
    - **connecitivity** constraints are useful to impose a certain local structure (only adjacent clusters can be merged together)
    - **cosine** distance works because it is invariant to global scalings of the signal
    - **compute_full_tree** was forced as True. This is because the documentation for AgglomerativeClustering says when varying the number of clusters and using caching, it may be advantageous to compute the full tree.
5. The cluster labels generated by Inductive Clustering were merged with combined dataframe. With Dataframe.groupby(), aggregate() and unstack() methods, a pivot table was generated classifying all the neighbourhoods of New York and Toronto.

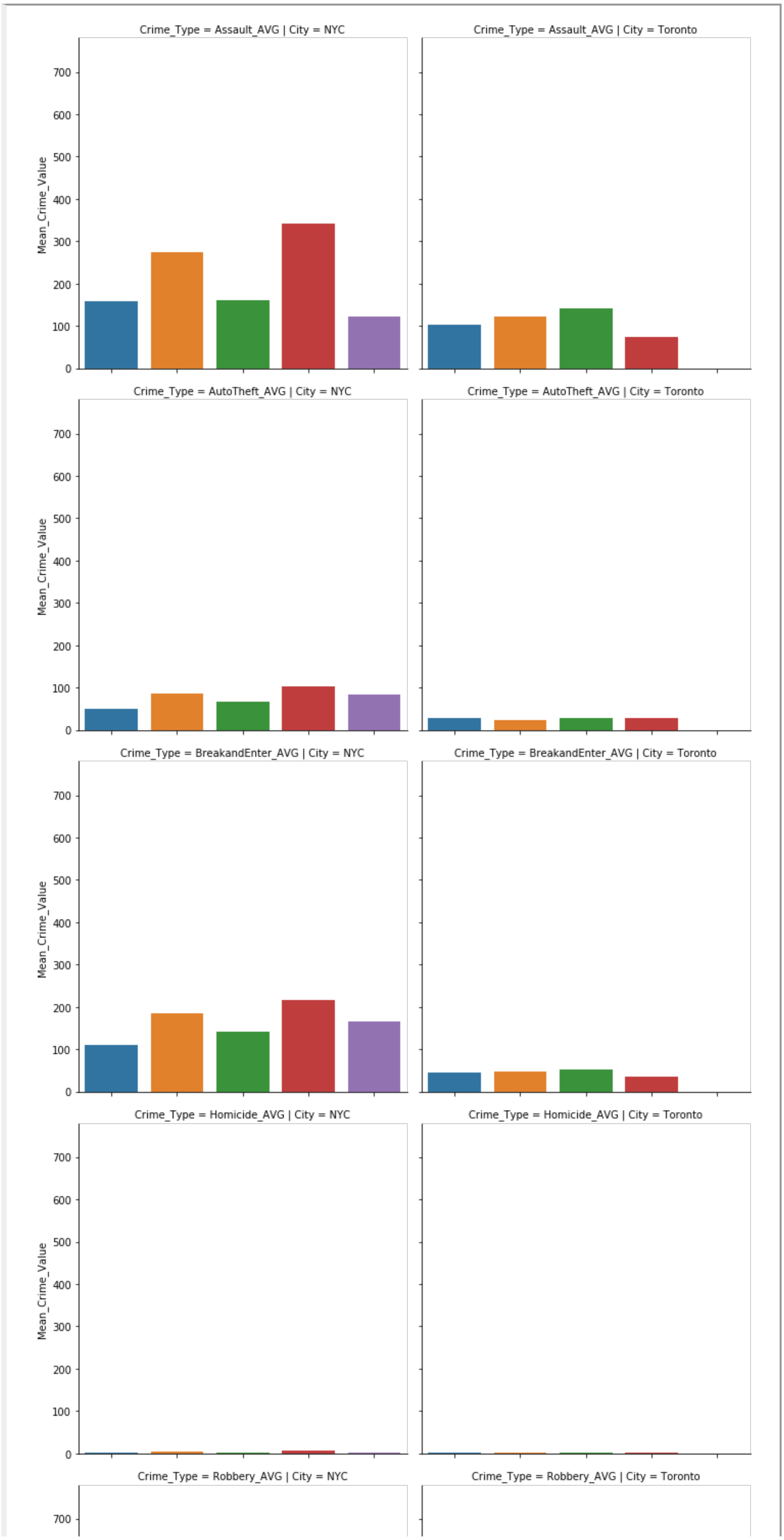
City	NYC	Toronto
Cluster Labels		
0	4	8
1	54	45
2	7	81
3	8	2
4	3	nan

4. Results

The final pivot table displaying the clustering of neighbourhoods between New York and Toronto can be seen below.

City	NYC	Toronto
Cluster Labels		
		Stonegate-Queensway
		Keelesdale-Eglinton West
	2824 W 17th St, Brooklyn, NY 11224	Casa Loma
0	Gulf Ave, Staten Island, NY 10314	High Park-Swansea
	391 Fairbanks Ave, Staten Island, NY 10306	Downsview-Roding-CFB
	170 Sharrotts Ln, Staten Island, NY 10309	Willowdale West
		Don Valley Village
		Woodbine Corridor
	Hugh L Carey Tunnel, New York, NY 10004	
	7 Catherine St, New York, NY 10038	
	744 Greenwich St, New York, NY 10014	
	271 Henry St, New York, NY 10002	
	119 Avenue B, New York, NY 10009	
	7 Peter Cooper Rd, New York, NY 10010	Yonge-St.Clair
	107 W 37th St, New York, NY 10018	The Beaches
	107 W 37th St, New York, NY 10018	Thorncliffe Park
	489 E 41st St, New York, NY 10017	Danforth East York
	338 E 80th St, New York, NY 10075	Humewood-Cedarvale
	307 W 71st St, New York, NY 10023	South Parkdale
	New York, NY 10028	South Riverdale
	327 E 105th St, New York, NY 10029	Church-Yonge Corridor
	824 West End Ave, New York, NY 10025	Flemingdon Park
	505 E 120th St, New York, NY 10035	Corso Italia-Davenport
	3096 Broadway, New York, NY 10027	Junction Area
	141 W 118th St, New York, NY 10026	Etobicoke West Mall
	639 W 142nd St, New York, NY 10031	Greenwood-Coxwell
	52 Macombs Pl, New York, NY 10039	Trinity-Bellwoods
	193 Fort Washington Ave, New York, NY 10032	West Humber-Clairville
	608 W 204th St, New York, NY 10034	Kensington-Chinatown
	350 Powers Ave, Bronx, NY 10454	Annex
	420 Tiffany St, Bronx, NY 10474	University
	1984 Randall Ave, Bronx, NY 10473	Hillcrest Village
	85 McClellan St, Bronx, NY 10452	Highland Creek
	1891 Harrison Ave, Bronx, NY 10453	North St.James Town
	1125 E 231st St, Bronx, NY 10466	Wexford/Maryvale
1	710 E 180th St, Bronx, NY 10457	Elms-Old Rexdale
	2053 Yates Ave, Bronx, NY 10461	Agincourt North
	Bronx, NY 10463	Agincourt South-Malvern West
	200 E 196th St, Bronx, NY 10458	Regent Park
	2151 Bath Ave, Brooklyn, NY 11214	Weston-Pellam Park
	483 E 48th St, Brooklyn, NY 11203	Bay Street Corridor
	654 E 17th St, Brooklyn, NY 11230	Cabbagetown-South St.James Town
	430 Lefferts Ave, Brooklyn, NY 11225	Mount Pleasant West
	Brooklyn, NY 11232	East End-Danforth
	1504 E New York Ave, Brooklyn, NY 11212	Palmerston-Little Italy
	147 Richards St, Brooklyn, NY 11231	Playter Estates-Danforth
	175 New York Ave, Brooklyn, NY 11216	Woburn
	673 Lafayette Ave, Brooklyn, NY 11216	Woodbine-Lumsden
	885 Gates Ave, Brooklyn, NY 11221	Bayview Village
	389 Central Ave, Brooklyn, NY 11221	Weston
	342 Devoe St, Brooklyn, NY 11211	Long Branch
	230 Java St, Brooklyn, NY 11222	Dufferin Grove
	85-21 111th St, Richmond Hill, NY 11418	Lawrence Park North
	Jamaica, NY 11433	Yonge-Eglinton
	5873 57th St, Maspeth, NY 11378	Moss Park
	139-02 233rd St, Rosedale, NY 11422	Little Portugal
	158-42 86th St, Howard Beach, NY 11414	Wychwood
	164-30 73rd Ave, Fresh Meadows, NY 11366	Briar Hill-Belgravia
	25-48 50th Ave, Long Island City, NY 11101	
	144-22 11th Ave, Whitestone, NY 11357	

For interpreting the different cluster labels, seaborn library's catplot function was used to produce a barplot for each Crime_type and City. Results can be seen below



Crime_Type = Assault_AVG | City = NYC
Crime_Type = Assault_AVG | City = Toronto
Mean_Crime_Value
0
100
200
300
400
500
600
700
Crime_Type = AutoTheft_AVG | City = NYC
Crime_Type = AutoTheft_AVG | City = Toronto
Mean_Crime_Value
0
100
200
300
400
500
600
700
Crime_Type = BreakandEnter_AVG | City = NYC
Crime_Type = BreakandEnter_AVG | City = Toronto
Mean_Crime_Value
0
100
200
300
400
500
600
700
Crime_Type = Homicide_AVG | City = NYC
Crime_Type = Homicide_AVG | City = Toronto
Mean_Crime_Value
0
100
200
300
400
500
600
700
Crime_Type = Robbery_AVG | City = NYC
Crime_Type = Robbery_AVG | City = Toronto
700

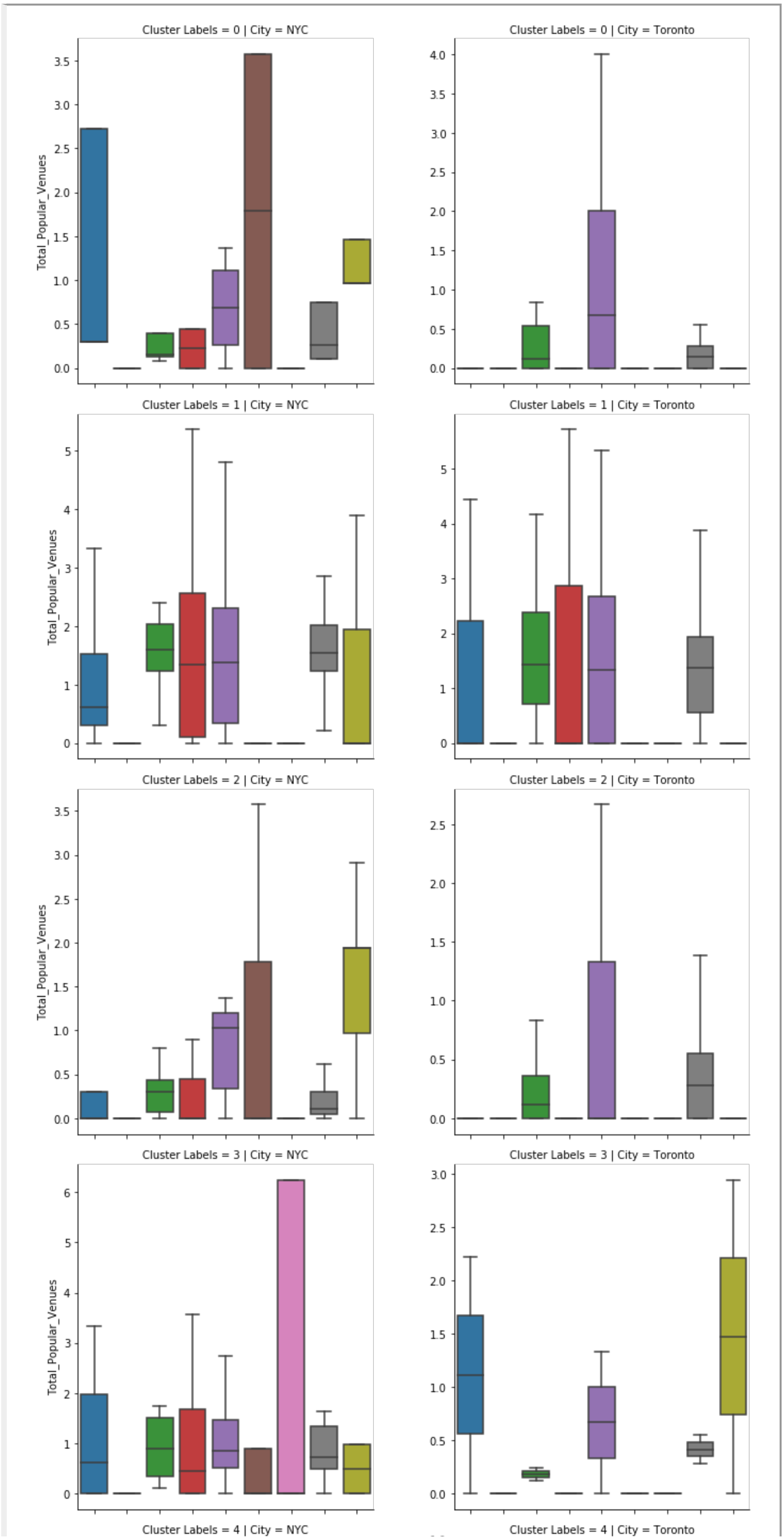
On the basis of crime rates, clusters can be categorized as follows:-

- Cluster 3 - Very High Crime Rate
- Cluster 1 - High Crime Rate
- Cluster 2 - Moderate Crime Rate
- Cluster 4 - Low Crime Rate
- Cluster 0 - Very Low Crime Rate

In order to categorize clusters on the basis of venues categories, following approach was used:-

- Fetch full category list from Foursquare using request.get() on this url (https://api.foursquare.com/v2/venues/categories)
- Full category list has 10 master venue categories under which all venue categories are classified.
  - Arts & Entertainment
  - College & University
  - Event
  - Food
  - Nightlife Spot
  - Outdoors & Recreation
  - Professional & Other Places
  - Residence
  - Shop & Service
  - Travel & Transport
- Create a dictionary of Venue Categories mapped to Master Category using a custom function image.png
- Use groupby on columns of combined Dataframe to condense all venue categories columns into 10 master category columns
  - Use Seaborn.catplot() function to plot a Boxplot of distribution of various venue master categories within each cluster and city.

5. The Following plot was obtained.



Cluster Labels = 0 | City = NYC
Cluster Labels = 0 | City = Toronto
Cluster Labels = 1 | City = NYC
Cluster Labels = 1 | City = Toronto
Cluster Labels = 2 | City = NYC
Cluster Labels = 2 | City = Toronto
Cluster Labels = 3 | City = NYC
Cluster Labels = 3 | City = Toronto
Cluster Labels = 4 | City = NYC
Cluster Labels = 4 | City = Toronto
Total_Popular_Venues

On the basis of analysis of above plots, clusters can be categorized in terms of venue types as follows:-

Cluster Number	Category by Venue	Category by Crime
Cluster 0	Arts & Entertainment,Professional & Other Places,Outdoors & Recreation	**(Very Low Crime Rate)**
Cluster 1	Food, Nightlife Spot, Outdoors & Recreation	**(High Crime Rate)**
Cluster 2	Travel & Transport, Outdoors & Recreation, Professional & Other Places	**(Moderate Crime Rate)**
Cluster 3	Arts & Entertainment, Residence, Travel & Transport	**(Very High Crime Rate)**
Cluster 4	Food, Shop & Service	**(Low Crime Rate)**

5. Discussion

5.1 Observations

- Both cities New York and Toronto have a lot of differences as far as crime data and venues data is concerned. While most neighbourhoods in New York have more than 100 popular venues within 500m radius, few neighbourhoods in Toronto have as many popular venues.
- On analysis of the distance matrices (sorted and unsorted) of neighbourhoods in n-dimensional feature space, it was seen that neighbourhoods in New York have high disssimilarity while neighbourhoods in Toronto are highly similar and tightly packed in feature space.
- Analysis of silhuette scores of Toronto dataset revealed reasonable clustering success with silhuette scores ranging from 0.3-0.35. However, silhuette scores for new York dataset were quite low (~ 0.1) implying poor clustering in feature space.
- During the clustering of combined dataset of Toronto and New York for finding similar neighbourhoods between the two cities, it was observed that KMeans didn't perform very satisfactorily. 135 out of 136 neighbourhoods of Toronto were clubbed into single cluster.
- Comparison of various clustering algorithms on combined dataset revealed similar bunching of most of Toronto neighbourhoods into one cluster of New York neighbourhoods. Only SpectralClustering algorithm created reasonable spread of Toronto neighbourhoods into all clusters.
- Run of Inductive Clustering method with use of AgglomerativeClustering to first cluster NYC neighbourhoods and then using KneighboursClassifier to predict cluster labels for Toronto resulted into much better results.
- Analysis of barplots of crime data across cluster and cities revealed different rates of crimes between clusters. Further, in every cluster, crime rates of Toronto neighbourhoods were significantly lower than New York neighbourhoods.
- Analysis of boxplots of Venue Master categories across cluster and cities revealed prominence of 2-3 differenct categories for each cluster. On the basis of this classification, prominent characterstics of each cluster were identified.

5.2 Recommendations

- Cluster 0 is the best cluster as far as crime rates are concerned. The neighbourhoods in this cluster are :-

NYC	Toronto	Crime Pattern	Venue Types
2824 W 17th St, Brooklyn, NY 11224	Stonegate-Queensway	Very Low	Arts & Entertainment,Professional & Other Places,Outdoors & Recreation
Gulf Ave, Staten Island, NY 10314	Keelesdale-Eglinton West	Very Low	Arts & Entertainment,Professional & Other Places,Outdoors & Recreation
391 Fairbanks Ave, Staten Island, NY 10306	Casa Loma	Very Low	Arts & Entertainment,Professional & Other Places,Outdoors & Recreation
170 Sharrotts Ln, Staten Island, NY 10309	High Park-Swansea	Very Low	Arts & Entertainment,Professional & Other Places,Outdoors & Recreation
	Downsview-Roding-CFB	Very Low	Arts & Entertainment,Professional & Other Places,Outdoors & Recreation
	Willowdale West	Very Low	Arts & Entertainment,Professional & Other Places,Outdoors & Recreation
	Don Valley Village	Very Low	Arts & Entertainment,Professional & Other Places,Outdoors & Recreation
	Woodbine Corridor	Very Low	Arts & Entertainment,Professional & Other Places,Outdoors & Recreation

- More features like population, property rents/prices, traffic congestion data, pollution data, etc. may be used to get better comparison and clustering results.

Thank you for going through my notebook.