BATTLE OF THE NEIGHBORHOODS: A COMPARISON OF TORONTO, NEW YORK, AND GREATER LONDON

Celine Canes

20th July 2021

1. Introduction

1.1. Background

This research aims to analyze and compare the neighborhoods of Toronto with those in New York and London. We believe that an exploratory analysis of these mammoth concrete jungles could provide valuable insights given their similarities and differences.

Toronto, New Work, and London are all international centers of business, arts, culture, and recognized as some of the most multicultural and cosmopolitan cities in the world. Toronto is the capital city of the Canadian province of Ontario, the most populous city in Canada, and the fourth most populous city in North America. London is one of the world's most important global cities, with considerable influence upon the arts, commerce, education, entertainment, and many other aspects of the world. Last but not least, New York City has been described as the cultural, financial, and media capital of the world – not to mention the most photographed city in the world[1].

Nonetheless, each city has their key differences from the others. To illustrate, New York has an estimated population density of 10,716.36/km^2[2] while Toronto's is estimated to be nearly half of that at 4,334.4/km^2[3], despite the fact that New York's metropolitan area is much larger than Toronto's. There are also further considerations for anyone thinking of living in one of these cities, such as London's generally cool winters with low temperature variations versus New York's chilly and damp winters. Perhaps most importantly, an aspiring traveler or mover would want to know the characteristics of the neighborhoods in these cities, as well as the

[1] Travelzoo. (2018). 10 Most Photographed Places in the World Will Surprise You. https://www.travelzoo.com/ca/blog/10-most-photographed-places-in-the-world-will-surprise-you-2/
[2] US Census Bureau. (2021). City and Town Population Totals: 2010-2020. https://www.census.gov/programs-surveys/popest/technical-documentation/research/evaluation-estimates/2020-evaluation-estimates/
[3] Statistics Canada. (2016). Census Profile, Canada 2016. https://www12.statcan.gc.ca/census-recensement/2016/

venues that could be found in each neighborhood. This is where our research will come in handy, as explained in the next part of this report.

1.2. Business Problem

This is an exploratory analysis aiming to determine whether the city of Toronto is more similar to New York or London. The aim is to help tourists choose their destinations depending on the venues and destinations that the neighborhoods have to offer. The findings from this research could also assist Torontonian thinking of migrating to New York or London. Overall, our findings will help stakeholders to make informed decisions regarding their travels and sate any curiosity they might have, including the different kinds of cuisines and entertainment destinations that a city might have to offer.

2. Data

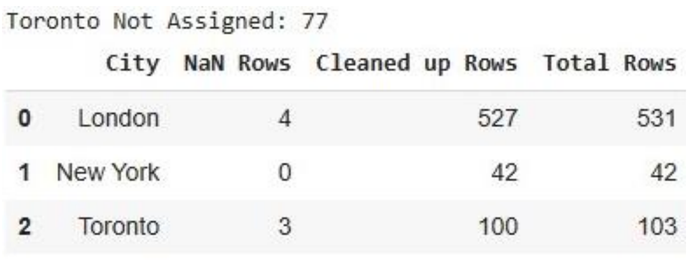
For the purpose of this research, we obtain geographical data from the cities of Toronto, New York, and London. This includes information regarding the neighborhoods of each city and their coordinates (longitude and latitude) which were scraped and transformed into *pandas* data frames from the following websites.

1. **Toronto**: The Wikipedia page for the list of postal codes in Toronto and this csv file from Coursera for the geospatial coordinates.
2. **New York**: This Geonames website for the list of New York postal codes and coordinates, which were linked to their respective neighborhoods through this NYV government pdf file.
3. **London**: This Wikipedia page for the list of areas of London wherein the coordinates were easily obtained from Britain's OS Grid.

This resulted in a dataset of 103 neighborhoods in Toronto, 42 neighborhoods in New York, and 531 neighborhoods in London. Toronto had 77 non-assigned postal codes which we opted to exclude from the dataset.

The coordinates are used to determine the venues within a 500-meters radius of the neighborhoods. Herein, the data on the venue's names and venue categories around each coordinate is gathered from the Foursquare API[4], and are limited to 100 venues for each neighborhood. After filtering out the neighborhoods with no venues within a 500-meters radius which resulted in *NaN* values, there were 100 neighborhoods in Toronto, 42 neighborhoods in New York, and 527 neighborhoods in London.

Table 1. Number of Observations Before and After Data Cleanup



```
Toronto Not Assigned: 77
```

	City	NaN Rows	Cleaned up Rows	Total Rows
0	London	4	527	531
1	New York	0	42	42
2	Toronto	3	100	103

3. Methodology

First and foremost, we entered the following codes to import the packages needed for the data processing, such as *pandas*, *Beautiful Soup*, and *matplotlib*, among others.

```python
import requests
import bs4 as bs
import pandas as pd
import numpy as np
from geopy.geocoders import Nominatim
import matplotlib.cm as cm
import matplotlib.colors as colors
from sklearn.cluster import KMeans
import folium
import matplotlib.pyplot as plt
```

The following codes were run in order to gather the data on the neighborhoods and boroughs and their coordinates from the data sources cited in Chapter 2 of this report. Following that, we constructed a *pandas* data frame table for each city, with the first being London.

[4] Using the following URL and explore query:
https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={},{}&radius={}&limit={}

```python
def get_london_data():
  res  = requests.get('https://en.wikipedia.org/wiki/List_of_areas_of_London')
  soup = bs.BeautifulSoup(res.content,'lxml')

  wiki_table = soup.findAll('table')[1]
  data = []

  #constructing the table
  table_df = pd.DataFrame()
  # skip header
  for row in wiki_table.findAll('tr')[1:]:
    row = row.findAll('td')

    nhood   = row[0].text
    borough = row[1].text
    coords  = row[-1].span.a['href'].split('params=')[-1].split('_')
    lat     = str(float(coords[0]) * (-1 if coords[1]=='S' else 1))
    long    = str(float(coords[2]) * (-1 if coords[3]=='W' else 1))

    data.append({
      'Neighbourhood': nhood,
      'Borough': borough,
      'Coordinates': [(lat, long)],
      'Latitude': float(lat),
      'Longitude': float(long),
    })

  return data

london = get_london_data()
```

And then for New York, which was done by querying the postal codes and neighborhoods from the pdf file and matching them with the Geonames data:

```python
def get_ny_coordinates(zipcode):
  url   = 'http://www.geonames.org/postalcode-search.html?q='+zipcode+'&country=
  res   = requests.get(url)
  soup  = bs.BeautifulSoup(res.content, 'lxml')
  table = soup.findAll('table', {'class': 'restable'})[0]

  coords = []
  # skip header and skip every other row (get only coords)
  for row in table.findAll('tr')[2::2]:
    row = row.find('small').text.split('/')
    coords.append((row[0], row[1]))

  return coords


def get_ny_data():
  ny = [
    ['Bronx', 'Kingsbridge - Riverdale',
      ['10463', '10471']],
    ['Bronx', 'Northeast Bronx',
      ['10466', '10469', '10470', '10475']],
    ['Bronx', 'Fordham - Bronx Park',
      ['10458', '10467', '10468']],
    ['Bronx', 'Pelham - Throgs Neck',
      ['10461', '10462', '10464', '10465', '10472', '10473']],
    ['Bronx', 'Crotona - Tremont',
      ['10453', '10457', '10460']],
  data = []

  for nhood in ny:
    borough  = nhood[0]
    zipcodes = nhood[2]
    nhood    = nhood[1]

    coords = []
    for z in zipcodes:
      c = get_ny_coordinates(z)
      coords += c

    data.append({
        'Neighbourhood': nhood,
        'Borough': borough,
        'Coordinates': coords,
        'Latitude': float(coords_avg(coords)[0]),
        'Longitude': float(coords_avg(coords)[1]),
    })

  return data

new_york = get_ny_data()
```

And lastly, for Toronto using the Wikipedia article and csv file:

```
def get_toronto_data():
  #obtaining data from wikipedia.org
  url   = "https://en.wikipedia.org/wiki/List_of_postal_codes_of_Canada:_M"
  res   = requests.get(url)
  soup  = bs.BeautifulSoup(res.content,'lxml')
  table = soup.find('table')
  df = pd.DataFrame()

  #constructing the table
  df = pd.DataFrame()
  for row in table.findAll('td'):
    cell = {}
    if row.span.text=='Not assigned':
      pass
    else:
      cell['PostalCode'] = row.p.text[:3]
      cell['Borough'] = (row.span.text).split('(')[0]
      cell['Neighbourhood'] = row.span.text.split('(')[1].strip(')').replace(' /',',').replace(')',' ').strip(' ')
      df = df.append(cell, ignore_index=True)

  #downloading geospatial data
  geospatial_url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DS0
  geospatial_data = pd.read_csv(geospatial_url)
  geospatial_data.columns = ['PostalCode', 'Latitude', 'Longitude']

  merged_data = pd.merge(df, geospatial_data, on='PostalCode')

  data = []
  for _, row in merged_data.iterrows():
    data.append({
        'Neighbourhood': row['Neighbourhood'],
        'Borough': row['Borough'],
        'Coordinates': [(row['Latitude'], row['Longitude'])],
        'Latitude': row['Latitude'],
        'Longitude': row['Longitude'],
    })

  return data

toronto = get_toronto_data()
```

Furthermore, we utilized the explore query on the Foursquare API to obtain the list of venues and venue categories within a 500-meter radius of each neighborhood, with a 100-venue limit.

```
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)
url
```

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

```
try:
  # parse response
  for venue in res['response']['groups'][0]['items']:
    venue  = venue['venue']
    id     = venue['id']
    name   = venue['name']
    coords = venue['location']

    latitude  = coords['lat']
    longitude = coords['lng']

    category = []
    if 'categories' in venue:
      category = venue['categories']
    elif 'venue.categories' in venue:
      category = venue['venue.categories']
    if len(category) == 0:
      category = None
    else:
      category = category[0]['name']

    venues[id] = {
        'Name': name,
        'Category': category,
        'Latitude': latitude,
        'Longitude': longitude,
    }
except Exception as e:
  print(e)
  print(res)

return list(venues.values())
```

Then the following codes were run to construct the *pandas* data frames containing the neighborhoods and their coordinates, venues, venues' coordinates, and venues' categories. The codes used for London are as given below.

```python
london_venues = []
# TODO: get all
for nhood in london:
  venues = get_venues(nhood['Coordinates'])
  for venue in venues:
    london_venues.append([
      nhood['Neighbourhood'],
      coords_avg(nhood['Coordinates'])[0],
      coords_avg(nhood['Coordinates'])[1],
      venue['Name'],
      venue['Latitude'],
      venue['Longitude'],
      venue['Category'],
    ])

london_venues = pd.DataFrame(london_venues)
london_venues.columns = [
  'Neighbourhood',
  'Neighbourhood Latitude',
  'Neighbourhood Longitude',
  'Venue',
  'Venue Latitude',
  'Venue Longitude',
  'Venue Category']

london_venues.head()
```

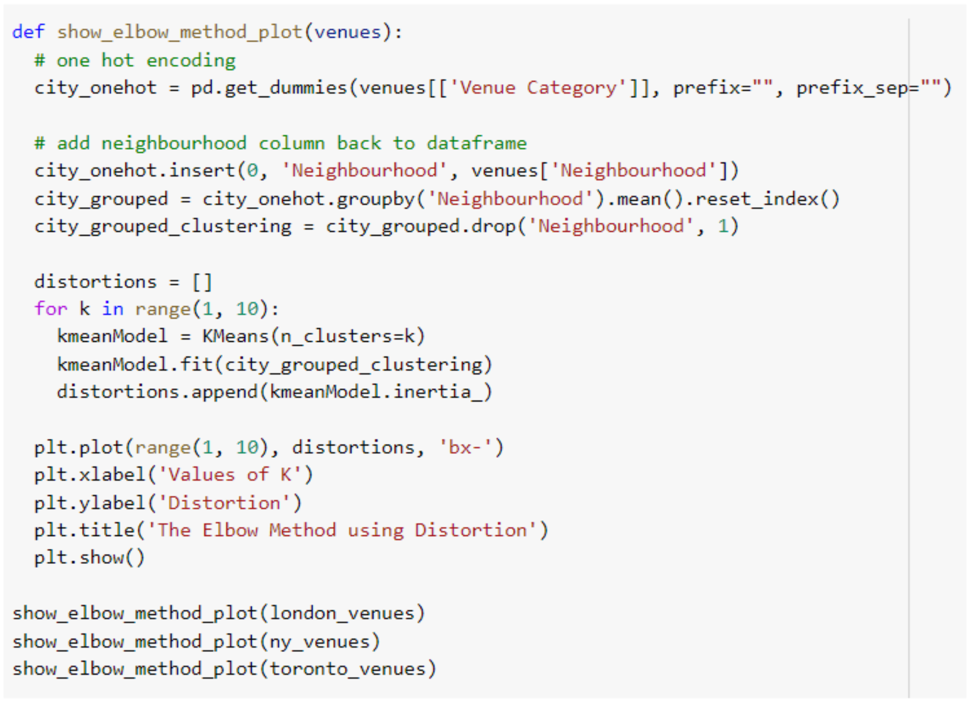
Repeated twice more for Toronto and New York, these codes resulted in the following outputs for (from top to bottom) London, New York, and Toronto.

	Neighbourhood	Neighbourhood Latitude	Neighbourhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Abbey Wood	51.486480	0.108592	Co-op Food	51.487490	0.113751	Grocery Store
1	Abbey Wood	51.486480	0.108592	Morley's	51.485610	0.102389	Fried Chicken Joint
2	Abbey Wood	51.486480	0.108592	East Ocean	51.485279	0.102426	Chinese Restaurant
3	Abbey Wood	51.486480	0.108592	Meghna Tandoori	51.485709	0.101681	Indian Restaurant
4	Acton	51.510588	-0.264989	The Station House	51.508877	-0.263076	Pub

	Neighbourhood	Neighbourhood Latitude	Neighbourhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Kingsbridge - Riverdale	40.8905	-73.906	Sam's Pizza	40.879435	-73.905859	Pizza Place
1	Kingsbridge - Riverdale	40.8905	-73.906	Tibbett Diner	40.880404	-73.908937	Diner
2	Kingsbridge - Riverdale	40.8905	-73.906	Estrellita Poblana V	40.879687	-73.906257	Mexican Restaurant
3	Kingsbridge - Riverdale	40.8905	-73.906	El Malecon	40.879338	-73.904457	Caribbean Restaurant
4	Kingsbridge - Riverdale	40.8905	-73.906	Garden Gourmet Market	40.881350	-73.903389	Gourmet Shop

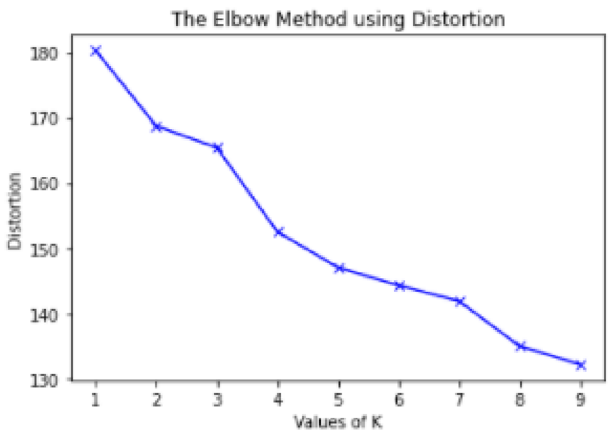
	Neighbourhood	Neighbourhood Latitude	Neighbourhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Parkwoods	43.753259	-79.329656	Brookbanks Park	43.751976	-79.332140	Park
1	Parkwoods	43.753259	-79.329656	KFC	43.754387	-79.333021	Fast Food Restaurant
2	Parkwoods	43.753259	-79.329656	Variety Store	43.751974	-79.333114	Food & Drink Shop
3	Victoria Village	43.725882	-79.315572	Victoria Village Arena	43.723481	-79.315635	Hockey Arena
4	Victoria Village	43.725882	-79.315572	Tim Hortons	43.725517	-79.313103	Coffee Shop

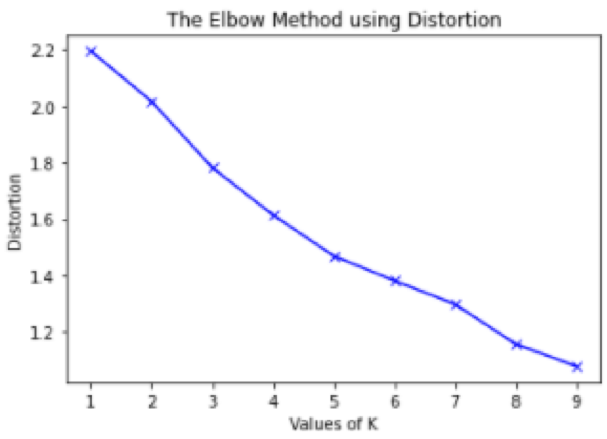
Proceeding to the clustering of the neighborhoods, we must first determine the optimal number of clusters or *k* value for each city. To do this, we utilize the *Elbow Method* using the following code:

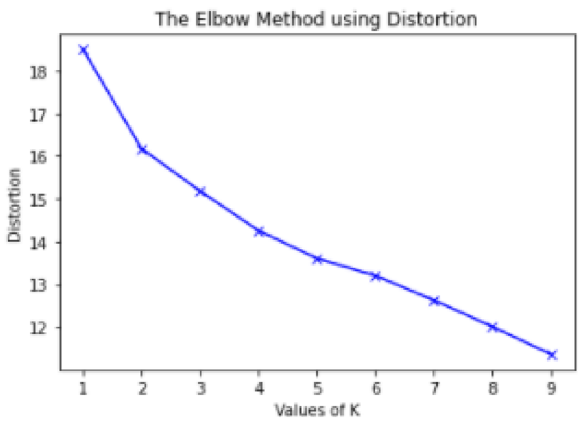


```
def show_elbow_method_plot(venues):
  # one hot encoding
  city_onehot = pd.get_dummies(venues[['Venue Category']], prefix="", prefix_sep="")

  # add neighbourhood column back to dataframe
  city_onehot.insert(0, 'Neighbourhood', venues['Neighbourhood'])
  city_grouped = city_onehot.groupby('Neighbourhood').mean().reset_index()
  city_grouped_clustering = city_grouped.drop('Neighbourhood', 1)

  distortions = []
  for k in range(1, 10):
    kmeanModel = KMeans(n_clusters=k)
    kmeanModel.fit(city_grouped_clustering)
    distortions.append(kmeanModel.inertia_)

  plt.plot(range(1, 10), distortions, 'bx-')
  plt.xlabel('Values of K')
  plt.ylabel('Distortion')
  plt.title('The Elbow Method using Distortion')
  plt.show()

show_elbow_method_plot(london_venues)
show_elbow_method_plot(ny_venues)
show_elbow_method_plot(toronto_venues)
```

This resulted in the following graphs for the *Elbow Method* using distortion for (from top to bottom) London, New York, and Toronto. Herein, we detemined that the optimal *k* values for London is 5, New York is 4, and Toronto is 5.



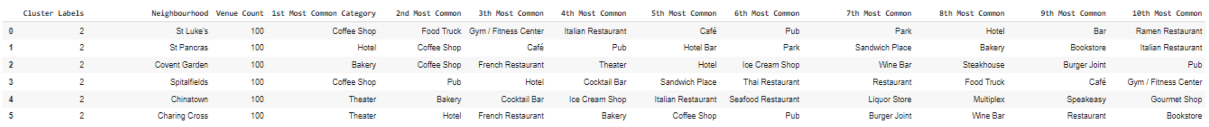




With the following codes, we utilize *one hot encoding* and create a new data frame in which we can show the number of venues each neighborhood has (although with the maximum of 100 venues as determined by the limit), the most common venue categories for each neighborhood, and the cluster each neighborhood falls into.

```
def get_most_common(city, venues, kclusters=3):
  # one hot encoding
  city_onehot = pd.get_dummies(venues[['Venue Category']], prefix="", prefix_sep="")

  # add neighbourhood column back to dataframe
  city_onehot.insert(0, 'Neighbourhood', venues['Neighbourhood'])

  city_grouped = city_onehot.groupby('Neighbourhood').mean().reset_index()

  num_top_venues = min(10, city_grouped.shape[1]-1)

  indicators = ['st', 'nd', 'rd']
  columns = ['Neighbourhood']
  for i in range(1, num_top_venues+1):
      columns.append('{}{} Most Common'.format(i, indicators[i-1] if i<3 else 'th') + (' Category' if i==1 else ''))

  # create a new dataframe
  neighbourhoods_venues_sorted = pd.DataFrame(columns=columns)
  neighbourhoods_venues_sorted['Neighbourhood'] = city_grouped['Neighbourhood']

  for i in np.arange(city_grouped.shape[0]):
    row_categories = city_grouped.iloc[i, :].iloc[1:]
    row_categories_sorted = row_categories.sort_values(ascending=False)
    neighbourhoods_venues_sorted.iloc[i, 1:] = row_categories_sorted.index.values[0:num_top_venues]

  city_grouped_clustering = city_grouped.drop('Neighbourhood', 1)

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

  # add clustering labels
  neighbourhoods_venues_sorted.insert(0, 'Cluster Labels', kmeans.labels_)

  city_merged = pd.DataFrame(data=city)


  # merge manhattan_grouped with manhattan_data to add latitude/longitude for each neighbourhood
  city_merged = city_merged.join(neighbourhoods_venues_sorted.set_index('Neighbourhood'), on='Neighbourhood')
  city_merged = city_merged.drop('Coordinates', 1)

  # count how many venues a neighbourhood has
  venue_count = {}
  for _, venue in venues.iterrows():
    nbhood = venue['Neighbourhood']
    if nbhood not in venue_count:
      venue_count[nbhood] = 0
    venue_count[nbhood] += 1

  # add venue count and sort by it
  neighbourhoods_venues_sorted.insert(2, 'Venue Count', [venue_count.get(n, 0) for n in neighbourhoods_venues_sorted['Neighbourhood']])
  neighbourhoods_venues_sorted = neighbourhoods_venues_sorted.sort_values('Venue Count', ascending=False, ignore_index=True)

  return city_merged, neighbourhoods_venues_sorted
```

This resulted in the following output which now starts with the neighborhood with the highest number of venue and continues in descending order, and informs on the most common venue categories for each neighborhood. The neighborhoods are now also clustered based on the optimal *k* value for each city that we found through the *Elbow Method* using distortion.



	Cluster Labels	Neighbourhood	Venue Count	1st Most Common Category	2nd Most Common	3th Most Common	4th Most Common	5th Most Common	6th Most Common	7th Most Common	8th Most Common	9th Most Common	10th Most Common
0	2	St Luke's	100	Coffee Shop	Food Truck	Gym / Fitness Center	Italian Restaurant	Café	Pub	Park	Hotel	Bar	Ramen Restaurant
1	2	St Pancras	100	Hotel	Coffee Shop	Café	Pub	Hotel Bar	Park	Sandwich Place	Bakery	Bookstore	Italian Restaurant
2	2	Covent Garden	100	Bakery	Coffee Shop	French Restaurant	Theater	Hotel	Ice Cream Shop	Wine Bar	Steakhouse	Burger Joint	Pub
3	2	Spitalfields	100	Coffee Shop	Pub	Hotel	Cocktail Bar	Sandwich Place	Thai Restaurant	Restaurant	Food Truck	Café	Gym / Fitness Center
4	2	Chinatown	100	Theater	Bakery	Cocktail Bar	Ice Cream Shop	Italian Restaurant	Seafood Restaurant	Liquor Store	Multiplex	Speakeasy	Gourmet Shop
5	2	Charing Cross	100	Theater	Hotel	French Restaurant	Bakery	Coffee Shop	Pub	Burger Joint	Wine Bar	Restaurant	Bookstore

Lastly, we visualize our findings through a *folium* map using the neighborhood names, coordinates, and cluster labels which will be represented by the different colors of the points on the map.

```python
def create_nbhood_map(merged, lat, long, kclusters=3):
  map_clusters = folium.Map(location=[lat, long], zoom_start=11)

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

  merged = merged
  # add markers to the map
  markers_colors = []
  for lat, lon, poi, cluster in zip(merged['Latitude'], merged['Longitude'], merged['Neighbourhood'], merged['Cluster Labels'].astype(int)):
    label = folium.Popup(str(poi) + ' Cluster ' + str(cluster), parse_html=True)
    folium.CircleMarker(
        [lat, lon],
        radius=5,
        popup=label,
        color=rainbow[cluster-1],
        fill=True,
        fill_color=rainbow[cluster-1],
        fill_opacity=0.7).add_to(map_clusters)

  return map_clusters
```

```python
create_nbhood_map(london_merged, 51.509865, -0.118092, kclusters=5)
```

```python
create_nbhood_map(ny_merged, 40.730610, -73.935242, kclusters=4)
```

```python
create_nbhood_map(toronto_merged, 43.651070, -79.347015, kclusters=5)
```

4. Results and Discussions

Within this section of the report, we will discuss our results and their implications, as well as how they address the business problems. First, we will start by discussing the clustered neighborhoods of Toronto, London, and New York, which are visualized in the folium maps. Herein, Figure 1 represents the city of Toronto, Figure 2 represents Greater London, and Figure 3 represents the city of New York.

Based on the number of neighborhoods, the distribution of the neighborhoods, and how that distribution affects the clustering method, Toronto is more similar to New York than it is to London. To illustrate, London has over 500 neighborhoods with several that do not have venues at all within a 500-meter radius and are thus excluded from the dataset. However, there are also similarities in how the neighborhoods of Downtown Toronto in Figure 1 collect densely together and are all categorized into a single cluster, which is also the case for Cluster 3 (blue-colored) of London, which are densely packed near the City of London at the center of the map. In comparison, the neighborhoods of New York are more evenly spread, with the exception of perhaps South Beach and Rockaway.

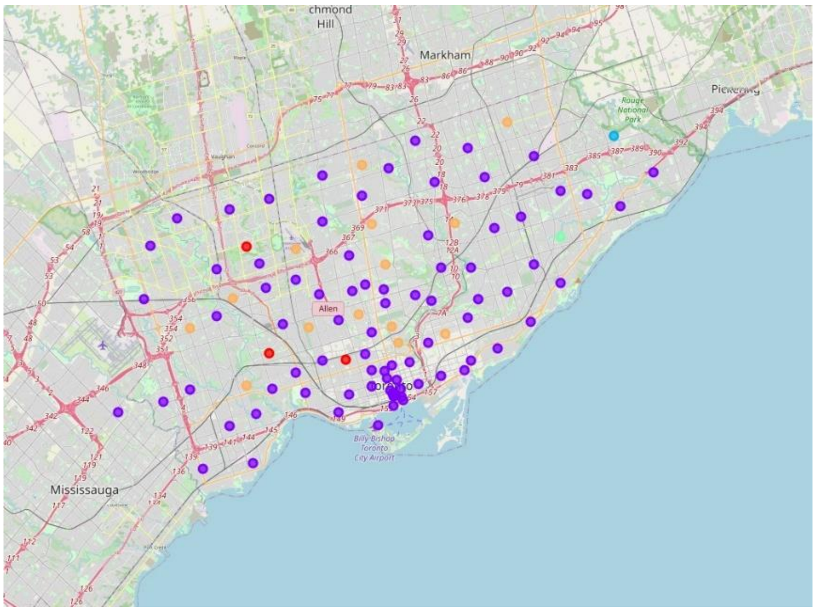


Figure 1. Clustered Neighborhoods in Toronto

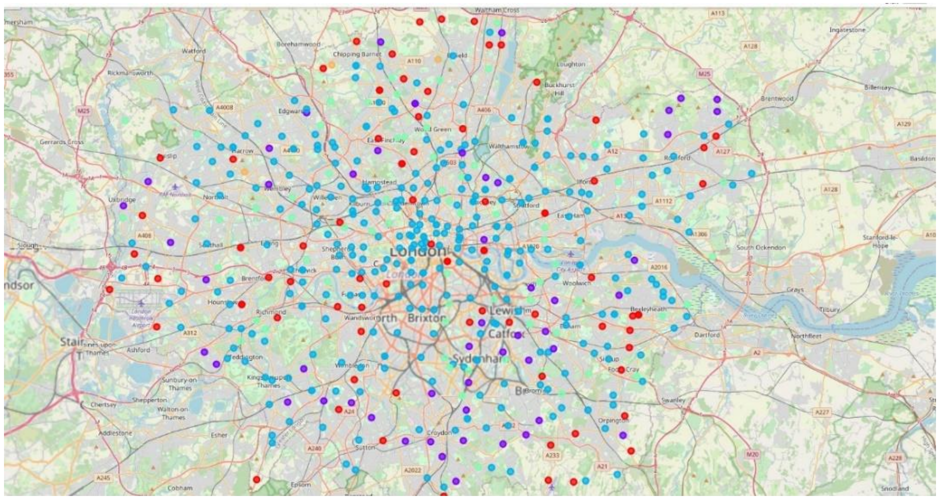


Figure 2. Clustered Neighborhoods in London

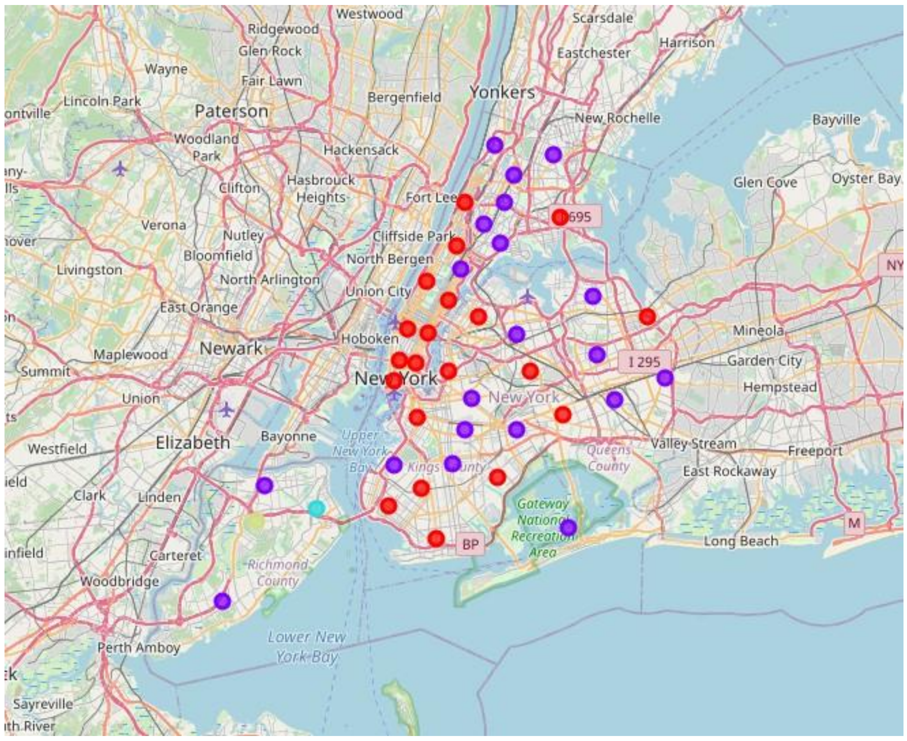


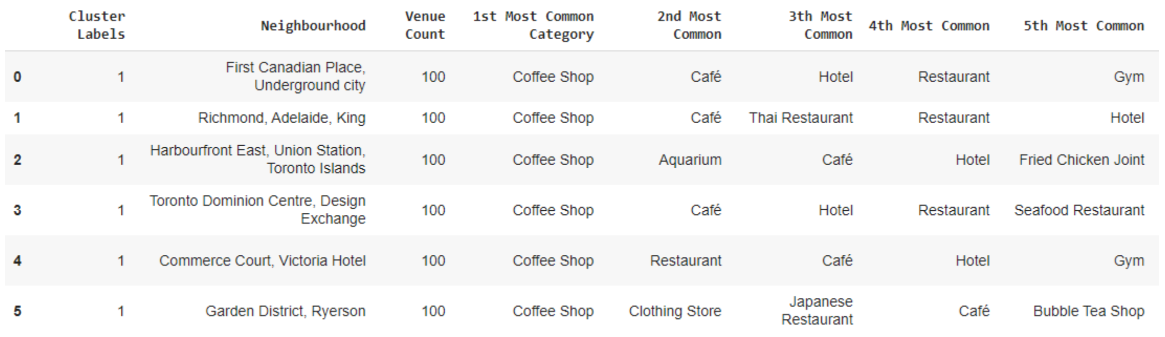
Figure 3. Clustered Neighborhoods in New York

This model could be used to examine the characteristics of each cluster so that aspiring movers or travellers can find the cluster with neighborhoods most similar to theirs or otherwise have venues that they are most interested in. To illustrate, we give an example of some of the characteristics of the clusters from the different cities below.

1. **Cluster 2 Toronto (Purple):** Abundance of cafes and coffee shops.
2. **Cluster 1 NYC (Red):** Abundance of coffee shops and Italian restaurants.
3. **Cluster 3 London (Blue):** Located near cafes, pubs, and hotels suitable for tourists.
4. **Cluster 4 NYC (Blue):** Located near tennis courts, athletic centers, and yoga studios.

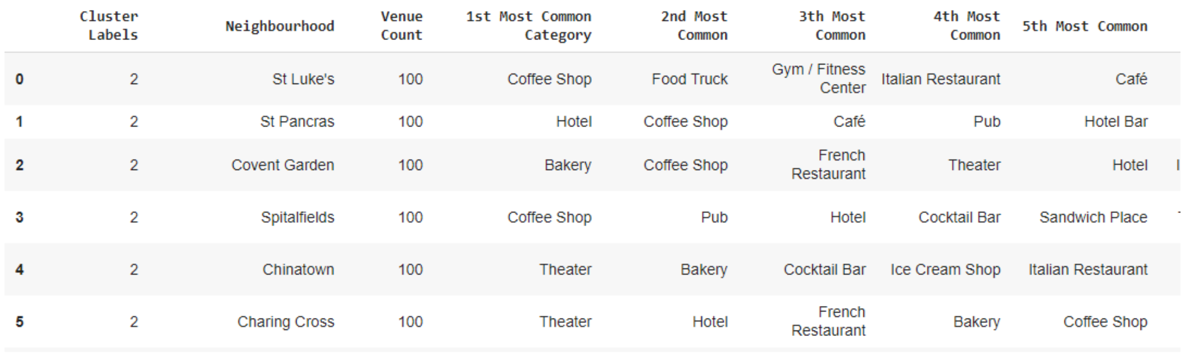
Personal considerations and motivations in travelling could be used by each person in regards to this model to determine which cluster's characteristics is most appealing to them. Following that, they could narrow down to determining which neighborhood has venues that most interest them in. Herein, we present 5 noteworthy neighborhoods from each city and their 5 most common venue categories in Tables 2, 3, and 4.

Table 2. Noteworthy Toronto Neighborhoods and Their 5 Most Common Venue Categories



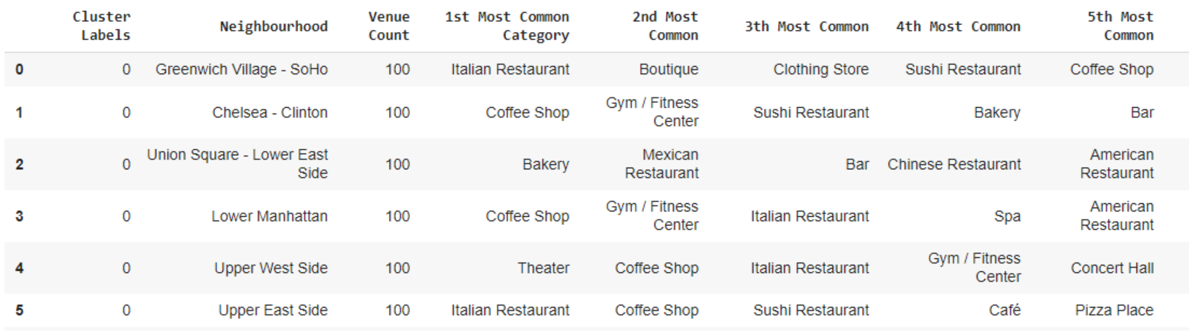
	Cluster Labels	Neighbourhood	Venue Count	1st Most Common Category	2nd Most Common	3th Most Common	4th Most Common	5th Most Common
0	1	First Canadian Place, Underground city	100	Coffee Shop	Café	Hotel	Restaurant	Gym
1	1	Richmond, Adelaide, King	100	Coffee Shop	Café	Thai Restaurant	Restaurant	Hotel
2	1	Harbourfront East, Union Station, Toronto Islands	100	Coffee Shop	Aquarium	Café	Hotel	Fried Chicken Joint
3	1	Toronto Dominion Centre, Design Exchange	100	Coffee Shop	Café	Hotel	Restaurant	Seafood Restaurant
4	1	Commerce Court, Victoria Hotel	100	Coffee Shop	Restaurant	Café	Hotel	Gym
5	1	Garden District, Ryerson	100	Coffee Shop	Clothing Store	Japanese Restaurant	Café	Bubble Tea Shop

Table 3. Noteworthy London Neighborhoods and Their 5 Most Common Venue Categories



	Cluster Labels	Neighbourhood	Venue Count	1st Most Common Category	2nd Most Common	3th Most Common	4th Most Common	5th Most Common
0	2	St Luke's	100	Coffee Shop	Food Truck	Gym / Fitness Center	Italian Restaurant	Café
1	2	St Pancras	100	Hotel	Coffee Shop	Café	Pub	Hotel Bar
2	2	Covent Garden	100	Bakery	Coffee Shop	French Restaurant	Theater	Hotel
3	2	Spitalfields	100	Coffee Shop	Pub	Hotel	Cocktail Bar	Sandwich Place
4	2	Chinatown	100	Theater	Bakery	Cocktail Bar	Ice Cream Shop	Italian Restaurant
5	2	Charing Cross	100	Theater	Hotel	French Restaurant	Bakery	Coffee Shop

Table 4. Noteworthy NY Neighborhoods and Their 5 Most Common Venue Categories



	Cluster Labels	Neighbourhood	Venue Count	1st Most Common Category	2nd Most Common	3th Most Common	4th Most Common	5th Most Common
0	0	Greenwich Village - SoHo	100	Italian Restaurant	Boutique	Clothing Store	Sushi Restaurant	Coffee Shop
1	0	Chelsea - Clinton	100	Coffee Shop	Gym / Fitness Center	Sushi Restaurant	Bakery	Bar
2	0	Union Square - Lower East Side	100	Bakery	Mexican Restaurant	Bar	Chinese Restaurant	American Restaurant
3	0	Lower Manhattan	100	Coffee Shop	Gym / Fitness Center	Italian Restaurant	Spa	American Restaurant
4	0	Upper West Side	100	Theater	Coffee Shop	Italian Restaurant	Gym / Fitness Center	Concert Hall
5	0	Upper East Side	100	Italian Restaurant	Coffee Shop	Sushi Restaurant	Café	Pizza Place

This research hopes that the constructed model could be used for the personal consideration of individuals in their travel decisions. For example, based on the tables given above, if Jane were interested in theater while in London, she could go to Chinatown or Charing Cross. Moreover, if she were interested in going shopping while in New York, then she could consider Greenwich Village in Soho. Furthermore, if Jane were staying for longer and wanted to live somewhere nearby workout facilities, she could look for neighborhoods where the most common venue categories are “Gym/ Fitness Center” or maybe even “Yoga Studio”.

5. Conclusion

And thus, we conclude the entirety of our research, wherein there are a few key highlights that we wish to reiterate. The number of neighborhoods and the distribution of those neighborhoods, as well as how the distribution affects the clustering method, is more similar between the cities of Toronto and New York. This might mean that Torontonians might adapt more easily when migrating to the city of New York than to London. Furthermore, using this model, aspiring travelers or movers might consider the characteristics of each cluster in the different cities to determine one that most suit their interests. After determining the cluster of interest, they could determine which neighborhood they have the most inclination towards based on the most common venue categories for that neighborhood.