Report

Toronto Airbnb and Nearby Venues

Table of Contents

1. Introduction

Airbnb is an online marketplace where millions of hosts and travellers can create a free account so they can list their space or book accommodations anywhere in the world. It covers more than 100,000 cities and 220 countries worldwide. Airbnb has been successful because it has managed to replace hotels as a traditional method of accommodation for travellers worldwide.
Being one of the most popular tourist destination in Canada, Toronto’s AirBnb market has been growing with an increased number of hosts joining and listing their homes for rent.

2. Business Problem

For this Capstone Project the business aim is to provide the travellers with relevant information about Airbnb offers and variety of restaurants to choose from in Toronto.
The listing price and amenities are an important factor in choosing a rental, but location is also important. For travellers who love food it might be helpful to know in advance where to find a place to rent that has also has nearby restaurants of interest.
Using FourSquare API venues data to compare different neighbourhoods could be a good opportunity to choose, for example, an area with italian restaurants, aisian cusine, irish pub or a dessert shop.
Our main focus will be finding the top five food venue categories around **Airbnb** listings from the **Downtown Toronto** neighbourhoods.

3. Data sources and description

In this project I used a few datasets as data sources:

- List of Postal code of Canada: M provided by the Wikipedia page to get information about Toronto's postal codes, boroughs and neighbourhoods.
- Toronto's geographical coordinates available for download as a csv file from the Week3 course link
- Toronto's Airbnb listings provided by the website: Inside Airbnb
- Foursquare API to retreive location and opther information about various venues in Toronto. Using the Foursquare's explore API (which gives venues recommendations), I'm fetching details about the venues up present in Toronto and collected their names, categories and locations (latitude and longitude). From Foursquare API, I retrieved the following for each venue:
    - Name: The name of the venue.
    - Category: The category type as defined by the API.
    - Latitude: The latitude value of the venue.
    - Longitude: The longitude value of the venue.

4. Methodology

I created my project using Jupyter Notebook with python kernel, so I started by importing all the necessary libraries:
code snippet:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import os
#import geopandas as gpd
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
import json # library for handling JSON files
#!conda install -c conda-forge geopy --yes # uncomment this line if you haven't installed geopy y
import geocoder
from geopy.geocoders import Nominatim # library that convserts an address into longitude and lati
import requests # library to handle requests
from pandas import json_normalize # library to tranform JSON file into a pandas dataframe

import math
import seaborn as sns

# Matplotlib and associated plotting modules
import matplotlib.cm as cm
import matplotlib.colors as colors
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

# import k-means from clustering stage
from sklearn.cluster import KMeans

#!conda install -c conda-forge folium=0.5.0 --yes # uncomment this line if you haven't installed i

# Folium and associated plugins for map rendering
import folium # map rendering library
from folium import Circle, Marker
from folium.plugins import HeatMap, MarkerCluster

print('Libraries imported.')
```

4.1 Data Collection

In the data collection stage I began with collecting the required data for the city of Toront's postal codes, geographical coordinates, and Toronto's Airbnb listings.
To collect data for postal codes, I scraped the Wikipedia page in order to obtain the data that is in the table of postal codes and to transform it into a pandas dataframe. I took data from the first table.
Next, I read the geospatial data csv file that has the geographical coordinates of each postal code, and join both data frames on `Postal Code` column.
Finally, I downloaded and read into a pandas data frame, the Toronto's Airbnb listings from February 2021, which is available from Inside Airbnb website.

a. Scraping Toronto Neighborhoods Table from Wikipedia

Data-frame will consist of three columns: Postal Code, Borough, and Neighbourhood

```
url='https://en.wikipedia.org/wiki/List_of_postal_codes_of_Canada:_M'
dfs=pd.read_html(url, header=0)
```

```
df_postal=dfs[0] # get the first table
df_postal.head()
```

	Postal Code	Borough	Neighbourhood
0	M1A	Not assigned	Not assigned
1	M2A	Not assigned	Not assigned
2	M3A	North York	Parkwoods
3	M4A	North York	Victoria Village
4	M5A	Downtown Toronto	Regent Park, Harbourfront

b. Add the geographical coordinates to the neighbourhoods table

Read the Geospatial_data csv file that has the geographical coordinates of each postal code, then join it with the neighbourhoods data frame by the postal code column

Postal Code	Latitude	Longitude
M1B	43.806686	-79.194353
M1C	43.784535	-79.160497
M1E	43.763573	-79.188711
M1G	43.770992	-79.216917
M1H	43.773136	-79.239476

c. Read the Toronto's Airbnb listings

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_reviews	las
0	1419	Beautiful home in amazing area!	1565	Alexandra	NaN	Little Portugal	43.64617	-79.42451	Entire home/apt	469	28	7	2(
1	8077	Downtown Harbourfront Private Room	22795	Kathie & Larry	NaN	Waterfront Communities-The Island	43.64105	-79.37628	Private room	96	180	169	2(
2	23691	Queen Bedroom close to downtown	93825	Yohan & Sarah	NaN	Briar Hill-Belgravia	43.69602	-79.45468	Private room	72	28	217	2(
3	27423	Executive Studio Unit-Ideal for One Person	118124	Brent	NaN	Greenwood-Coxwell	43.66890	-79.32592	Entire home/apt	45	365	26	2(
4	30931	Downtown Toronto - Waterview Condo	22795	Kathie & Larry	NaN	Waterfront Communities-The Island	43.64151	-79.37643	Entire home/apt	128	180	1	2(

Read the csv file into a dataframe, and take a look at the headers and a few rows to better understand the data structure

4.2 Data preprocessing

Toronto's neighbourhoods data frame will consist of three columns: PostalCode, Borough, and Neighbourhood.
Only the cells that have an assigned borough will be processed. Borough that is not assigned are ignored. More than one neighborhood can exist in one postal code area. For example, in the table on the Wikipedia page, you will notice that M5A is listed twice and has two neighbourhoods: Harbourfront and Regent Park. These two rows will be combined into one row with the neighbourhood separated with a comma. If a cell has a borough but a Not assigned neighbourhood, then the neighbourhood will be the same as the borough.

	PostalCode	Borough	Neighbourhood
0	M1B	Scarborough	Malvern, Rouge
1	M1C	Scarborough	Rouge Hill, Port Union, Highland Creek
2	M1E	Scarborough	Guildwood, Morningside, West Hill
3	M1G	Scarborough	Woburn
4	M1H	Scarborough	Cedarbrae

Join both, the Toronto's neighbourhoods postal codes data frame and geospatial data frame, on `Postal Code` column

	PostalCode	Borough	Neighbourhood	Latitude	Longitude
0	M1B	Scarborough	Malvern, Rouge	43.806686	-79.194353
1	M1C	Scarborough	Rouge Hill, Port Union, Highland Creek	43.784535	-79.160497
2	M1E	Scarborough	Guildwood, Morningside, West Hill	43.763573	-79.188711
3	M1G	Scarborough	Woburn	43.770992	-79.216917
4	M1H	Scarborough	Cedarbrae	43.773136	-79.239476

Because the interest is only in *Downtown Toronto*, so I selected only the neighbourhoods from this borough.
Create a new dataframe that has the 'Borough' column value equal with 'Downtown Toronto'.

```
downtown_data = toronto_data[toronto_data['Borough']=='Downtown Toronto'].r
eset_index(drop=True)
downtown_data
```

There are **19 neighbourhoods** in Downtown Toronto area. From the Airbnb listings I selected only the following columns:
'neighbourhood', 'latitude', 'longitude', 'room_type', 'price'

	neighbourhood	latitude	longitude	room_type	price
0	Little Portugal	43.64617	-79.42451	Entire home/apt	469
1	Waterfront Communities-The Island	43.64105	-79.37628	Private room	96
2	Briar Hill-Belgravia	43.69602	-79.45468	Private room	72
3	Greenwood-Coxwell	43.66890	-79.32592	Entire home/apt	45
4	Waterfront Communities-The Island	43.64151	-79.37643	Entire home/apt	128

The data frame has **15832 rows and 5 columns**, also from the `info()` method it is observed that there are no missing values.

```
(15832, 5)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15832 entries, 0 to 15831
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   neighbourhood  15832 non-null  object 
 1   latitude       15832 non-null  float64
 2   longitude      15832 non-null  float64
 3   room_type      15832 non-null  object 
 4   price          15832 non-null  int64  
dtypes: float64(2), int64(1), object(2)
memory usage: 618.6+ KB
```

4.3 Feature Engineering

Data engineering for Downtown Toronto neighbours dataframe

To get data about different venues in different neighbourhoods from Downtown Toronto, the **Foursqare API** it is used. This is a location data provider that helps to obtain information about all kind of venues and events within an area of interest. Such information includes: venue names, locations, menus and even photos. For each neighbourhood, it was chosen the radius to be 500 meters.

The data retrieved from Foursquare contains information of venues within a specified distance of the longitude and latitude of the postcodes. The information obtained per venue is as follows:
Neighbourhood: Name of the Neighbourhood
Neighbourhood Latitude: Latitude of the Neighbourhood
Neighbourhood Longitude: Longitude of the Neighbourhood
Venue: Name of the Venue
Venue Latitude: Latitude of Venue
Venue Longitude: Longitude of Venue
Venue Category: Category of Venue

To use Foursquare API, the required credentials have to be defined:

```
# Loading Foursquare credentials
import yaml

with open("settings.yaml", 'r') as yamlfile:
    sett = yaml.safe_load(yamlfile)

CLIENT_ID = sett['CLIENT_ID']# your Foursquare ID
CLIENT_SECRET = sett['CLIENT_SECRET'] # your Foursquare Secre
VERSION = '20180605' # Foursquare API version
LIMIT = 100 # A default Foursquare API limit value
radius = 500 # define radius
```

Get coordinates of Downtown Toronto using geolocator and geopy libraries

To obtain latitude and longitude, I imported `Nominatim` module from `gelolocator`, define a `user agent` and pass the address to the `geocode()` method.
The geographical coordinates of Downtown Toronto are (43.6563221, -79.3809161).

Explore Downtown Toronto venues
In order to do the exploration, I create a function, **`getNearbyVenues`**, that makes a call to the **Foursqare API** to explore all restaurants from Downtown Toronto neighbourhoods, the use it to retrieve the venues.

```
def getNearbyVenues(names, latitudes, longitudes, radius=500):

    venues_list=[]
    for name, lat, lng in zip(names, latitudes, longitudes):
        #print(name)
        restaurants='4d4b7105d754a06374d81259'

        # Create the API request URL
        url = 'https://api.foursquare.com/v2/venues/explore?&client_id
={}&client_secret={}&v={}&ll={},{}&categoryId={}&radius={}&limit={}'.f
ormat(
            CLIENT_ID,
            CLIENT_SECRET,
            VERSION,
            lat,
            lng,
            restaurants,
            radius,
            LIMIT)

        # make the GET request
        results = requests.get(url).json()["response"]['groups'][0]['i
tems']

        # 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 f
or item in venue_list])
    nearby_venues.columns = ['Neighbourhood',
                  'Neighbourhood Latitude',
                  'Neighbourhood Longitude',
                  'Venue',
                  'Venue Latitude',
                  'Venue Longitude',
                  'Venue Category']

    return(nearby_venues)
```

This is how the data frame looks like, and there are 80 unique venue categories.

	Neighbourhood	Neighbourhood Latitude	Neighbourhood Longitude	Venue	Venue Latitude	Venue Longitude	Venu Catego
0	Rosedale	43.679563	-79.377529	Saigonlotus	43.681281	-79.382401	Japanes Restaura
1	St. James Town, Cabbagetown	43.667967	-79.367675	Cranberries	43.667843	-79.369407	Din
2	St. James Town, Cabbagetown	43.667967	-79.367675	Kingyo Toronto	43.665895	-79.368415	Japanes Restaura
3	St. James Town, Cabbagetown	43.667967	-79.367675	Murgatroid	43.667381	-79.369311	Restaura
4	St. James Town, Cabbagetown	43.667967	-79.367675	F'Amelia	43.667536	-79.368613	Itali Restaura

Data Engineering for Toronto's Airbnb dataframe

Looking at the Airbnb listings **dataframe**, it was observed that it does not contain a **borough column** to be used for filtering the listings. We are interested in getting the listings only for the Downtown borough.

To handle this, it is needed to find the **postal code** for each listing, then join it with the `toronto_data` **dataframe** by the **Postal Code** column. After this step, I'll be able to obtain the listings for the Downtown borough.

First step is to select only the listings that are in the boundary area of Downtown Toronto, the **second step** is to find the postal code for each listing from the resulting dataframe, and the **last step** is to merge this dataframe with the neighbourhoods dataframe.

A function that uses `geolocator.reverse()` method was created to obtain the postal codes from latitude and longitude.

```
def get_postalCode(lat, lon):
    postal_code=None
    while(postal_code is None):
        location = geolocator.reverse((lat, lon))
        if location.raw['address']['postcode']:
            postal_code=location.raw['address']['postcode'][0:3]
        else:
            postal_code=None
    return postal_code

get_postalCode(43.650571,-79.384568)
```

```
'M5H'
```

Downtown boundaries:

```
boundaries.raw['boundingbox']
```

```
['43.6463221', '43.6663221', '-79.3909161', '-79.3709161']
```

Find the postal code for each listing:

```
tobnb_downtown['PostalCode']=tobnb_downtown.apply(lambda x: get_postal
Code(x['latitude'], x['longitude']), axis=1)
```

Merge the downtown listings with downtown neighbourhoods

```
listings_conds=tobnb_downtown.merge(toronto_data, on='PostalCode', how
='inner')
```

After renaming and selecting only the column of interest, the new data frame is as following:

	PostalCode	Borough	Neighbourhood	Listing Latitude	Listing Longitude	Room Type	Price
0	M4Y	Downtown Toronto	Church and Wellesley	43.66103	-79.38340	Entire home/apt	120
1	M4Y	Downtown Toronto	Church and Wellesley	43.66420	-79.38465	Entire home/apt	198
2	M4Y	Downtown Toronto	Church and Wellesley	43.66455	-79.38533	Private room	73
3	M4Y	Downtown Toronto	Church and Wellesley	43.66445	-79.37991	Entire home/apt	64
4	M4Y	Downtown Toronto	Church and Wellesley	43.66483	-79.38388	Private room	100

Using this new data frame for merging with Downtown Toronto venues by neighbourhood, further analysis can be done.

```
# merge on neighborhoods
merged_df = listings_conds.merge(downtown_toronto_venues, on='Neighbourhood', how='inner')

# drop borough column
merged_df.drop(['Borough'], axis=1, inplace=True)

merged_df.head()
```

	PostalCode	Neighbourhood	Listing Latitude	Listing Longitude	Room Type	Price	Neighbourhood Latitude	Neighbourhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	M4Y	Church and Wellesley	43.66103	-79.3834	Entire home/apt	120	43.66586	-79.38316	Storm Crow Manor	43.666840	-79.381593	Theme Restaurant
1	M4Y	Church and Wellesley	43.66103	-79.3834	Entire home/apt	120	43.66586	-79.38316	Fabarnak	43.666377	-79.380964	Restaurant
2	M4Y	Church and Wellesley	43.66103	-79.3834	Entire home/apt	120	43.66586	-79.38316	Smith	43.666927	-79.381421	Breakfast Spot
3	M4Y	Church and Wellesley	43.66103	-79.3834	Entire home/apt	120	43.66586	-79.38316	Sansotei Ramen 三草亭	43.666735	-79.385353	Ramen Restaurant
4	M4Y	Church and Wellesley	43.66103	-79.3834	Entire home/apt	120	43.66586	-79.38316	Como En Casa	43.665160	-79.384796	Mexican Restaurant

4.4 Analysis

```
print('There are {} uniques categories.'.format(len(merged_df['Venue Category'].unique())))
```

```
There are 77 uniques categories.
```

One Hot Encoding

Since I am trying to find out what are the different kinds of venue categories present in each neighbourhood and then calculate the top five common venues, I use the One Hot Encoding to work with the categorical datatype of the venue categories. This helps to convert the categorical data into numeric data.

I performed one hot encoding and then calculated the mean of the grouped venue categories for each of the neighbourhoods.

```
# One hot encoding
toronto_onehot = pd.get_dummies(merged_df[['Venue Category']], prefix
="", prefix_sep="")

# Add neighborhood column back to dataframe
toronto_onehot['Neighbourhood'] = merged_df['Neighbourhood']

# Move neighborhood column to the first column
fixed_columns = [toronto_onehot.columns[-1]] + list(toronto_onehot.col
umns[:-1])
toronto_onehot = toronto_onehot[fixed_columns]
toronto_onehot.head()
```

	Neighbourhood	Afghan Restaurant	American Restaurant	Arepa Restaurant	Asian Restaurant	BBQ Joint	Bagel Shop	Bakery	B Rest
0	Church and Wellesley	0	0	0	0	0	0	0	
1	Church and Wellesley	0	0	0	0	0	0	0	
2	Church and Wellesley	0	0	0	0	0	0	0	
3	Church and Wellesley	0	0	0	0	0	0	0	
4	Church and Wellesley	0	0	0	0	0	0	0	

Let's group rows by neighbourhood and by taking the mean of the frequency of occurrence of each category

```
toronto_grouped = toronto_onehot.groupby('Neighbourhood').mean().reset
_index()
toronto grouped
```

	Neighbourhood	Afghan Restaurant	American Restaurant	Arepa Restaurant	Asian Restaurant	BBQ Joint	Bagel Shop	Bake
0	Berczy Park	0.000000	0.019231	0.000000	0.019231	0.000000	0.019231	0.0769
1	CN Tower, King and Spadina, Railway Lands, Har...	0.000000	0.500000	0.000000	0.000000	0.000000	0.000000	0.0000
2	Central Bay Street	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0000
3	Church and Wellesley	0.015385	0.015385	0.000000	0.000000	0.000000	0.000000	0.0000
4	Commerce Court, Victoria Hotel	0.000000	0.040000	0.000000	0.030000	0.000000	0.000000	0.0400
5	First Canadian Place, Underground city	0.000000	0.050000	0.000000	0.030000	0.000000	0.000000	0.0400

Now let's create a new data frame and display the top 5 venue categories for each neighbourhood. To do this, I used a helping function to sort the venues in descending order.

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

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

```
for idx in np.arange(toronto_grouped.shape[0]):
    neighborhoods_venues_sorted.iloc[idx, 1:] = return_most_common_ven
ues(toronto_grouped.iloc[idx, :], num_top_venues)

neighborhoods venues sorted.head()
```

The new data frame:

	Neighbourhood	1st Most common venue	2nd Most common venue	3rd Most common venue	4th Most common venue	5th Most common venue
0	Berczy Park	Bakery	Sandwich Place	Sushi Restaurant	Italian Restaurant	Restaurant
1	CN Tower, King and Spadina, Railway Lands, Har...	American Restaurant	Tapas Restaurant	Middle Eastern Restaurant	Pizza Place	Persian Restaurant
2	Central Bay Street	Café	Sandwich Place	Chinese Restaurant	Japanese Restaurant	Italian Restaurant
3	Church and Wellesley	Sushi Restaurant	Japanese Restaurant	Fast Food Restaurant	Pizza Place	Restaurant
4	Commerce Court, Victoria Hotel	Restaurant	Café	Italian Restaurant	Salad Place	Sandwich Place

4.5 Modelling and Visualization

Model building using K-means

In this part I clustered the venues categories by location of listings. Since this is analysis of unlabelled data, I made use of KMeans Clustering Machine learning algorithm to cluster similar neighbourhoods together. I used a fixed number of clusters as 5.

Run *k*-means to cluster the neighbourhood into 5 clusters.

```
# set number of clusters
kclusters = 5

toronto_grouped_clustering = toronto_grouped.drop('Neighbourhood', 1)

# run k-means clustering
kmeans = KMeans(n_clusters=kclusters, random_state=25, n_init=10).fit
(toronto_grouped_clustering)

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

```
array([0, 1, 2, 2, 0, 0, 2, 4, 0, 0])
```

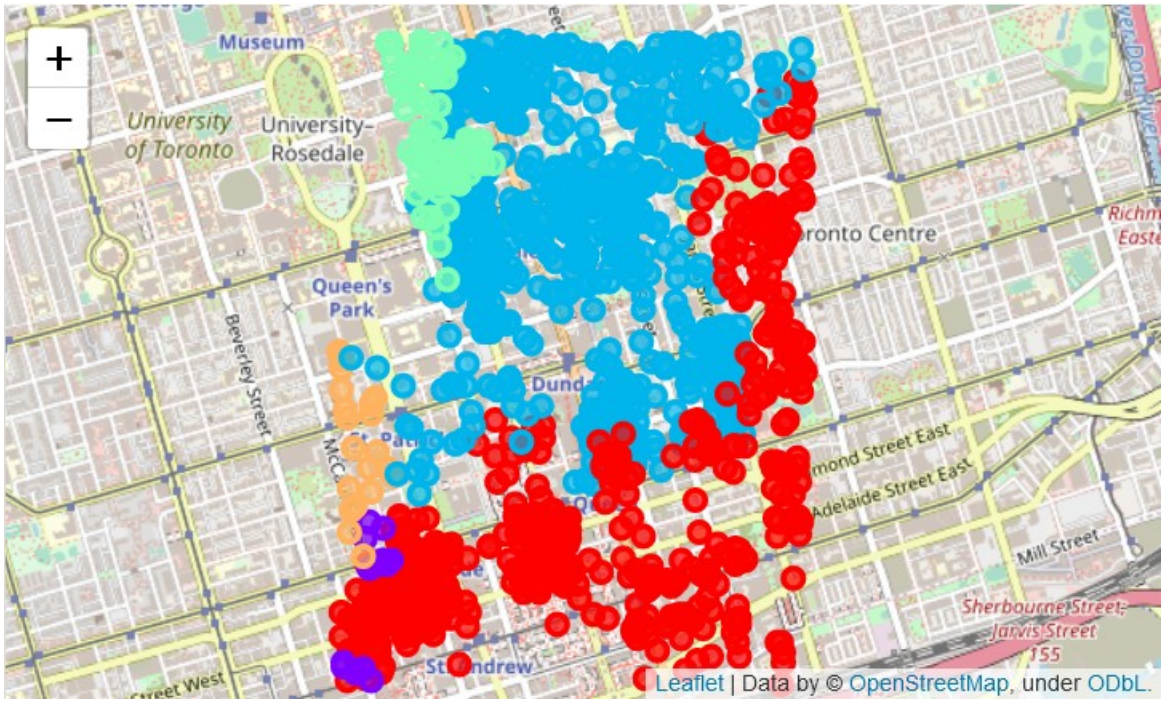
Next, I created a new data frame that includes the cluster as well as the top 5 venues for each neighbourhood.

This is the resulting data frame:

	Neighbourhood	Listing Latitude	Listing Longitude	Room Type	Price	Cluster Labels	1st Most common venue	2nd Most common venue	3rd M com ve
0	Church and Wellesley	43.66103	-79.38340	Entire home/apt	120	2	Sushi Restaurant	Japanese Restaurant	Fast F Restau
1	Church and Wellesley	43.66420	-79.38465	Entire home/apt	198	2	Sushi Restaurant	Japanese Restaurant	Fast F Restau
2	Church and Wellesley	43.66455	-79.38533	Private room	73	2	Sushi Restaurant	Japanese Restaurant	Fast F Restau
3	Church and Wellesley	43.66445	-79.37991	Entire home/apt	64	2	Sushi Restaurant	Japanese Restaurant	Fast F Restau
4	Church and Wellesley	43.66483	-79.38388	Private room	100	2	Sushi Restaurant	Japanese Restaurant	Fast F Restau

To visualize the resulting clusters I created a map using *Folium* library and Leaflet Map. Folium is a python library, used to draw an interactive leaflet map using coordinate data.

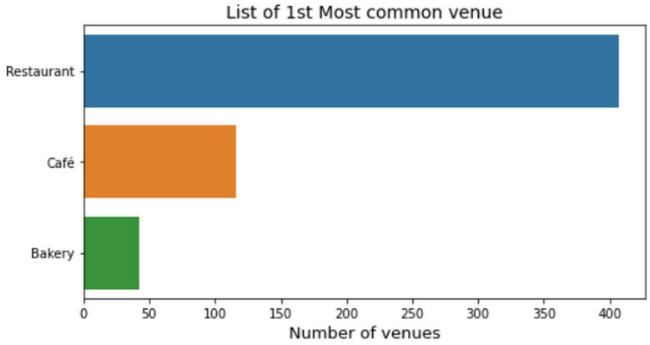
```
# create map
map_clusters = folium.Map(location = [latitude, longitude], zoom_start
=13)

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

# add markers to the map
markers_colors=[]
for lat, lon, poi, cluster, room, price in zip(toronto_merged['Listing
Latitude'],
                                               toronto_merged['Listing
Longitude'],
                                               toronto_merged['Neighbo
urhood'],
                                               toronto_merged['Cluster
Labels'],
                                               toronto_merged['Room Ty
pe'],
                                              toronto_merged['Price
']):
    label = folium.Popup(str(poi) + ' Cluster ' + str(cluster) +
                         ' (room type: ' + str(room) + ' price: ' + st
r(price) + ')', parse_html=True)
    folium.CircleMarker([lat, lon],
                        radius=5,
                        popup=label,
                        color=rainbow[cluster-1],
                        fill=True,
                        fill_color=rainbow[cluster-1],
                        fill_opacity=0.7).add_to(map_clusters)
map clusters
```

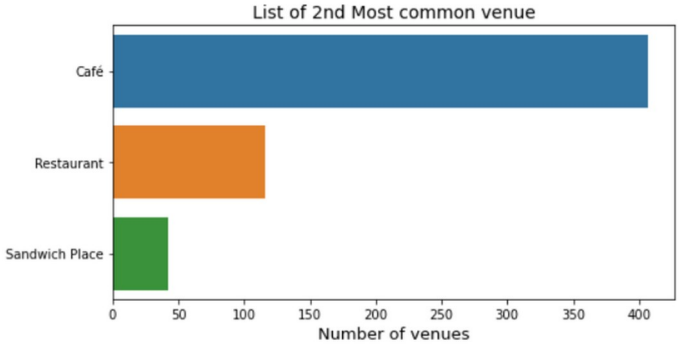


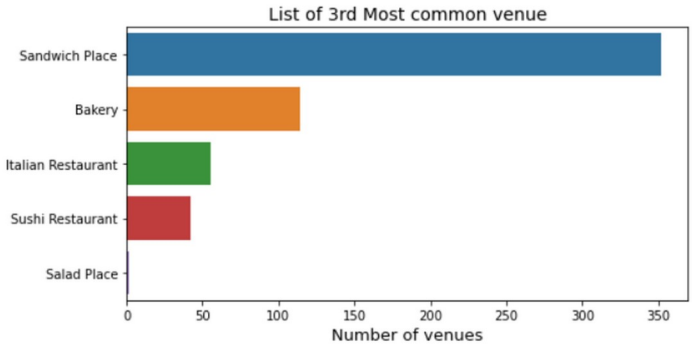
Cluster Analysis

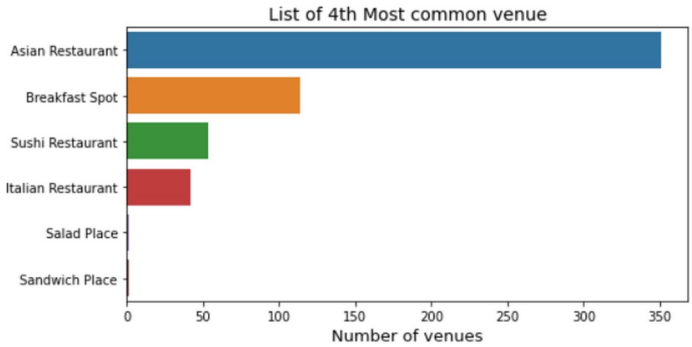
Cluster0

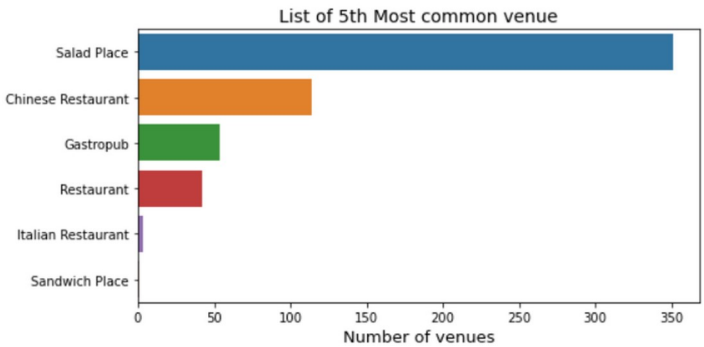
	Neighbourhood	Cluster Labels	1st Most common venue	2nd Most common venue	3rd Most common venue	4th Most common venue	5th Most common venue
439	Regent Park, Harbourfront	0	Café	Restaurant	Bakery	Breakfast Spot	Chinese Restaurant
440	Regent Park, Harbourfront	0	Café	Restaurant	Bakery	Breakfast Spot	Chinese Restaurant
441	Regent Park, Harbourfront	0	Café	Restaurant	Bakery	Breakfast Spot	Chinese Restaurant
442	Regent Park, Harbourfront	0	Café	Restaurant	Bakery	Breakfast Spot	Chinese Restaurant
443	Regent Park, Harbourfront	0	Café	Restaurant	Bakery	Breakfast Spot	Chinese Restaurant


List of 1st Most common venue
Restaurant
Café
Bakery
0
50
100
150
200
250
300
350
400
Number of venues


List of 2nd Most common venue
Café
Restaurant
Sandwich Place
0
50
100
150
200
250
300
350
400
Number of venues

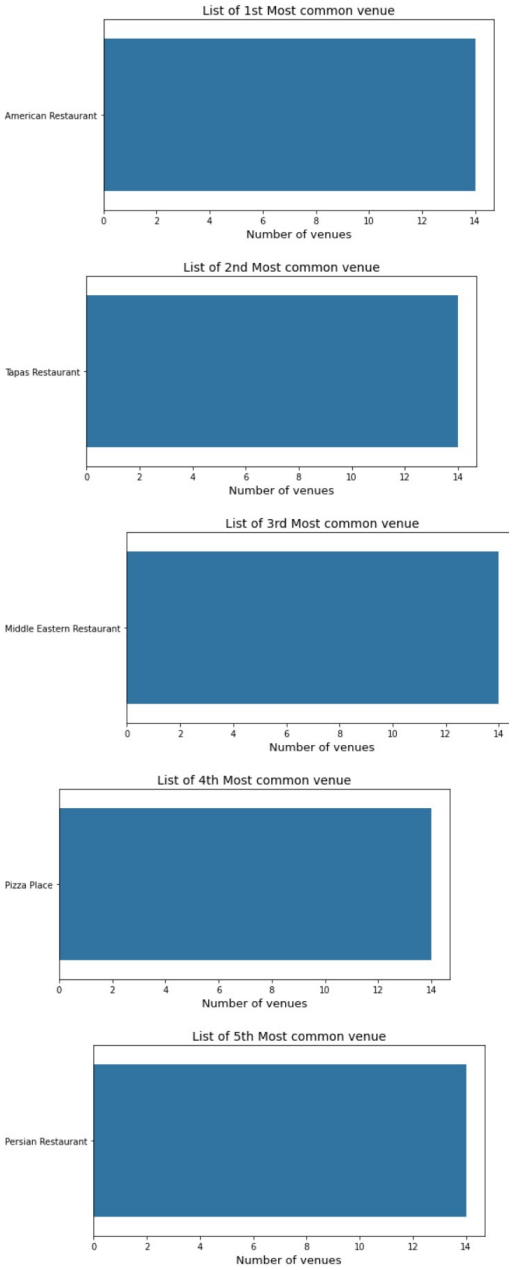

List of 3rd Most common venue
Sandwich Place
Bakery
Italian Restaurant
Sushi Restaurant
Salad Place
0
50
100
150
200
250
300
350
Number of venues


List of 4th Most common venue
Asian Restaurant
Breakfast Spot
Sushi Restaurant
Italian Restaurant
Salad Place
Sandwich Place
0
50
100
150
200
250
300
350
Number of venues


List of 5th Most common venue
Salad Place
Chinese Restaurant
Gastropub
Restaurant
Italian Restaurant
Sandwich Place
0
50
100
150
200
250
300
350
Number of venues

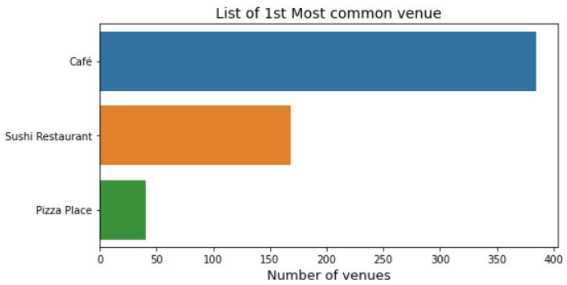
Cluster1

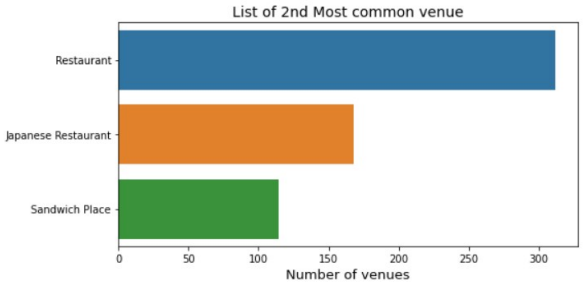
	Neighbourhood	Cluster Labels	1st Most common venue	2nd Most common venue	3rd Most common venue	4th Most common venue	5th Most common venue
946	CN Tower, King and Spadina, Railway Lands, Har...	1	American Restaurant	Tapas Restaurant	Middle Eastern Restaurant	Pizza Place	Persian Restaurant
947	CN Tower, King and Spadina, Railway Lands, Har...	1	American Restaurant	Tapas Restaurant	Middle Eastern Restaurant	Pizza Place	Persian Restaurant
948	CN Tower, King and Spadina, Railway Lands, Har...	1	American Restaurant	Tapas Restaurant	Middle Eastern Restaurant	Pizza Place	Persian Restaurant
949	CN Tower, King and Spadina, Railway Lands, Har...	1	American Restaurant	Tapas Restaurant	Middle Eastern Restaurant	Pizza Place	Persian Restaurant
950	CN Tower, King and Spadina, Railway Lands, Har...	1	American Restaurant	Tapas Restaurant	Middle Eastern Restaurant	Pizza Place	Persian Restaurant

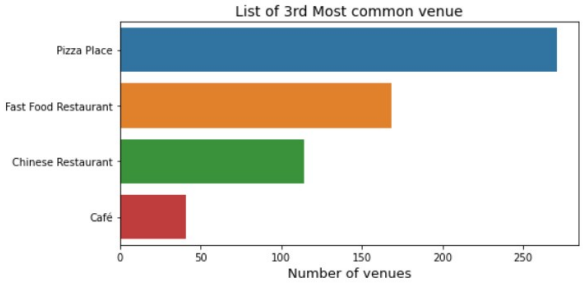


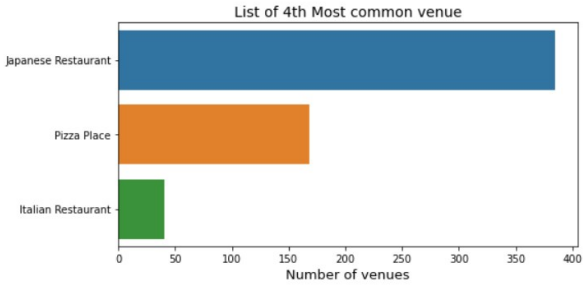
Cluster2

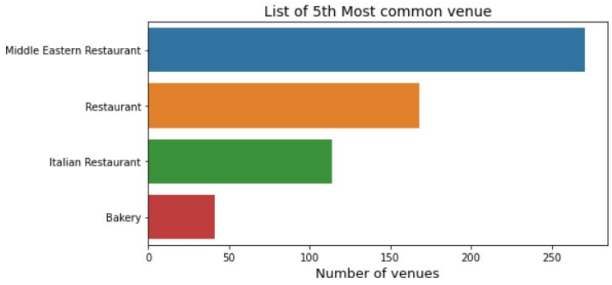
	Neighbourhood	Cluster Labels	1st Most common venue	2nd Most common venue	3rd Most common venue	4th Most common venue	5th Most common venue
0	Church and Wellesley	2	Sushi Restaurant	Japanese Restaurant	Fast Food Restaurant	Pizza Place	Restaurant
1	Church and Wellesley	2	Sushi Restaurant	Japanese Restaurant	Fast Food Restaurant	Pizza Place	Restaurant
2	Church and Wellesley	2	Sushi Restaurant	Japanese Restaurant	Fast Food Restaurant	Pizza Place	Restaurant
3	Church and Wellesley	2	Sushi Restaurant	Japanese Restaurant	Fast Food Restaurant	Pizza Place	Restaurant
4	Church and Wellesley	2	Sushi Restaurant	Japanese Restaurant	Fast Food Restaurant	Pizza Place	Restaurant





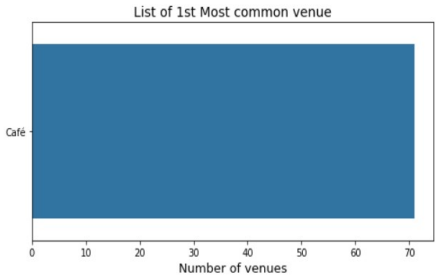


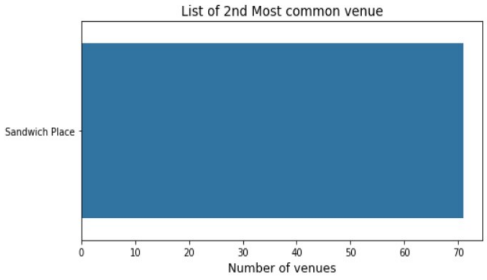


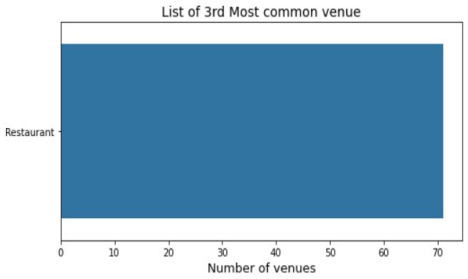


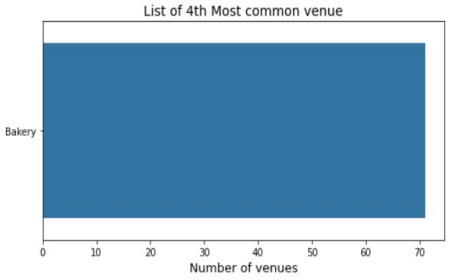
Cluster3

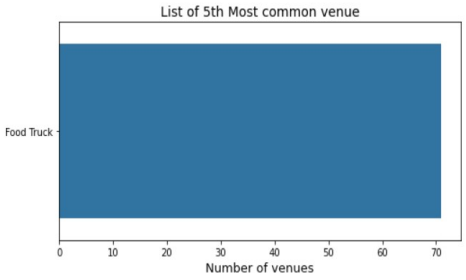
	Neighbourhood	Cluster Labels	1st Most common venue	2nd Most common venue	3rd Most common venue	4th Most common venue	5th Most common venue
1191	University of Toronto, Harbord	3	Café	Sandwich Place	Restaurant	Bakery	Food Truck
1192	University of Toronto, Harbord	3	Café	Sandwich Place	Restaurant	Bakery	Food Truck
1193	University of Toronto, Harbord	3	Café	Sandwich Place	Restaurant	Bakery	Food Truck
1194	University of Toronto, Harbord	3	Café	Sandwich Place	Restaurant	Bakery	Food Truck
1195	University of Toronto, Harbord	3	Café	Sandwich Place	Restaurant	Bakery	Food Truck





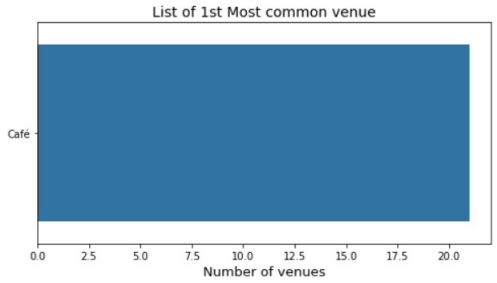


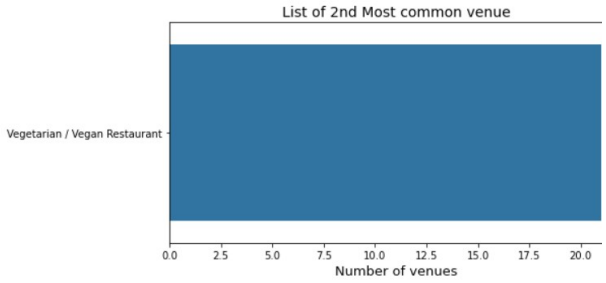


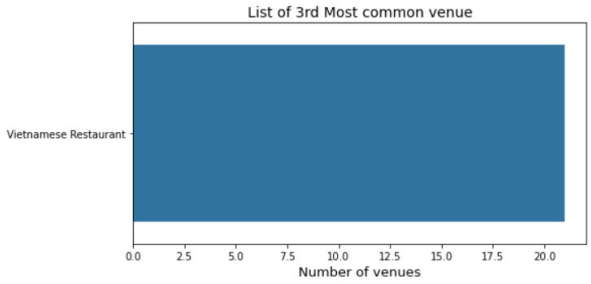


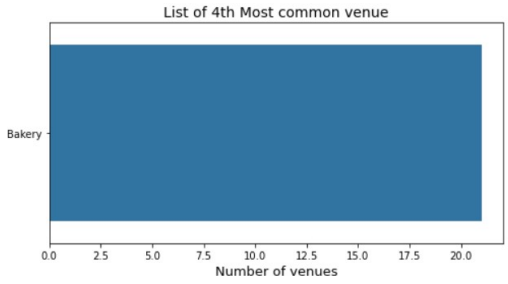
Cluster4

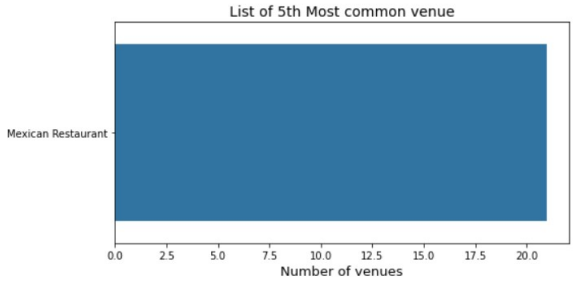
	Neighbourhood	Cluster Labels	1st Most common venue	2nd Most common venue	3rd Most common venue	4th Most common venue	5th Most common venue
960	Kensington Market, Chinatown, Grange Park	4	Café	Vegetarian / Vegan Restaurant	Vietnamese Restaurant	Bakery	Mexican Restaurant
961	Kensington Market, Chinatown, Grange Park	4	Café	Vegetarian / Vegan Restaurant	Vietnamese Restaurant	Bakery	Mexican Restaurant
962	Kensington Market, Chinatown, Grange Park	4	Café	Vegetarian / Vegan Restaurant	Vietnamese Restaurant	Bakery	Mexican Restaurant
963	Kensington Market, Chinatown, Grange Park	4	Café	Vegetarian / Vegan Restaurant	Vietnamese Restaurant	Bakery	Mexican Restaurant
964	Kensington Market, Chinatown, Grange Park	4	Café	Vegetarian / Vegan Restaurant	Vietnamese Restaurant	Bakery	Mexican Restaurant



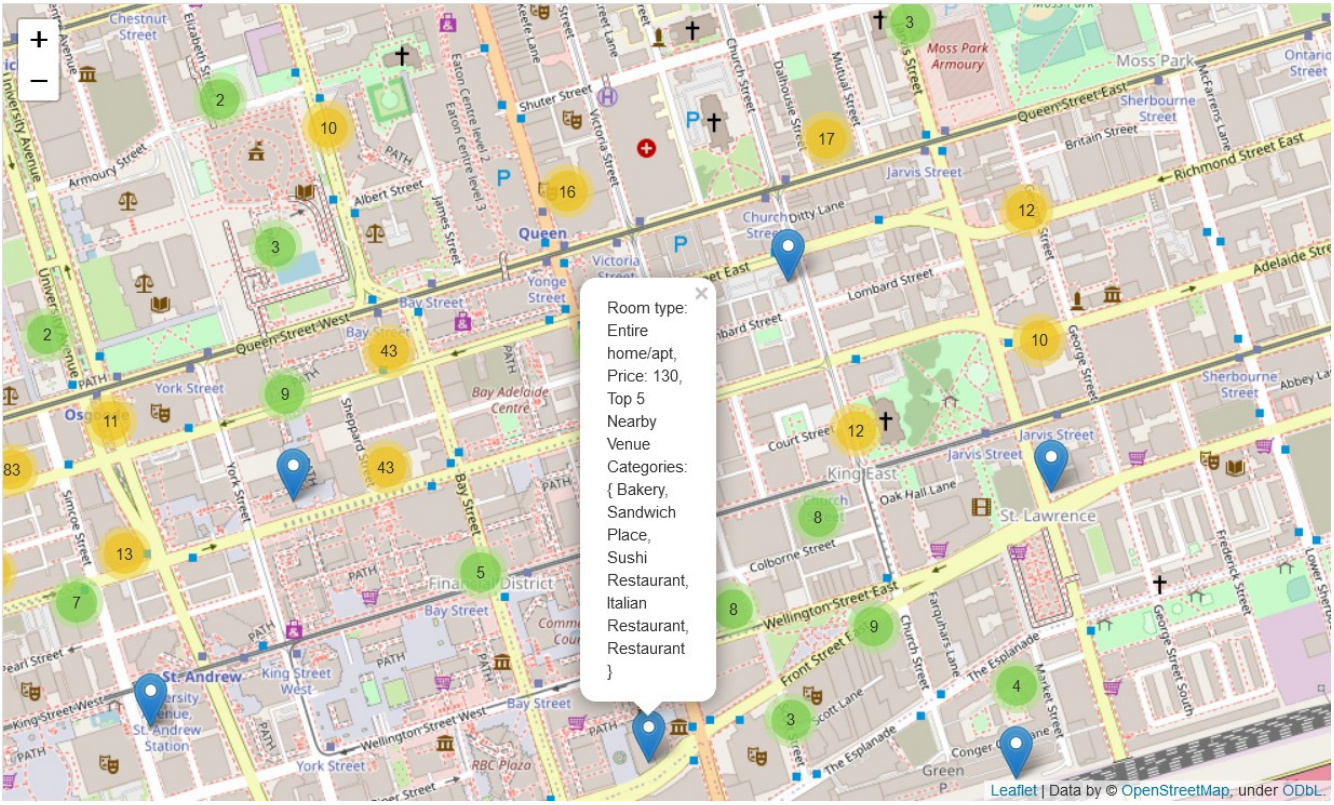








Let's pick up cluster0 and create a map of all listings



5. Results and Discussion

Results

In the result section I can document all the findings from above clustering & visualization of the data. In this project, as the business problem stated, I started with identifying the Airbnb listing from Downtown Toronto and food places around them.
I looked into Toronto neighbourhoods and selected those of interest, then found the listings associated with Downtown area. I have used data from web resources like Wikipedia, geospatial coordinates of Toronto neighbourhoods, Toronto Airbnb listings, and Foursquare API, to set up a very realistic data-analysis scenario. The following findings are the result of the analysis:

- there are **19** neighbourhoods in Downtown Toronto
- the total number of **Airbnb** listings in Toronto area is **15832**
- the number of **Airbnb** listing in Downtown area is **1265**
- the number of neighbourhoods from Downtown area that have listings is **14**
- the total number of Downtown venues is **80**
- there are **77** unique food categories associated with these listings.

  There are five clusters of food categories. Plotting the number of top five number of food categories for each cluster we can see that:

- cluster0 and cluster2 have more variety of restaurants like Italian, Asian, coffee places

- cluster1 has a limited options of restaurant types, the most common is the American cuisine.
- most of the places in cluster3 are places where people can buy a coffee and have a sandwich
- cluster4 most common places are also coffee shops and vegetarian/vegan cuisine

If travellers are more interested in fancy restaurants they can choose listings from cluster0 or cluster2. If they want to eat local, the cluster1 is a good option to choose from. If they like to enjoy a cup of coffee and have a snack, cluster3 is a good option, and finally if they are more into vegetarian cuisine, they can opt for cluster4.

Discussion

The project is a good practice opportunity in terms of working with Foursquare API, Folium library, and clustering method in machine learning. The K-Means algorithm is an unsupervised learning technique and simple algorithm capable of clustering the dataset very quickly and efficiently, often in just a few iterations.
Choosing different number of clusters (n_clusters) effectively affects the result.
The project approaches the k-means method to clustering as the end goal. The project does not intentionally find the best k in terms of k-means.

6. Conclusion

In this project, I've got the chance to work on a business problem like a real data scientist would do. I was able to find locations of places to rent in Downtown Toronto **Airbnb** where are also near food places of interest using Foursquare API, **Folium**, and machine learning approaches.

I used many python libraries to fetch the data, to manipulate, analyses and visualize the content. I made use of Foursquare API to explore the nearby venues of Downtown Toronto, then got data from Wikipedia and visualized using **matplotlib** plotting library. I also applied machine learning technique to predict the output given the data and used **Folium** to visualize it on a map.

I obtained lists of the most common food places in Downtown Toronto for each cluster.

I picked a cluster to visualize on a map the listings, and for each one, to show the top 5 venues available.

Also, some of the drawbacks or areas of improvements shows us that this analysis can further be improved with the help of more data and different machine learning technique. Similarly we can use this project to analysis any scenario such finding venues from different venue categories such as Arts & Entertainment, Outdoors & Recreation, or even College & University.