Association Rule Analysis of Venues & Venue Categories of Toronto Neighborhoods

April 30, 2019

1 Association Rule Analysis of Venues & Venue Categories of Toronto Neighborhoods

2 Table of contents

2.1 1. Introduction

Apriori is a simple and powerful statistical unsupervised learning algorithm that was developed to implement association rule mining and analysis for finding frequent item sets in a given data set. Apriori algorithm are mostly applied in business analysis, auto-complete applications like google search, recommender systems and data mining such as market basket analysis to understand the customer buying behavior and to find the relationships between items. It really is about the analysis of a pattern like, people who bought something also bought something else or watched something also watched something else, or who did something also did something else etc. So this whole association rule learning is all about analyzing when things come in pairs or in triplicates or when they are combined together for some reason, looking for those rules and those ways that this happens. In this article I will focus on applying the Apriori learning algorithm to find the association pattern among different venues and venue categories in Toronto Neighborhood.

2.2 2. Methodology: Data Collection, Exploratory Data Analysis & Apriori Association Rule Analysis

2.1 Data collection This project focuses first on applying well-known machine learning algorithms to the dataset available from Wikipedia & Foursquare API. The first task is to define the data requirements for the Apriori Association Rule Analysis on the neighborhoods of the Toronto area. #### **2.2. Exploratory Data Analysis** Exploratory Data Analysis (EDA) techniques such as descriptive statistics and visualization can be applied to the data set, to assess the content, quality, and initial insights about the data. Gaps in data will be identified and plans to either fill or make substitutions will have to be made #### **2.3 Apriori Association Rule Analysis** Apriori Association rules analysis is a technique to uncover how items are associated to each other. The association can be measured using three major components;

Support: This is the measure of association based on the proportion of the occurrence of X out of the total occurrences. Support refers to the popularity of a venue, X (or venue category, X) and can be calculated by finding number of occurrences containing a particular venue divided by total number of occurrences. *Support(X) = (Occurrences containing (X))/(Total number of Occurrences)*

Confidence: This measure the likelihood of one item X relative to another item Y. This says how likely X will be found when Y has appeared, expressed as {X -> Y}. This is measured by the proportion of occurrences with item X, in which item Y also appears. In other words, number of occurrences in which both X and Y occurs divided by the total number of occurrence in which Y occurs. This is similar to the Naive Bayes (NB) algorithm, however, the two algorithms are meant for different types of problems, for example NB is used for supervised classification problems and Apriori as mentioned before, for unsupervised association rule analysis. *Confidence(Y X) = (Occurrences containing both (X and Y))/(Occurrences containing Y)*

Lift: This measures the likelihood of an occurrence X occurring out of total occurrences in which Y has occurred. In other words, the number of occurrences that contain both X and Y. This says how likely item X is occurring when item Y occurs, while controlling for how popular item X is. Lift(Y -> X) refers to the increase in the ratio of appearance of X where Y can be found. Lift(Y –> X) can be calculated by dividing Confidence(Y-> X) divided by Support(X). It is mathematically represented as: *Lift(YX) = (Confidence (YX))/(Support (X))*

Lift basically indicates that the likelihood of occurring both X and Y together is a particular times more than the likelihood of just occurring the X. A Lift of 1 means there is no association between the categories X and Y. Lift of greater than 1 means categories X and Y are more likely to be appearing together. Finally, Lift of less than 1 refers to the case where two categories are unlikely to be visible together.

2.3 3. Data Collection

2.3.1 Import Libraries & Modules

In [1]:
```
from geopy.geocoders import Nominatim # convert an address into latitude and longitude values
import folium # map rendering library
import requests # library to handle requests
from bs4 import BeautifulSoup
import numpy as np
import pandas as pd
import json # library to handle JSON files
from pandas.io.json import json_normalize # tranform JSON file into a pandas dataframe
# Matplotlib and associated plotting modules
```

```
import matplotlib.cm as cm
import matplotlib.colors as colors
import matplotlib.pyplot as plt
import seaborn as sns
print('Libraries imported.')
```

Libraries imported.

2.3.2 3.1 Foursquare location data

I start with using the Foursquare API, to search for the type of venues or stores in the city of Toronto. To utilize the Foursquare location data, I need to get the latitude and the longitude coordinates of each neighborhood. To convert the Toronto address into latitude and longitude values, a search engine for OpenStreetMap data geocoding tool named Nominatim is employed. By making the call to the database entering the developer account credentials, which are my Client ID and Client Secret as well as what is called the version of the API. Again because I'm searching for different type of venues, I pass the Nominatim returned location latitude and longitude coordinates along with the search query radius & limits. This completes the URI to make the call to the database and in return a .JSON file format of the venues that match the query with its name, unique ID, location, and category information is downloaded.

In [2]:
```
from geopy.geocoders import Nominatim
geolocator = Nominatim(user_agent="Toronto_explorer")
address = 'Toronto'
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude
print('The geograpical coordinate of Toronto are {}, {}.'.format(latitude, longitude))
```

The geograpical coordinate of Toronto are 43.653963, -79.387207.

In [3]:
```
# @hidden_cell
```

Your credentails: Kept Hidden

I get a .json file of the venues with its name, unique ID, location, and category. Apply the get_category_type function from the Foursquare lab, followed by cleaning the .JSON file and structuring it into a pandas dataframe.

In [4]:
```
# function that extracts the category of the venue
def get_category_type(row):
    try:
        categories_list = row['categories']
    except:
        categories_list = row['venue.categories']
```

```
        if len(categories_list) == 0:
            return None
        else:
            return categories_list[0]['name']

In [5]: def getNearbyVenues(names, latitudes, longitudes, radius=500):
        venues_list=[]
        for name, lat, lng in zip(names, latitudes, longitudes):
            print(name)
            # create the API request URL
            url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={}
                CLIENT_ID,
                CLIENT_SECRET,
                VERSION,
                lat,
                lng,
                radius,
                LIMIT)
            # make the GET request
            results = requests.get(url).json()["response"]['groups'][0]['items']
            # return only relevant information for each nearby venue
            venues_list.append([(
                name,
                lat,
                lng,
                v['venue']['name'],
                v['venue']['location']['lat'],
                v['venue']['location']['lng'],
                v['venue']['categories'][0]['name']) for v in results])
        nearby_venues = pd.DataFrame([item for venue_list in venues_list for item in venue_list])
        nearby_venues.columns = ['Neighborhood',
                      'Neighborhood Latitude',
                      'Neighborhood Longitude',
                      'Venue',
                      'Venue Latitude',
                      'Venue Longitude',
                      'Venue Category']
        return(nearby_venues)
```

2.3.3 3.2 Wikipedia List of Postal Codes of Canada

In order to explore and cluster the neighborhoods in Toronto, the Toronto neighborhood data of postal codes of each neighborhood along with the borough name and neighborhood name, is obtained from the Wikipedia page, https://en.wikipedia.org/wiki/List_of_postal_codes_of_Canada:_M,.

Using BeautifulSoup package in Python the table on the Wikipedia page is scrapped which is then wrangled, cleaned, and then read into a structured format like pandas dataFrame. I use the 'request' and 'BeautifulSoup' libraries to get the 'lxml' file and then find the all table tags, build

a loop to extract all data and store as dictionary for subsequent dataframe generation. Once the data is in a structured format, the analysis is done to explore and cluster the neighborhoods in the city of Toronto.

In [6]:
```
#Target page
website_url = requests.get('https://en.wikipedia.org/wiki/List_of_postal_codes_of_Canada:_M').text

#Use BeautifulSoup to parse HTML
soup = BeautifulSoup(website_url,'lxml')

#Extract table from wiki
Extract = soup.find('table',class_='wikitable sortable')

#Create table and populate with lines from extract
table_list = []
for rows in Extract.find_all('td'):
    row = rows.text
    row = row.replace('\n', '')
    table_list.append(row)

    #Create empty pandas dataframe with field titles
column_names = ['PostalCode', 'Borough', 'Neighborhood']
toronto_data=[]
toronto_data=pd.DataFrame(columns=column_names)

#Populate df with table rows
toronto_data.iloc[:,0]=table_list[::3]
toronto_data.iloc[:,1]=table_list[1::3]
toronto_data.iloc[:,2]=table_list[2::3]
toronto_data.replace("Not assigned", np.nan, inplace = True)
toronto_data.dropna(subset=["Borough"], axis=0, inplace = True)
toronto_data.reset_index(drop=True, inplace=True)

#Where Neighborhood is not assigned use Borough
for i in range(0, toronto_data.shape[0]):
    if pd.isnull(toronto_data.loc[i,'Neighborhood']):
        toronto_data.replace(toronto_data.loc[i,'Neighborhood'], toronto_data.loc[i,'Borough'],inplace=True

#Cleanse Neighborhood column based on unique postcodes/boroughs and append neighborhoods in each u
toronto_data['Neighborhood'] = toronto_data[['PostalCode','Borough','Neighborhood']].groupby(['PostalCo
toronto_data.drop_duplicates(inplace=True)
toronto_data.reset_index(drop=True, inplace=True)
toronto_data.head()
# print number of rows of dataframe
print(toronto_data.shape)
```

(103, 3)

2.3.4 3.3 Geospatial Data

From the csv file, a dataFrame containing the geographical coordinates corresponding to each postal code is created.

```
In [7]: Geospatial_data = pd.read_csv("http://cocl.us/Geospatial_data")
        neighborhoods = toronto_data.merge(Geospatial_data, how = "left", left_on = "PostalCode", right_on =
        neighborhoods = neighborhoods.drop(["Postal Code"], axis = 1)
        neighborhoods.head()
```

Out[7]:

	PostalCode	Borough	Neighborhood	Latitude	\
0	M3A	North York	Parkwoods	43.753259	
1	M4A	North York	Victoria Village	43.725882	
2	M5A	Downtown Toronto	Harbourfront, Regent Park	43.654260	
3	M6A	North York	Lawrence Heights, Lawrence Manor	43.718518	
4	M7A	Queen's Park	Queen's Park	43.662301	

	Longitude
0	-79.329656
1	-79.315572
2	-79.360636
3	-79.464763
4	-79.389494

Combining all three sources of data after applying getNearbyVenues function which I have already defined before, a new dataFrame, Toronto_venues is obtained.

```
In [8]: Toronto_venues = getNearbyVenues(names=neighborhoods['Neighborhood'],
                                         latitudes=neighborhoods['Latitude'],
                                         longitudes=neighborhoods['Longitude']
                                        )
        print(Toronto_venues.shape)
        Toronto_venues.head()
```

```
Parkwoods
Victoria Village
Harbourfront, Regent Park
Lawrence Heights, Lawrence Manor
Queen's Park
Islington Avenue
Rouge, Malvern
Don Mills North
Woodbine Gardens, Parkview Hill
Ryerson, Garden District
Glencairn
Cloverdale, Islington, Martin Grove, Princess Gardens, West Deane Park
Highland Creek, Rouge Hill, Port Union
Flemingdon Park, Don Mills South
Woodbine Heights
```

St. James Town
Humewood-Cedarvale
Bloordale Gardens, Eringate, Markland Wood, Old Burnhamthorpe
Guildwood, Morningside, West Hill
The Beaches
Berczy Park
Caledonia-Fairbanks
Woburn
Leaside
Central Bay Street
Christie
Cedarbrae
Hillcrest Village
Bathurst Manor, Downsview North, Wilson Heights
Thorncliffe Park
Adelaide, King, Richmond
Dovercourt Village, Dufferin
Scarborough Village
Fairview, Henry Farm, Oriole
Northwood Park, York University
East Toronto
Harbourfront East, Toronto Islands, Union Station
Little Portugal, Trinity
East Birchmount Park, Ionview, Kennedy Park
Bayview Village
CFB Toronto, Downsview East
The Danforth West, Riverdale
Design Exchange, Toronto Dominion Centre
Brockton, Exhibition Place, Parkdale Village
Clairlea, Golden Mile, Oakridge
Silver Hills, York Mills
Downsview West
The Beaches West, India Bazaar
Commerce Court, Victoria Hotel
Downsview, North Park, Upwood Park
Humber Summit
Cliffcrest, Cliffside, Scarborough Village West
Newtonbrook, Willowdale
Downsview Central
Studio District
Bedford Park, Lawrence Manor East
Del Ray, Keelesdale, Mount Dennis, Silverthorn
Emery, Humberlea
Birch Cliff, Cliffside West
Willowdale South
Downsview Northwest
Lawrence Park
Roselawn

The Junction North, Runnymede
Weston
Dorset Park, Scarborough Town Centre, Wexford Heights
York Mills West
Davisville North
Forest Hill North, Forest Hill West
High Park, The Junction South
Westmount
Maryvale, Wexford
Willowdale West
North Toronto West
The Annex, North Midtown, Yorkville
Parkdale, Roncesvalles
Canada Post Gateway Processing Centre
Kingsview Village, Martin Grove Gardens, Richview Gardens, St. Phillips
Agincourt
Davisville
Harbord, University of Toronto
Runnymede, Swansea
Clarks Corners, Sullivan, Tam O'Shanter
Moore Park, Summerhill East
Chinatown, Grange Park, Kensington Market
Agincourt North, L'Amoreaux East, Milliken, Steeles East
Deer Park, Forest Hill SE, Rathnelly, South Hill, Summerhill West
CN Tower, Bathurst Quay, Island airport, Harbourfront West, King and Spadina, Railway Lands, South Niagara
Humber Bay Shores, Mimico South, New Toronto
Albion Gardens, Beaumond Heights, Humbergate, Jamestown, Mount Olive, Silverstone, South Steeles, Thistleto
L'Amoreaux West
Rosedale
Stn A PO Boxes 25 The Esplanade
Alderwood, Long Branch
Northwest
Upper Rouge
Cabbagetown, St. James Town
First Canadian Place, Underground city
The Kingsway, Montgomery Road, Old Mill North
Church and Wellesley
Business Reply Mail Processing Centre 969 Eastern
Humber Bay, King's Mill Park, Kingsway Park South East, Mimico NE, Old Mill South, The Queensway East, R
Kingsway Park South West, Mimico NW, The Queensway West, Royal York South West, South of Bloor
(2265, 7)

Out[8]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	\
0	Parkwoods	43.753259	-79.329656	
1	Parkwoods	43.753259	-79.329656	
2	Parkwoods	43.753259	-79.329656	
3	Victoria Village	43.725882	-79.315572	

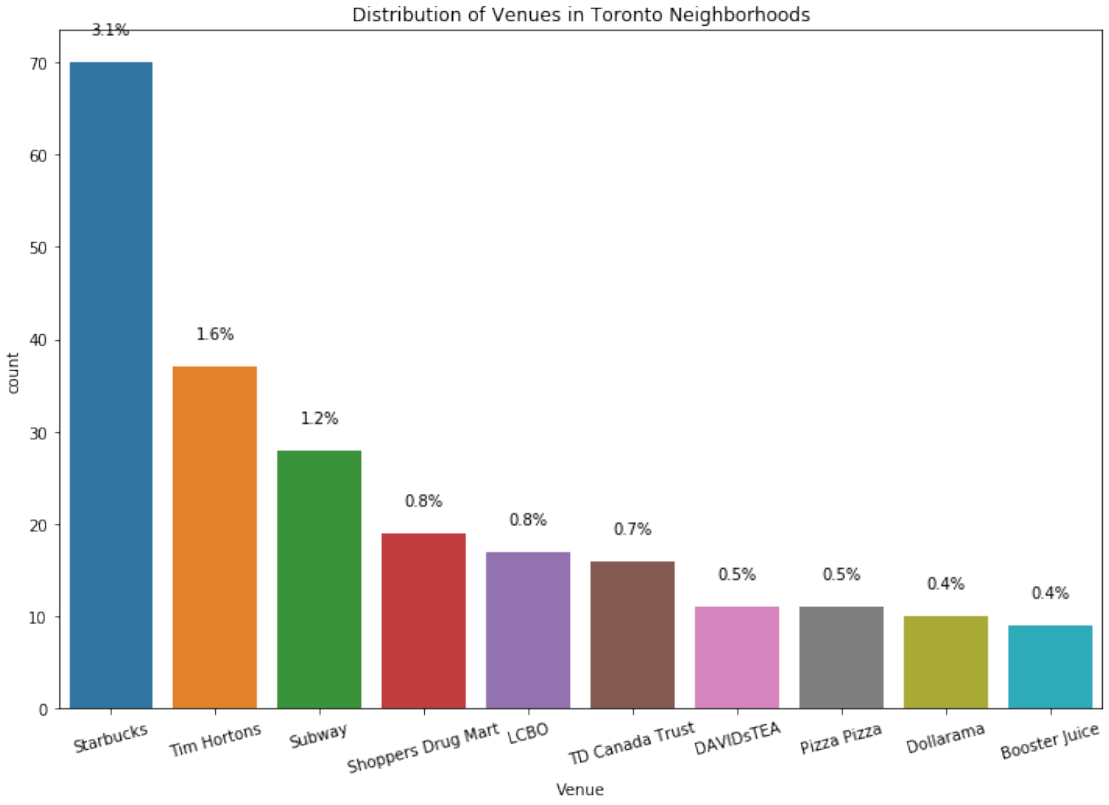
```
4  Victoria Village          43.725882          -79.315572

                  Venue  Venue Latitude  Venue Longitude  \
0        Brookbanks Park       43.751976       -79.332140
1                    KFC       43.754387       -79.333021
2          Variety Store       43.751974       -79.333114
3  Victoria Village Arena       43.723481       -79.315635
4           Tim Hortons       43.725517       -79.313103

       Venue Category
0                Park
1  Fast Food Restaurant
2     Food & Drink Shop
3         Hockey Arena
4          Coffee Shop
```

2.4 4. Results & Discussion

2.4.1 4.1 Exploratory Data Analysis (EDA)

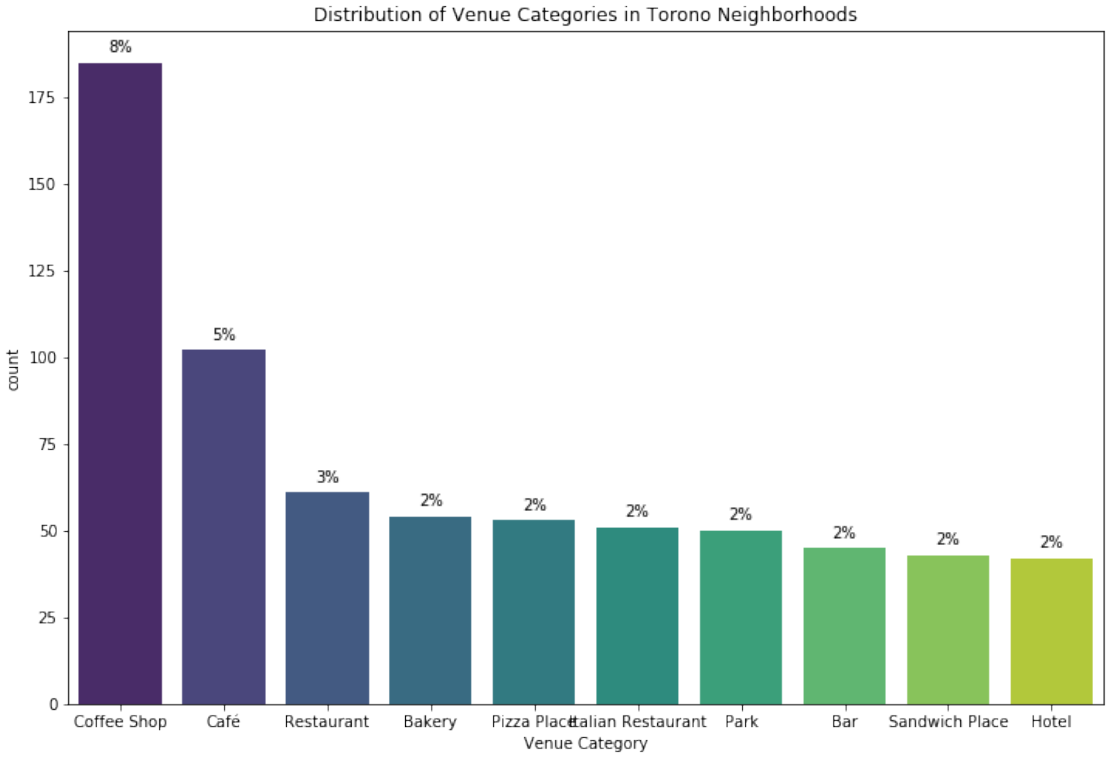
Exploratory data analysis (EDA)is done using Matplotlib, Seaborn, Folium Heatmap & Folium Choropleth.

```
In [9]: plt.figure(figsize=(12,8))
        plt.title('Distribution of Venues in Toronto Neighborhoods')
        plt.xticks(rotation= 15)
        ax = sns.countplot(data=Toronto_venues, x = 'Venue', order=Toronto_venues["Venue"].value_counts().ilo
        total = float(len(Toronto_venues))
        for p in ax.patches:
            height = p.get_height()
            ax.text(p.get_x()+p.get_width()/2.,
                    height + 3,
                    '{0:1.1%}'.format(height/total),
                    ha="center")
```



The seaborn count plot shows that Starbucks, followed by Tim Hortons & Subway are the 3 top most frequent Venues in the city of Toronto. Starbucks accounts for the 3.1% of all the venues in the city.

```
In [10]: plt.figure(figsize=(12,8))
         ax = sns.countplot(data=Toronto_venues, x = 'Venue Category', order=Toronto_venues["Venue Category
         plt.title('Distribution of Venue Categories in Torono Neighborhoods')
         total = float(len(Toronto_venues))
         for p in ax.patches:
             height = p.get_height()
             ax.text(p.get_x()+p.get_width()/2.,
                     height + 3,
                     '{0:1.0%}'.format(height/total),
                     ha="center")
```



Count plot for venue categories gives an idea that Toronto neighborhoods venues have 8% Coffee shops, 5% Café categories, followed by 3% Restaurant categories.

Heat Map of Latitudes & Longitudes of Points of Interest A Heat map of venues(Figure 3) is generated using latitude and longitude values obtained from the merged data of Foursquare data. Markers are added to specify the location of the venues. To do that, a for loop is used to iterate through each row and then adding them to the Folium map. The columns to be iterated are specified and hence all the blue marks representing venue positions are superimposed on top of the map. Labels are also added to these markers in order to let the readers know what they actually represents. This is done by using the marker function and the pop up parameter to pass in venue names to add to this marker.

```
In [11]: # Isolating "northing" and "easting" information with labels for each point
         locationlist = Toronto_venues[["Venue Latitude","Venue Longitude"]].values.tolist()
         location1 = Toronto_venues['Venue Latitude'].mean(), Toronto_venues['Venue Longitude'].mean()
         labels = Toronto_venues["Venue"].values.tolist()

         import folium
         # create map of Toronto using latitude and longitude values
         map_Toronto = folium.Map(location=location1, zoom_start=11)
         # Create map and drop points onto it
         for point in range(len(locationlist)):
             popup = folium.Popup(labels[point], parse_html=True)
             folium.Marker(locationlist[point], popup=popup).add_to(map_Toronto)
```

```
from folium import plugins
# convert to (n, 2) nd-array format for heatmap
stationArr = Toronto_venues[['Venue Latitude', 'Venue Longitude']].values

# plot heatmap
map_Toronto.add_child(plugins.HeatMap(stationArr.tolist(), radius=35))
map_Toronto
```

Out[11]: <folium.folium.Map at 0x2911cdcec50>

Choropleth Map to show the Distribution of Venues across the City of Toronto I create a Choropleth map (Figure 4) for Toronto venue data obtained from Foursquare API. As this requires structured data in .GEOJSON format as an input, I generated one, fetching a shape file of Census Boundary Files data for Canadian FSAs (Forward Sortation Area—the first three digits of the Canadian Postal Code) from a publicly accessible API endpoint provided by Statistics Canada. The .shp file downloaded is then converted to .GEOJSON file using QGis, filtering all Toronto CFSAUID (FSA) corresponding to the postal codes in the dataFrame, Toronto_data.

Linking the FSA stored in the key feature.properties.CFSAUID, a choropleth map of Toronto venues, which provides a detailed information about venue distribution in each neighborhood is generated. Choropleth map is superimposed on top of Folium, a Python map rendering Library that is handy to visualize spatial data in an interactive manner, straight within the notebooks environment. It is quite straight forward provided a GeoJSON file for the area is available. The default map style is the open street map, which shows a street view of an area when it is zoomed in. First a map centered around Toronto is created by passing in the latitude and the longitude values of Toronto using the location parameter. With Folium the initial zoom level can be set and here I use the zoom start parameter as 11. The zoom level can be changed easily after the map is rendered by zooming in or zooming out.

In [12]:
```
VenueDensity = Toronto_venues.groupby('Neighborhood').count().sort_values(by = "Venue Category", a
VenueDensity = VenueDensity.reset_index()
VenueCount = VenueDensity.merge(neighborhoods, how = "inner", on = "Neighborhood")
pdf1 = VenueCount.drop(["Borough"], axis = 1)

# create choropleth map of Toronto Venue density
location = pdf1['Latitude'].mean(), pdf1['Longitude'].mean()
map_toronto = folium.Map(location=location, zoom_start=11)
toronto_geo = "Toronto.geojson"
map_toronto.choropleth(geo_data=toronto_geo,
    data = pdf1,
    columns=['PostalCode','Venue'],
    key_on='feature.properties.CFSAUID',
    fill_color='OrRd',
    fill_opacity=0.7,
    line_opacity=0.2,
    legend_name='Toronto Venue density by Postal Codes')

map_toronto
```

Out[12]: <folium.folium.Map at 0x2911cd31a90>

As can be seen from the map most of the venues are densely located on the Old Toronto Neighborhood side.

2.4.2 4.2 Analyze Each Neighborhood to build the datasets for Apriori Algorithm

Dummy variables are created for each venue & category, followed by one hot encoding. The rows are grouped by neighborhood and by taking the mean of the frequency of occurrence of each category. To generate 20 most common venues & venue categories in each Neighborhood I then define a function named return_most_common_venues. I limit the number to 20 because the Apriori algorithm stops working when I use too many venues & categories.

In [13]:
```
# Top 20 Venues in the City of Toronto
# one hot encoding
Toronto_onehotvenue = pd.get_dummies(Toronto_venues[['Venue']], prefix="", prefix_sep="")

# add neighborhood column back to dataframe
Toronto_onehotvenue['Neighborhood'] = Toronto_venues['Neighborhood']
Toronto_groupedvenue = Toronto_onehotvenue.groupby('Neighborhood').mean().reset_index()

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

    return row_categories_sorted.index.values[0:num_top_venues]

num_top_venues = 20

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

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

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

for ind in np.arange(Toronto_groupedvenue.shape[0]):
    neighborhoods_venues_sorted.iloc[ind, 1:] = return_most_common_venues(Toronto_groupedvenue.ilo
print("The list of top 20 most common venues for each neighborhood.")
neighborhoods_venues_sorted.head()
```

The list of top 20 most common venues for each neighborhood.

Out[13]:

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue	9th Most Common Venue	...	11th Most Common Venue	12th Most Common Venue	13th Most Common Venue	14th Most Common Venue
0	Adelaide, King, Richmond	Starbucks	M Square Coffee Co	Textile Museum of Canada	Tachi	Daily Press Juicery	Design Exchange	Sud Forno	Dineen @CommerceCourt	Dineen Coffee	...	South Street Burger	Earls Kitchen & Bar	Soho House Toronto	Eggspectation Bell Trinity Square
1	Agincourt	Royal Chinese Seafood Restaurant	Panagio's Breakfast & Lunch	Commander Arena	Twilight	Subway	Lucky Moose Food Mart	Grover Pub and Grub	Great North American Grill	Greenhawk on the Avenue	...	Greenwood Cigar & Variety	Grey Gardens	Grill It Up	Gushi
2	Agincourt North, L'Amoreaux East, Milliken, St...	Lickety's	Milliken Public School Playground	Port Royal Park	Goodfellas Wood Oven Pizza	Granite Brewery	Great Grassy Grounds	Great North American Grill	Greenhawk on the Avenue	Greens Vegetarian Restaurant	...	Grey Gardens	Grill It Up		
3	Albion Gardens, Beaumond Heights, Humbergate, ...	Sheriff's No Frills	Subway	Sunny Foodmart	Pizza Pizza	Shoppers Drug Mart	The Beer Store	McDonald's	Popeyes Louisiana Kitchen	Lucky Moose Food Mart	...	Granite Brewery	Great Grassy Grounds		
4	Alderwood, Long Branch	Il Paesano Pizzeria & Restaurant	Timothy's Pub	Tim Hortons	Subway	Pizza Pizza	Franklin Horner Community Centre	Toronto Gymnastics International	Sir Adam Beck Rink	Rexall	...	Grey Gardens	Great Grassy Grounds		

```
2   Lucky Moose Food Mart                    Gushi
3    Great North American Grill      Greenhawk on the Avenue
4    Great North American Grill      Greenhawk on the Avenue

        15th Most Common Venue     16th Most Common Venue  \
0          Shangri-La Toronto                    Ematei
1             Granite Brewery             Gusto Pizza
2                Gusto Pizza                      Gym
3  Greens Vegetarian Restaurant  Greenwood Cigar & Variety
4  Greens Vegetarian Restaurant  Greenwood Cigar & Variety

  17th Most Common Venue 18th Most Common Venue      19th Most Common Venue  \
0     Equinox Bay Street       Estiatorio Volos  Sam James Coffee Bar (SJCB)
1                    Gym  Gyu-Kaku Japanese BBQ                      Gyubee
2  Gyu-Kaku Japanese BBQ                Gyubee                      H Mart
3    Grover Pub and Grub           Grill It Up               GoodLife Fitness
4    Grover Pub and Grub           Grill It Up  Goodfellas Wood Oven Pizza

      20th Most Common Venue
0  FLOCK Rotisserie + Greens
1                    H Mart
2                       H&M
3                     Gushi
4                     Gushi

[5 rows x 21 columns]
```

In [14]:
```
## Top 20 Venue categories in the City of Toronto
# one hot encoding
Toronto_onehot = pd.get_dummies(Toronto_venues[['Venue Category']], prefix="", prefix_sep="")
# add neighborhood column back to dataframe
Toronto_onehot['Neighborhood'] = Toronto_venues['Neighborhood']

#To group rows by neighborhood and by taking the mean of the frequency of occurrence of each category
Toronto_grouped = Toronto_onehot.groupby('Neighborhood').mean().reset_index()

def return_most_common_venues(row, num_top_venue_categories):
    row_categories = row.iloc[1:]
    row_categories_sorted = row_categories.sort_values(ascending=False)

    return row_categories_sorted.index.values[0:num_top_venue_categories]
num_top_venue_categories = 25
indicators = ['st', 'nd', 'rd']

# create columns according to number of top venues
columns = ['Neighborhood']
for ind in np.arange(num_top_venue_categories):
    try:
```

```
            columns.append('{}{} Most Common VenueCat'.format(ind+1, indicators[ind]))
        except:
            columns.append('{}th Most Common VenueCat'.format(ind+1))

    # create a new dataframe
    neighborhoods_venueCat_sorted = pd.DataFrame(columns=columns)
    neighborhoods_venueCat_sorted['Neighborhood'] = Toronto_grouped['Neighborhood']
    for ind in np.arange(Toronto_grouped.shape[0]):
        neighborhoods_venueCat_sorted.iloc[ind, 1:] = return_most_common_venues(Toronto_grouped.iloc[i
    print("The list of top 20 most common venue categories for each neighborhood.")
    neighborhoods_venueCat_sorted.head()
```

The list of top 20 most common venue categories for each neighborhood.

Out[14]:

	Neighborhood	1st Most Common VenueCat	2nd Most Common VenueCat	3rd Most Common VenueCat	4th Most Common VenueCat	5th Most Common VenueCat	6th Most Common VenueCat	7th Most Common VenueCat	8th Most Common VenueCat	9th Most Common VenueCat	...	16th Most Common VenueCat	17th Most Common VenueCat
0	Adelaide, King, Richmond	Coffee Shop	Café	Thai Restaurant	Steakhouse	American Restaurant	Hotel	Bar	Bakery	Sushi Restaurant	...	Gym	Cosmetics Shop
1	Agincourt	Chinese Restaurant	Lounge	Sandwich Place	Breakfast Spot	Skating Rink	Yoga Studio	Diner	Discount Store	Dog Run	...	Electronics Store	Empanada Restaurant
2	Agincourt North, L'Amoreaux East, Milliken, St...	Park	Playground	Coffee Shop	Yoga Studio	Doner Restaurant	Dessert Shop	Dim Sum Restaurant	Diner	Discount Store	...	Electronics Store	Empanada Restaurant
3	Albion Gardens, Beaumond Heights, Humbergate, ...	Grocery Store	Beer Store	Fried Chicken Joint	Fast Food Restaurant	Pizza Place	Sandwich Place	Pharmacy	Airport Lounge	Farmers Market	...	Drugstore	Donut Shop
4	Alderwood, Long Branch	Pizza Place	Pharmacy	Athletics & Sports	Coffee Shop	Pub	Sandwich Place	Skating Rink	Gym	Airport Service	...		

4	Drugstore	Donut Shop

	18th Most Common VenueCat	19th Most Common VenueCat
0	Breakfast Spot	Concert Hall
1	Dim Sum Restaurant	Department Store
2	Department Store	Curling Ice
3	Doner Restaurant	Dog Run
4	Doner Restaurant	Dog Run

	20th Most Common VenueCat	21th Most Common VenueCat
0	Gastropub	Gluten-free Restaurant
1	Event Space	Cosmetics Shop
2	Dance Studio	Event Space
3	Discount Store	Diner
4	Discount Store	Diner

	22th Most Common VenueCat	23th Most Common VenueCat
0	Jazz Club	Salon / Barbershop
1	Colombian Restaurant	Comfort Food Restaurant
2	Cupcake Shop	Cuban Restaurant
3	Dim Sum Restaurant	Dessert Shop
4	Dim Sum Restaurant	Dessert Shop

	24th Most Common VenueCat	25th Most Common VenueCat
0	Ice Cream Shop	Vegetarian / Vegan Restaurant
1	Comic Shop	Concert Hall
2	Creperie	Coworking Space
3	Department Store	Deli / Bodega
4	Department Store	Deli / Bodega

[5 rows x 26 columns]

The two data sets generated for top 20 venues & venue categories are used further for Apriori Analysis.

2.4.3 4.3 Apriori Analysis

In this section I will apply the Apriori algorithm to find rules that describe associations between different venues and between different venue categories in Toronto neighborhoods. The first step in Apriori analysis is to apply the data preprocessing on the dataset to convert the pandas dataFrame to the form of list of lists of venues that I want to extract rules from. To do the Apriori analysis on this, I have to use the apriori class that I import from the apyori library. The apriori class requires some parameter values to work for reasons explain as follows. The Apriori algorithm tries to extract rules for each possible combination of frequent sets of venues and describes how often venues are seen together in a neighborhood. A set is called frequent if its support meets a given absolute minimal user-specified support threshold. The task of discovering all frequent sets is quite challenging. Because of extremely slow computational speed, I need to set certain parameters for Apriori algorithm to run smoothly. Analysis of large inventories would in-

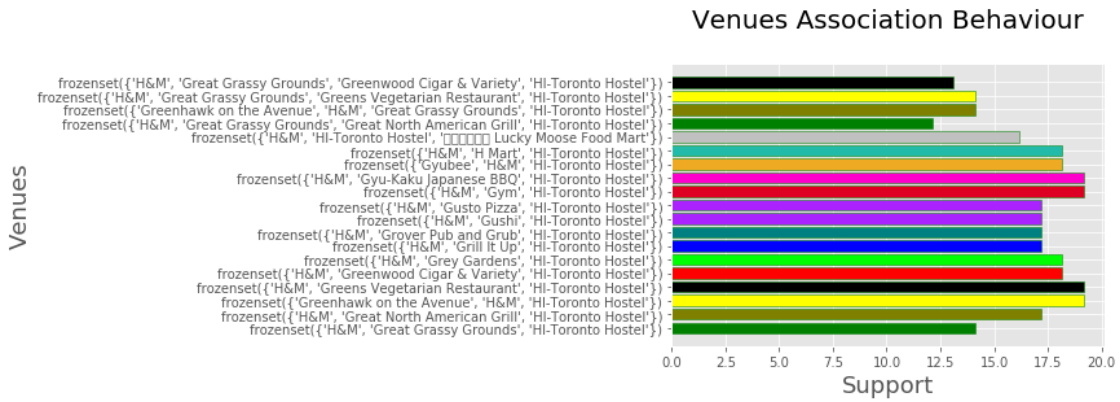
volve more item set configurations, and the support threshold might have to be lowered to detect certain associations. However, lowering the support threshold might also increase the number of false associations detected. The parameters I need to set are min_support , min_confidence, min_lift, & min_length to filter those rules that have the parameters greater than the threshold parameters specified by the user. I enter values of 0.1 (10%), 0.8(80%), 3, & 2 respectively for minimum support, confidence, lift and length so as to extract all the subsets having a higher value of support, confidence, lift & length than a minimum threshold. min_length is taken as 2 since I want at least two categories in the rules and thus to prevent neighborhood with only one venue or venue categories occurring. This helps me to omit less associated items & selects only those rules for the venues that have certain default frequency (e.g. support), have a minimum value for co-occurrence with other venues (e.g. confidence and lift). These values are mostly just arbitrarily chosen by trial and error method, see what difference it makes in the rules I get back & thus to choose the best fit parameters. Then I convert the rules found by the apriori class into a list to view the results.

Apriori Analysis on Venues

In [15]:
```
# Data Preprocessing
dataset1 = neighborhoods_venues_sorted

dataset1.columns

dataset1 = dataset1.drop(['Neighborhood'], axis = 1)

venues = []
for i in range(0, 99):
    venues.append([str(dataset1.values[i,j]) for j in range(0, 20)])

# Training Apriori on the dataset
from apyori import apriori
rules1 = apriori(venues, min_support = 0.10, min_confidence = 0.8, min_lift = 3, min_length = 2)
# min_length is to prevent neighborhood with only one venue occuring.
# Too high confidence keep unrealted venues together.

# Visualising the results
results1 = list(rules1)
print(results1[0])
df=pd.DataFrame(results1)
plt.style.use('ggplot')
figsize=(9,12)
def simple_bar_chart(support,products):
    labels=np.array(products)
    colors = ['#008000','#808000','#FFFF00','#000000','#FF0000','#00FF00','#0000FF','#008080','#aa2
    y_pos = np.arange(len(labels))
    x_pos = np.array(support)
    plt.barh(y_pos, x_pos, color=colors, align='center' ,edgecolor='green')
    plt.yticks(y_pos, labels)
    plt.ylabel('Venues',fontsize=18)
```

```
    plt.xlabel('Support',fontsize=18)
    plt.title('Venues Association Behaviour\n',fontsize=20)
    plt.show()
support=df.iloc[0:19]['support']*100
products=df.iloc[0:19]['items']
simple_bar_chart(support,products)
```

RelationRecord(items=frozenset({'H&M', 'Great Grassy Grounds', 'HI-Toronto Hostel'}), support=0.1414141414



Note that the entries in the "itemsets" column are of type frozenset, which is built-in Python type that is similar to a Python set but immutable, which makes it more efficient for certain query or comparison operations. The first rule consists of a list of items that can be found together. The first item in the list is a list itself containing three items. The first item of the list shows the categories in the rule. For instance from the first item, I can see that 'H&M', 'HI-Toronto Hostel' and 'Great Grassy Grounds' are commonly found together. The support value for the first rule is 0.1414. This number is calculated by dividing the number of categories containing H&M divided by total number of categories. The confidence level for the rule is 0.875 which shows that out of all the categories that contain 'HI-Toronto Hostel', and 'Great Grassy Grounds', 87.5% of the categories also contain 'H&M'. Finally, the lift of 3.094 says that H&M is 3.094 times more likely to be found in neighborhoods containing 'HI-Toronto Hostel', and 'Great Grassy Grounds' compared to the default likelihood of the occurrence of 'H&M' alone.

```
In [16]: results = []
        for item in results1:
            pair = item[0]
            items = [x for x in pair]
            value0 = str(items[0])
            value1 = str(items[1])
            value2 = str(item[1])[:7]
            value3 = str(item[2][0][2])[:7]
            value4= str(item[2][0][3])[:7]
            rows = (value0, value1, value2, value3, value4)
            results.append(rows)
```

```
labels = ["Venue1", "Venue2", "Support", "Confidence", "Lift"]
venue_association = pd.DataFrame.from_records(results, columns = labels)
venue_association
venue_association.drop_duplicates(["Venue1", "Venue2"], keep = "first", inplace = True)
venue_association.sort_values(by = 'Lift', ascending = False).head(1)
```

Out[16]:

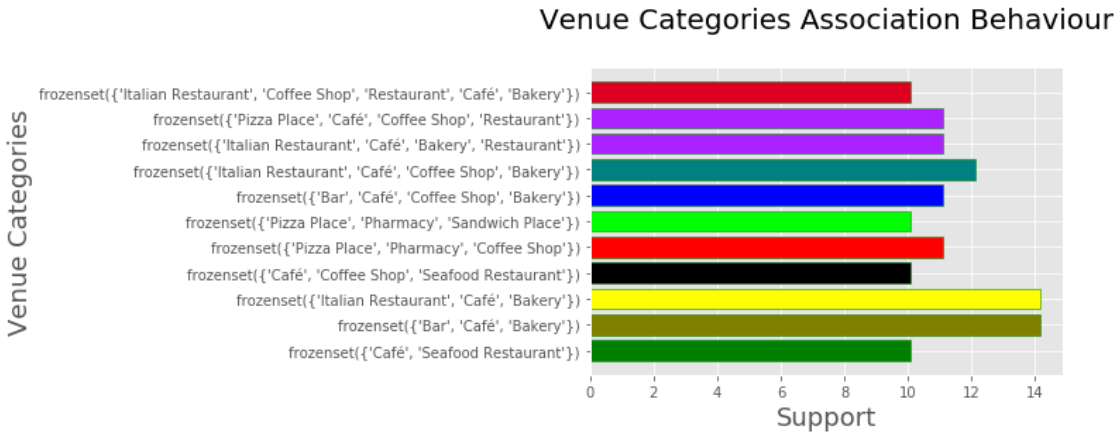
	Venue1	Venue2	Support	Confidence	Lift
514	H Mart	Grill It Up	0.17171	1.0	3.53571

Another instance, from the above rule, I can see that 'H Mart' and 'Grill It Up' are frequently found together. The support for 'H Mart' is 0.1515. The confidence level for the rule is 1.0 which means that out of all the occurrences containing 'Grill It Up', 100% of the occurrences are likely to contain 'H Mart' as well. Finally, the lift of 3.536 shows that 'H Mart' is 3.536 times more likely to be found when 'Grill It Up' is present compared to the default likelihood of the presence of 'H Mart' alone.

Apriori Analysis on Venue Categories Let me check the analysis report of Venue Categories

In [17]:
```
# Data Preprocessing
dataset2 = neighborhoods_venueCat_sorted
dataset2 = dataset2.drop(['Neighborhood'], axis = 1)
dataset2.columns

categories = []
for i in range(0, 99):
    categories.append([str(dataset2.values[i,j]) for j in range(0, 20)])

# Training Apriori on the dataset
from apyori import apriori
rules2 = apriori(categories, min_support = 0.10, min_confidence = 0.8, min_lift = 3, min_length = 2)
# min_length is to prevent neighborhood with only one venue categories occuring.
# Too high confidence keep unrealted venue categories together.

# Visualising the results
results2 = list(rules2)
df2=pd.DataFrame(results2)
plt.style.use('ggplot')
figsize=(9,12)
def simple_bar_chart(support,products):
    labels=np.array(products)
    colors = ['#008000','#808000','#FFFF00','#000000','#FF0000','#00FF00','#0000FF','#008080','#aa2
    y_pos = np.arange(len(labels))
    x_pos = np.array(support)
    plt.barh(y_pos, x_pos, color=colors, align='center' ,edgecolor='green')
    plt.yticks(y_pos, labels)
    plt.ylabel('Venue Categories',fontsize=18)
    plt.xlabel('Support',fontsize=18)
    plt.title('Venue Categories Association Behaviour\n',fontsize=20)
```

```
    plt.show()
support=df2.iloc[0:19]['support']*100
products=df2.iloc[0:19]['items']
simple_bar_chart(support,products)
```



In [18]:
```
results = []
for item in results2:
    pair = item[0]
    items = [x for x in pair]
    value0 = str(items[0])
    value1 = str(items[1])
    value2 = str(item[1])[:7]
    value3 = str(item[2][0][2])[:7]
    value4= str(item[2][0][3])[:7]
    rows = (value0, value1, value2, value3, value4)
    results.append(rows)
labels = ["Category1", "Category2", "Support", "Confidence", "Lift"]
category_association = pd.DataFrame.from_records(results, columns = labels)
category_association.drop_duplicates(["Category1", "Category2"], keep = "first", inplace = True)
category_association.sort_values(by = 'Lift', ascending = False).head(1)
```

Out[18]:

	Category1	Category2	Support	Confidence	Lift
9	Pizza Place	Café	0.11111	0.91666	3.24107

The support vector of 0.111 the first rule is calculated by dividing the number of categories containing 'Pizza Place' divided by the total number of categories. The confidence of 0.917, says that of the total categories containing 'Café' 91.7 % of categories also contain 'Pizza Place'. Finally, the lift of 3.24 shows that there are 3.24 times chances that 'Pizza Place' will be present where a 'Café' is present.

In [19]:
```
df2["ordered_statistics"].values
```

Out[19]: array([list([OrderedStatistic(items_base=frozenset({'Seafood Restaurant'}), items_add=frozenset({'Cafe
list([OrderedStatistic(items_base=frozenset({'Bar', 'Bakery'}), items_add=frozenset({'Café'}), confi
list([OrderedStatistic(items_base=frozenset({'Italian Restaurant', 'Bakery'}), items_add=frozenset({
list([OrderedStatistic(items_base=frozenset({'Coffee Shop', 'Seafood Restaurant'}), items_add=froze
list([OrderedStatistic(items_base=frozenset({'Pharmacy', 'Coffee Shop'}), items_add=frozenset({'Pi
list([OrderedStatistic(items_base=frozenset({'Pharmacy', 'Sandwich Place'}), items_add=frozenset(
list([OrderedStatistic(items_base=frozenset({'Bar', 'Coffee Shop', 'Bakery'}), items_add=frozenset({
list([OrderedStatistic(items_base=frozenset({'Italian Restaurant', 'Coffee Shop', 'Bakery'}), items_a
list([OrderedStatistic(items_base=frozenset({'Italian Restaurant', 'Bakery', 'Restaurant'}), items_ad
list([OrderedStatistic(items_base=frozenset({'Pizza Place', 'Café', 'Coffee Shop'}), items_add=frozen
list([OrderedStatistic(items_base=frozenset({'Italian Restaurant', 'Coffee Shop', 'Bakery', 'Restauran
dtype=object)

The first rule here states that there is 100% chance a 'Cafe' category will be seen where a 'Seafood Restaurant' is present. Lift 0f 3.09 gives indication that there is 3.09 times probability of finding a 'Cafe' venue category wherever Seafood restaurant is present. And checking the last rule: With a confidence of 1 means, along with 'Bakery', 'Italian Restaurant', 'Restaurant'& 'Coffee Shop' there is 100 % chance that the category 'Café' will also be seen. And there is a lift of 3.09 indicating that there are 3.09 times chances that the 'Café' will be present in a neighborhood if a neighborhood contains all of the above mentioned categories.

2.5 5. Conclusion

Although widely used in market basket analysis and understanding the customer buying behavior, here I have successfully implemented Apriori Association rule Algorithm to study the association behavior of the venues and venue categories of Toronto Neighborhoods. Apriori Association rule mining algorithms are found to be very useful for finding simple associations between the data items of the datasets venues & venue categories of Toronto neighborhoods. It is easy to implement and the results are self-explanatory.