Association Rule Analysis of Venues & Venue Categories of Toronto Neighborhoods

Table of Contents

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. 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(Y→X) = (Confidence (Y→X))/(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.

3. Data Collection

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.
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.

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.

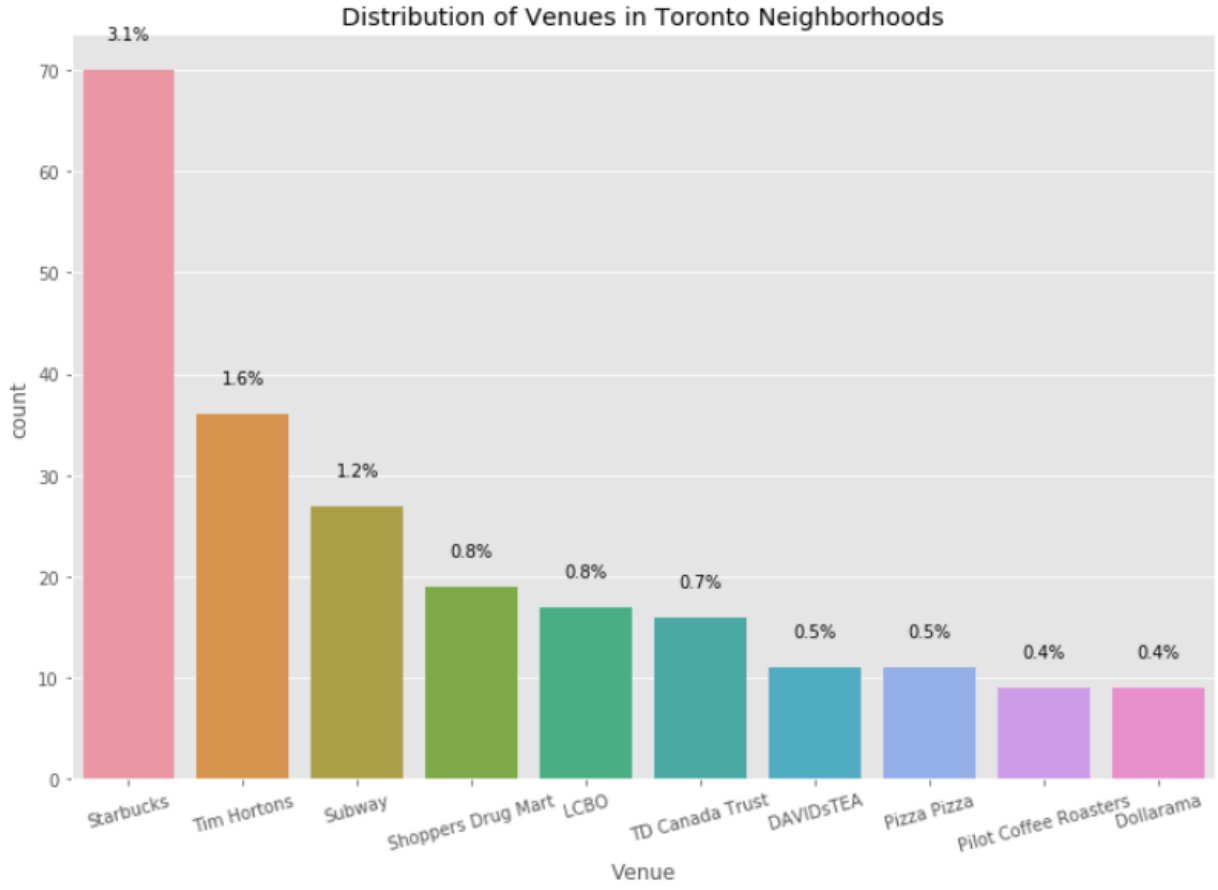
3.3 Geospatial Data

From the csv file, a dataFrame containing the geographical coordinates corresponding to each postal code is created.
Combining all three sources of data after applying getNearbyVenues function which I have already defined before, a new dataFrame, Toronto_venues is obtained.

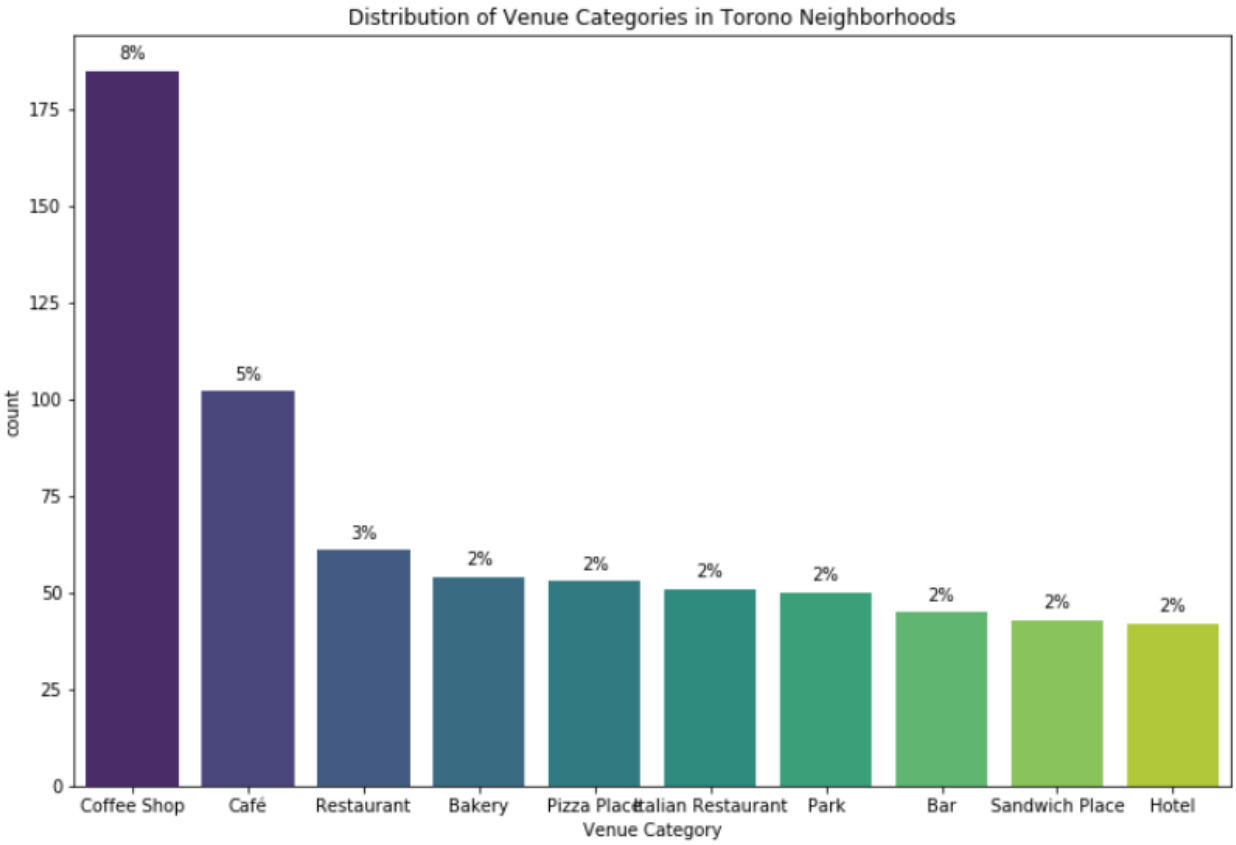
4. Results & Discussion

4.1 Exploratory Data Analysis (EDA)

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



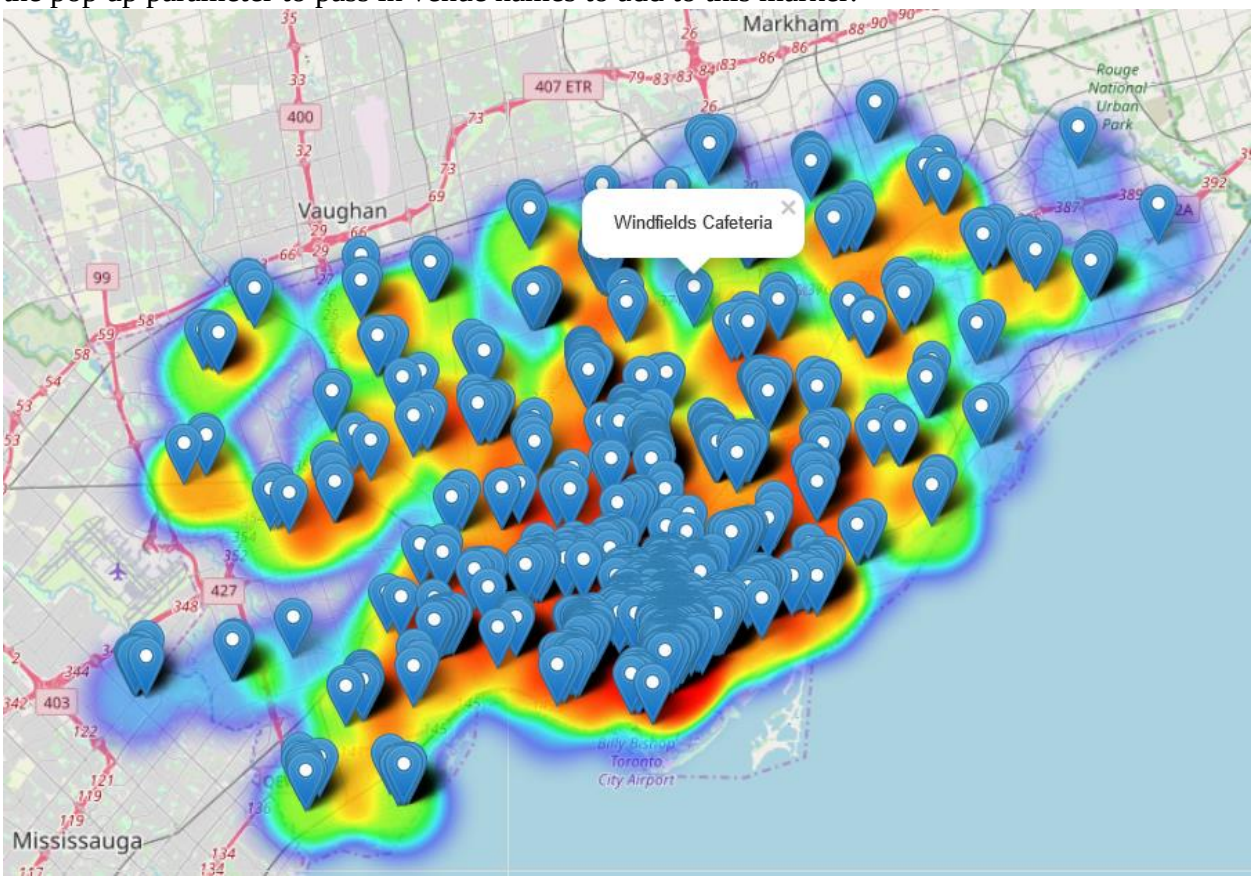
The seaborn count plot shows that Starbucks, followed by Tim Hortons & Subway are the 3 top most frequent Venues in the city of Toronto. Starbucks accounts for the 3.1% of all the venues in the city.



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

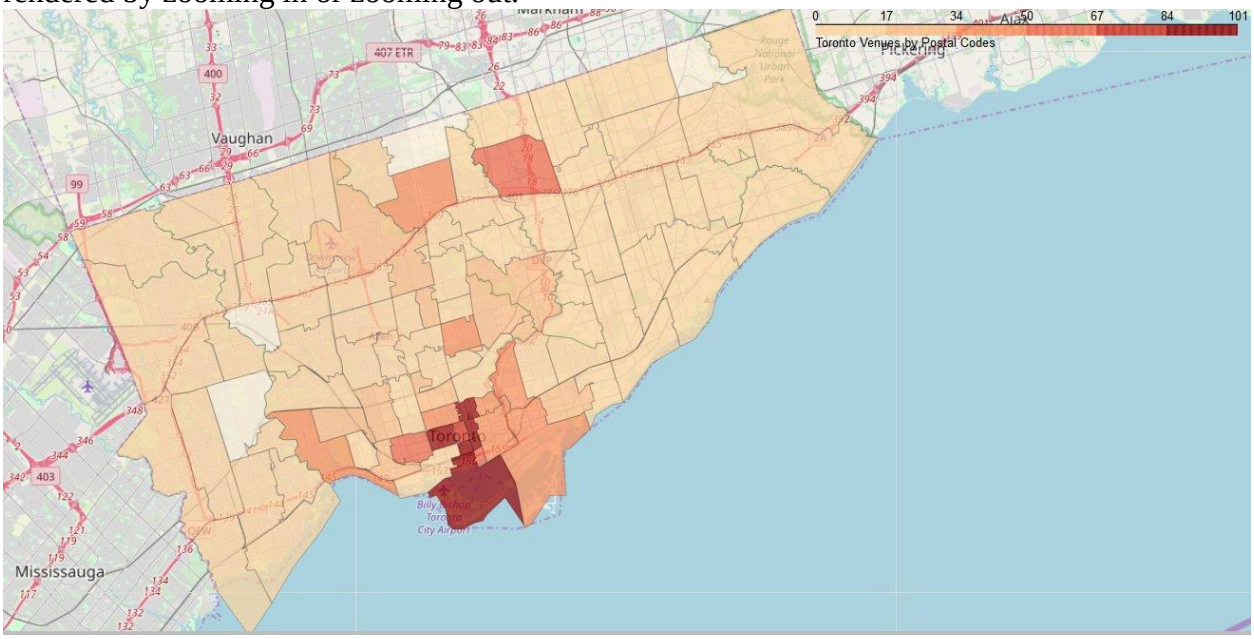
A Heat map of venues(Figure 3) is generated using latitude and longitude values obtained from the merged data of Foursquare data. Markers are added to specify the location of the venues. To do that, a for loop is used to iterate through each row and then adding them to the Folium map. The columns to 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.



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.



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

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.

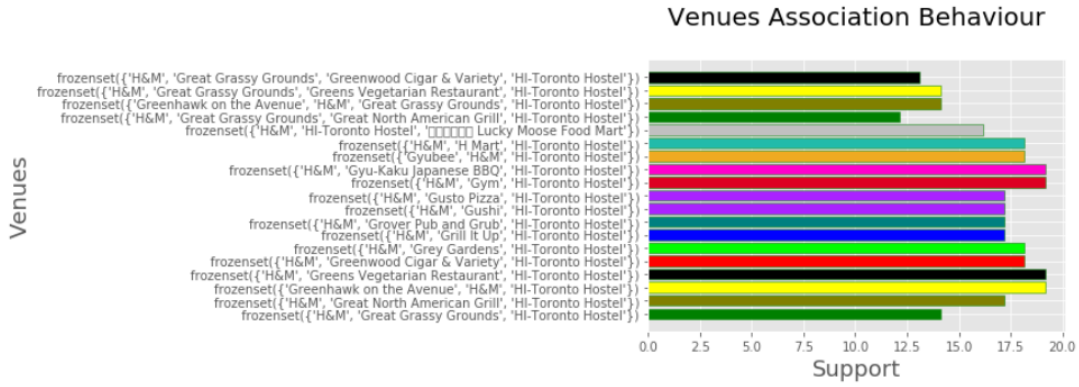
The two data sets generated for top 20 venues & venue categories are used further for Apriori Analysis.

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 involve 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

```
RelationRecord(items=frozenset({'H&M', 'Great Grassy Grounds', 'HI-Toronto Hostel'}), support=0.1414141414141414, ord
ered_statistics=[OrderedStatistic(items_base=frozenset({'Great Grassy Grounds', 'HI-Toronto Hostel'}), items_add=froz
enset({'H&M'}), confidence=0.8749999999999999, lift=3.0937499999999996)])
```



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

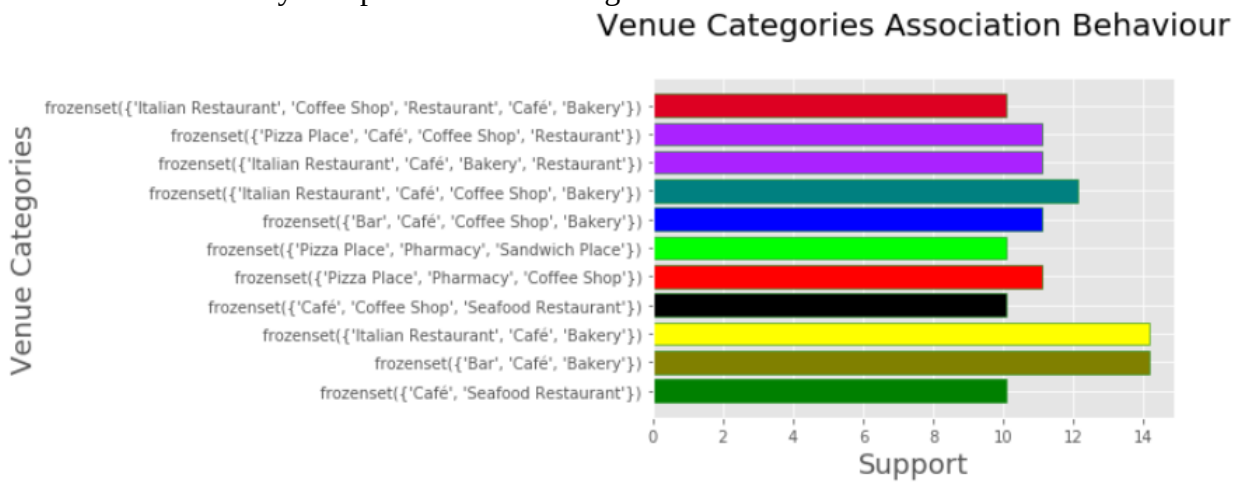
0.1414. This number is calculated by dividing the number of categories containing H&M divided by total number of categories. The confidence level for the rule is 0.875 which shows that out of all the categories that contain 'HI-Toronto Hostel', and 'Great Grassy Grounds', 87.5% of the categories also 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.

	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



	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.

```
array([list([OrderedStatistic(items_base=frozenset({'Seafood Restaurant'}), items_add=frozenset({'Café'}), confidence
=1.0, lift=3.0937499999999996)]),
       list([OrderedStatistic(items_base=frozenset({'Bakery', 'Bar'}), items_add=frozenset({'Café'}), confidence=1.0,
lift=3.0937499999999996), OrderedStatistic(items_base=frozenset({'Café', 'Bar'}), items_add=frozenset({'Bakery'}), co
nfidence=0.8749999999999999, lift=3.3317307692307683)]),
       list([OrderedStatistic(items_base=frozenset({'Bakery', 'Italian Restaurant'}), items_add=frozenset({'Café'}),
confidence=1.0, lift=3.0937499999999996)]),
       list([OrderedStatistic(items_base=frozenset({'Seafood Restaurant', 'Coffee Shop'}), items_add=frozenset({'Café
'}), confidence=1.0, lift=3.0937499999999996)]),
       list([OrderedStatistic(items_base=frozenset({'Pharmacy', 'Coffee Shop'}), items_add=frozenset({'Pizza Place'})
, confidence=0.9166666666666666, lift=3.025)]),
       list([OrderedStatistic(items_base=frozenset({'Pharmacy', 'Sandwich Place'}), items_add=frozenset({'Pizza Place
'}), confidence=0.9090909090909092, lift=3.0)]),
       list([OrderedStatistic(items_base=frozenset({'Bakery', 'Coffee Shop', 'Bar'}), items_add=frozenset({'Café'}),
confidence=1.0, lift=3.0937499999999996), OrderedStatistic(items_base=frozenset({'Café', 'Coffee Shop', 'Bar'}), item
s_add=frozenset({'Bakery'}), confidence=0.846153846153846, lift=3.2218934911242596)]),
       list([OrderedStatistic(items_base=frozenset({'Bakery', 'Italian Restaurant', 'Coffee Shop'}), items_add=frozen
set({'Café'}), confidence=1.0, lift=3.0937499999999996)]),
       list([OrderedStatistic(items_base=frozenset({'Bakery', 'Italian Restaurant', 'Restaurant'}), items_add=frozens
et({'Café'}), confidence=1.0, lift=3.0937499999999996)]),
       list([OrderedStatistic(items_base=frozenset({'Café', 'Coffee Shop', 'Pizza Place'}), items_add=frozenset({'Res
taurant'}), confidence=0.9166666666666666, lift=3.2410714285714284)]),
       list([OrderedStatistic(items_base=frozenset({'Bakery', 'Italian Restaurant', 'Restaurant', 'Coffee Shop'}), it
ems_add=frozenset({'Café'}), confidence=1.0, lift=3.0937499999999996)])],
      dtype=object)
```

The first rule here in ordered statistics 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.

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.