1. INTRODUCTION

Toronto is the provincial city of Ontario and the most populous city in Canada. It had a population of 2.731.571 people in 2016. This city has too many places to visit, like the Royal Ontario Museum, the Art Gallery of Ontario, the Gardiner museum of Ceramic Art and Ontario Science Centre (Wikipedia, 2019). Toronto in an international centre for business and finance. It has a high concentration of banks and brokerage firms on Bay street. The city is an important centre for the media. Some of the corporations are the Bell media, Rogers communication and Torstar (Wikipedia - Toronto, 2019).

2. Objective

Marcos is a cooker that lives in the borough of Etobicoke and at the neighborhood of Cloverdale in Toronto. He comes from a family of famous cookers of Canada. He wants to build his first restaurant at Canada but he wants to build in a place that has the same characteristics as his neighborhood. He could build his first restaurant at this neighborhood, but he wants to know if there are other places with the same characteristics. He looked for a statistician to see if there are other places in Toronto with the same characteristics as his neighborhood. So, let's get the job done and find an answer for Marcos.

3. Dataset

The dataset used at this project come from different sources. One of the datasets is from wikipedia dataset that contains the boroughs and neighborhoods from the city of Toronto (link: https://en.wikipedia.org/wiki/List_of_postal_codes_of_Canada:_M (https://en.wikipedia.org/wiki/List_of_postal_codes_of_Canada:_M)), and the other is from foursquare. It will be used to generate the geolocalization for the neighborhoods and its respective venues.

The datasets are gonna be used to generate a cluster model to find the neighborhoods that are similar to Cloverdale, according to the objective of this project.

4. Methodology

In this project we gonna use google colab jupyter notebook to generate the data and models. Data are gonna be purchased according to the section 3, and we gonna use the Cluster analysis to find the neighborhoods that are similar to Cloverdale.

Cluster analysis or Clustering is a task of grouping different objects in such a way that objects in the same group (or cluster) are more similar (acoording to criterias) to each other than those on other groups. (Wikipedia - Cluster Analysis, 2019)

5. Results and Discussion

First, we gonna import the libraries to use in this notebook.

In [0]:
```
#Importing the Libraries
import pandas as pd
import requests
from bs4 import BeautifulSoup
from geopy.geocoders import Nominatim
import folium
import requests
import json
from pandas.io.json import json_normalize
import numpy as np
from sklearn.cluster import KMeans
import matplotlib.cm as cm
import matplotlib.colors as colors
```

After we import the libraries, we need to purchase the data. We used the package Beautifulsoup to this task. The site used was wikipedia containing data about Toronto, its Boroughs and Neighborhoods.

In [0]:
```
#Url that contains all the boroughs from Toronto.
url='https://en.wikipedia.org/wiki/List_of_postal_codes_of_Canada:_M'
```

In [0]:
```
#Requesting the text from the wikipedia site
html = requests.get(url).text
soup = BeautifulSoup(html, 'xml')
```

The tables that we gonna use in this project are contained in the tag **<'tr'>** and the text in the tag **<'td'>**. We used the Beautifulsoup package again to extract all the informations necessary to generate the table.

In [0]:
```
#Searching for table class wikitable sortble - This class has the table that we need
Table = soup.find('table',{'class':'wikitable sortable'})
```

In [0]:
```
#Searching for <tr>
table_rows = Table.find_all('tr')
```

In [0]:
```
#Getting the data that we need to construct the dataframe
data = []
for r in table_rows:
  data.append([t.text.strip() for t in r.find_all('td')])
```

The pandas package was used to generate the table. There were some Boroughs and Neighborhoods that were not assigned any value, so they were excluded.

In [0]:
```
#Transforming the data into a dataframe
df = pd.DataFrame(data, columns = ["Postal Code","Borough","Neighborhood"])
df.loc[(df.Borough != 'Not assigned') & (df.Neighborhood != 'Not assigned')][1:].head()
```

Out[0]:

	Postal Code	Borough	Neighborhood
3	M3A	North York	Parkwoods
4	M4A	North York	Victoria Village
5	M5A	Downtown Toronto	Harbourfront
6	M5A	Downtown Toronto	Regent Park
7	M6A	North York	Lawrence Heights

The table was extracted, but, there is no geolocalization of the neighborhoods. We used the url down to get these data and merged the two tables using Postal Code as the linking between these two tables.

In [0]:
```
url2 = 'https://cocl.us/Geospatial_data'
geosptcoord = pd.read_csv(url2)
```

In [0]:
```
#Merging the two tables
df2 = pd.merge(df,geosptcoord, on = 'Postal Code')
df2.head()
```

Out[0]:

	Postal Code	Borough	Neighborhood	Latitude	Longitude
0	M3A	North York	Parkwoods	43.753259	-79.329656
1	M4A	North York	Victoria Village	43.725882	-79.315572
2	M5A	Downtown Toronto	Harbourfront	43.654260	-79.360636
3	M5A	Downtown Toronto	Regent Park	43.654260	-79.360636
4	M6A	North York	Lawrence Heights	43.718518	-79.464763

A map of Toronto was generated with its boroughs and neighborhoods using the package folium.

In [0]:
```
#Geolocalization of Toronto city
address = 'Toronto'

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

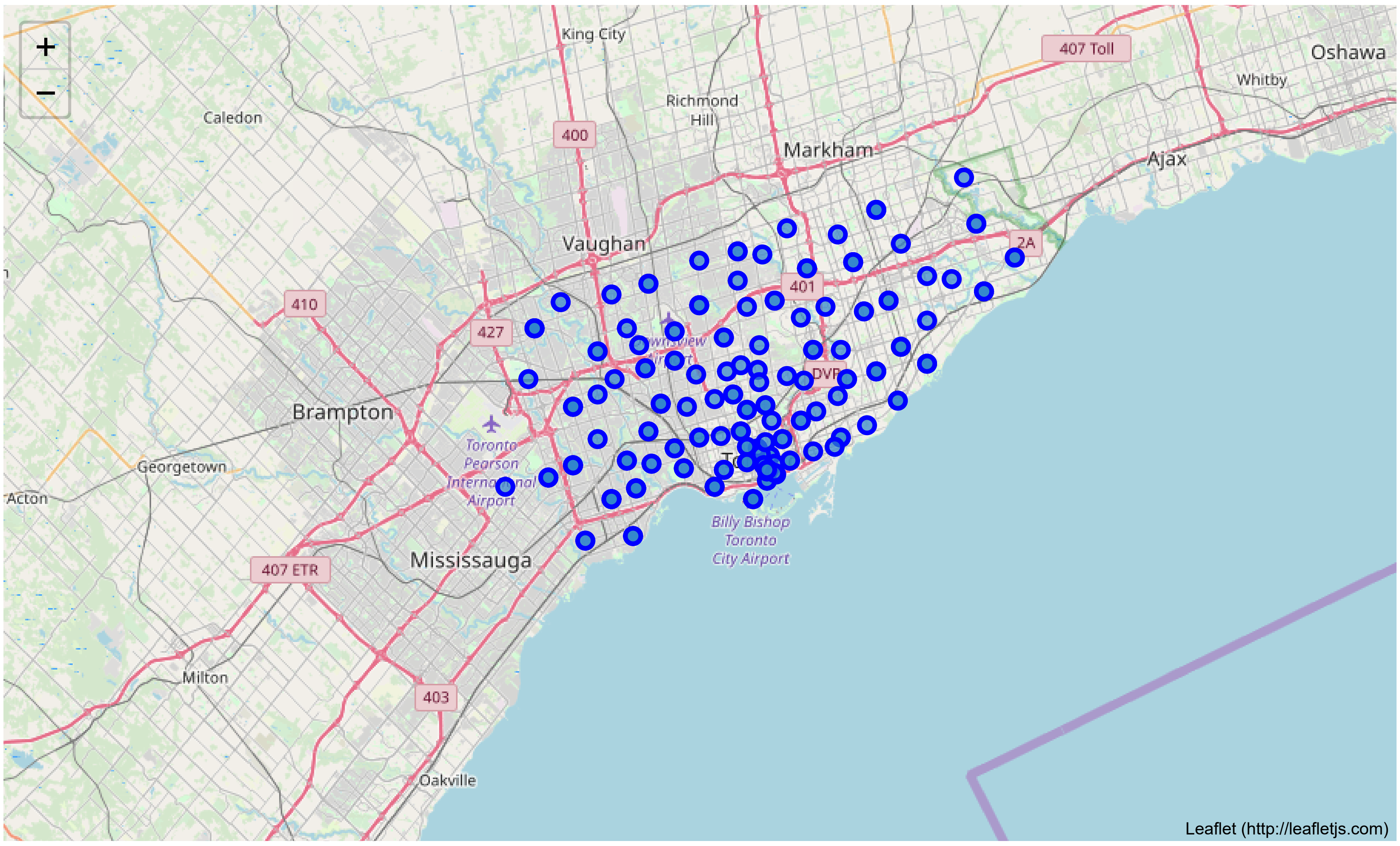
```
The geograpical coordinate of Toronto are 43.653963, -79.387207.
```

In [0]:
```
# create map of Toronto using latitude and longitude values
map_toronto = folium.Map(location=[latitude, longitude], zoom_start=10)

# add markers to map
for lat, lng, borough, neighborhood in zip(df2['Latitude'], df2['Longitude'], df2['Borough'], df2['Neighborhood']):
    label = '{}, {}'.format(neighborhood, borough)
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius=5,
        popup=label,
        color='blue',
        fill=True,
        fill_color='#3186cc',
        fill_opacity=0.7,
        parse_html=False).add_to(map_toronto)

map_toronto
```

Out[0]:



The table was generated but we need to know the venues that each neighborhood has. This information will be used to cluster the neighborhoods and see which ones are similar to Marcos neighborhood. The venues information was generated for just one neighborhood, first, and after, it was generated for all neighborhoods using the function *getNearbyVenues* that is defined bellow.

In [0]:
```
# type your answer here

LIMIT = 100 # limit of number of venues returned by Foursquare API
radius = 500 # define radius

url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={},{}&radius={}&limit={}'.f
ormat(
    CLIENT_ID,
    CLIENT_SECRET,
    VERSION,
    neighborhood_latitude,
    neighborhood_longitude,
    radius,
    LIMIT)
url # display URL
```

Out[0]:
```
'https://api.foursquare.com/v2/venues/explore?&client_id=1QJX4QNAPKG3LXG5OXBEEZYEZVOPL1B0HP03VYM3SSLVOXF5&client_secr
et=DD4N1LP4EJXT24PM3IJKQBJZTIETLNFU0KSRCMPVQVT2AW5F&v=20190906&ll=43.7532586,-79.3296565&radius=500&limit=100'
```

In [0]:
```
results = requests.get(url).json()
```

In [0]:
```python
# 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 [0]:
```python
venues = results['response']['groups'][0]['items']

nearby_venues = json_normalize(venues) # flatten JSON

# filter columns
filtered_columns = ['venue.name', 'venue.categories', 'venue.location.lat', 'venue.location.lng']
nearby_venues =nearby_venues.loc[:, filtered_columns]

# filter the category for each row
nearby_venues['venue.categories'] = nearby_venues.apply(get_category_type, axis=1)

# clean columns
nearby_venues.columns = [col.split(".")[-1] for col in nearby_venues.columns]

nearby_venues.head()
```

Out[0]:

	name	categories	lat	lng
0	Brookbanks Park	Park	43.751976	-79.332140
1	KFC	Fast Food Restaurant	43.754387	-79.333021
2	Variety Store	Food & Drink Shop	43.751974	-79.333114

In [0]:
```python
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={},{}&radius={}&lim
it={}'.format(
            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']
```

In [0]:
```python
# Getting the venues for all the neighborhoods

toronto_venues = getNearbyVenues(names=df2['Neighborhood'],
                                   latitudes=df2['Latitude'],
                                   longitudes=df2['Longitude']
                                  )
```

In [0]:
```
toronto_venues.head()
```

Out[0]:

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

Bellow, we see that cofee shop is the venue with higher frequency of appearance. Café is the same as cofee shop. Just after restaurants, bakery and bar. Marcos wants to build one restaurant, and there are many competitors as we saw.

In [0]:
```
#The venues with the higher frequencies of Toronto city
toronto_venues['Venue Category'].value_counts().head(20)
```

Out[0]:
```
Coffee Shop             347
Café                    199
Pizza Place             122
Restaurant              120
Bakery                  118
Italian Restaurant       94
Bar                      92
Fast Food Restaurant     87
Park                     85
Sandwich Place           80
Hotel                    75
Clothing Store           64
Japanese Restaurant      63
American Restaurant      63
Grocery Store            58
Gym                      53
Pharmacy                 52
Burger Joint             51
Breakfast Spot           49
Pub                      48
Name: Venue Category, dtype: int64
```

To cluster the neighborhoods was generated a table containing all the venues categories for each neighborhood. After, the venues were grouped to each unique neighborhood.

In [0]:
```
# 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']

# move neighborhood column to the first column
fixed_columns = [toronto_onehot.columns[-1]] + list(toronto_onehot.columns[:-1])
toronto_onehot = toronto_onehot[fixed_columns]

toronto_onehot.head()
```

Out[0]:

	Yoga Studio	Accessories Store	Adult Boutique	Afghan Restaurant	Airport	Airport Food Court	Airport Gate	Airport Lounge	Airport Service	Airport Terminal	American Restaurant	Antique Shop	Aquarium	Art Gallery	Muse
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

In [0]:
```
toronto_onehot.shape
```

Out[0]:
```
(4438, 280)
```

In [0]:
```
toronto_grouped = toronto_onehot.groupby('Neighborhood').mean().reset_index()
toronto_grouped.head()
```

Out[0]:

	Neighborhood	Yoga Studio	Accessories Store	Adult Boutique	Afghan Restaurant	Airport	Airport Food Court	Airport Gate	Airport Lounge	Airport Service	Airport Terminal	American Restaurant	Antique Shop	Aquarium
0	Adelaide	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.04	0.0	0.0
1	Agincourt	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00	0.0	0.0
2	Agincourt North	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00	0.0	0.0
3	Albion Gardens	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00	0.0	0.0
4	Alderwood	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00	0.0	0.0

In [0]:
```
toronto_grouped.shape
```

Out[0]: (205, 280)

Defining a function *return_most_common_venues* to return the most common venues for each neighborhood.

In [0]:
```
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]
```

In [0]:
```
num_top_venues = 10

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_grouped['Neighborhood']

for ind in np.arange(toronto_grouped.shape[0]):
    neighborhoods_venues_sorted.iloc[ind, 1:] = return_most_common_venues(toronto_grouped.iloc[ind, :], num_top_venues
)

neighborhoods_venues_sorted.head()
```

Out[0]:

	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	10th Most Common Venue
0	Adelaide	Coffee Shop	Café	Bar	American Restaurant	Thai Restaurant	Steakhouse	Hotel	Cosmetics Shop	Burger Joint	Bakery
1	Agincourt	Chinese Restaurant	Lounge	Sandwich Place	Breakfast Spot	Women's Store	Discount Store	Dog Run	Doner Restaurant	Donut Shop	Drugstore
2	Agincourt North	Park	Asian Restaurant	Playground	Women's Store	Donut Shop	Dim Sum Restaurant	Diner	Discount Store	Dog Run	Doner Restaurant
3	Albion Gardens	Grocery Store	Liquor Store	Sandwich Place	Fried Chicken Joint	Video Store	Coffee Shop	Pharmacy	Pizza Place	Beer Store	Fast Food Restaurant
4	Alderwood	Pizza Place	Coffee Shop	Gym	Skating Rink	Pharmacy	Pub	Dance Studio	Pool	Sandwich Place	Women's Store

After all these steps with a table appropriate to cluster analysis, we generated five diferent cluster.

In [0]:
```
# set number of clusters
kclusters = 5

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

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

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

Out[0]:
```
array([0, 0, 4, 0, 0, 0, 0, 0, 0, 0], dtype=int32)
```

In [0]:
```
# add clustering labels
neighborhoods_venues_sorted.insert(0, 'Cluster Labels', kmeans.labels_)

toronto_merged = df2

# merge toronto_grouped with toronto_data to add latitude/longitude for each neighborhood
toronto_merged = toronto_merged.join(neighborhoods_venues_sorted.set_index('Neighborhood'), on='Neighborhood')

toronto_merged.head() # check the last columns!
```

Out[0]:

	Postal Code	Borough	Neighborhood	Latitude	Longitude	Cluster Labels	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 Mo Comm Ven
0	M3A	North York	Parkwoods	43.753259	-79.329656	4.0	Fast Food Restaurant	Park	Food & Drink Shop	Event Space	Ethiopian Restaurant	Empanada Restaurant	Electroni Sto
1	M4A	North York	Victoria Village	43.725882	-79.315572	0.0	Coffee Shop	Hockey Arena	Intersection	Portuguese Restaurant	Women's Store	Donut Shop	Dim Su Restaura
2	M5A	Downtown Toronto	Harbourfront	43.654260	-79.360636	0.0	Coffee Shop	Park	Pub	Bakery	Theater	Breakfast Spot	Restaura
3	M5A	Downtown Toronto	Regent Park	43.654260	-79.360636	0.0	Coffee Shop	Park	Pub	Bakery	Theater	Breakfast Spot	Restaura
4	M6A	North York	Lawrence Heights	43.718518	-79.464763	0.0	Clothing Store	Furniture / Home Store	Women's Store	Event Space	Vietnamese Restaurant	Boutique	Coff Sh

The cluster are in the map bellow. Cloverdale belongs to cluster 2. There aren't many of them, so, it's not good news for Marcos. This restricts his options.

In [0]:
```
toronto_merged = toronto_merged.dropna()
toronto_merged['Cluster Labels'] = toronto_merged['Cluster Labels'].astype('int64')
```

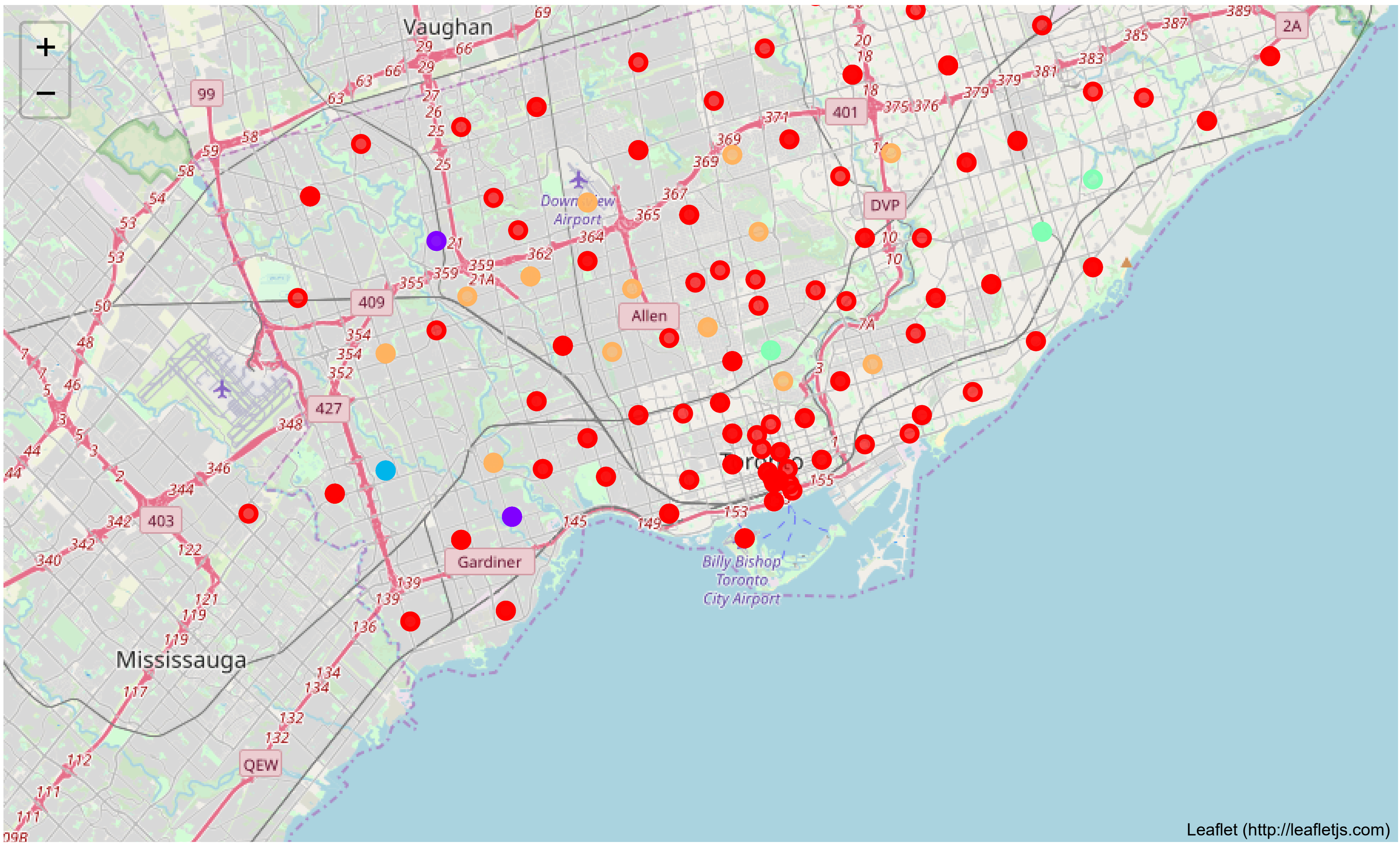
In [0]:
```python
# create map
map_clusters = folium.Map(location=[latitude, longitude], zoom_start=11)

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

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

map_clusters
```

Out[0]:



In [0]:
```python
toronto_merged['Cluster Labels'].value_counts()
```

Out[0]:
```
0    160
4     26
1     10
3      6
2      5
Name: Cluster Labels, dtype: int64
```

In [0]:
```
#Cluster 2
toronto_merged.loc[toronto_merged['Cluster Labels'] == 2, toronto_merged.columns[[1,2] + list(range(4, toronto_merged.shape[1]))]]
```

Out[0]:

	Borough	Neighborhood	Longitude	Cluster Labels	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
16	Etobicoke	Cloverdale	-79.554724	2	Bank	Women's Store	Drugstore	Diner	Discount Store	Dog Run	Doner Restaurant	Donut Shop	Dumpling Restaurant
17	Etobicoke	Islington	-79.554724	2	Bank	Women's Store	Drugstore	Diner	Discount Store	Dog Run	Doner Restaurant	Donut Shop	Dumpling Restaurant
18	Etobicoke	Martin Grove	-79.554724	2	Bank	Women's Store	Drugstore	Diner	Discount Store	Dog Run	Doner Restaurant	Donut Shop	Dumpling Restaurant
19	Etobicoke	Princess Gardens	-79.554724	2	Bank	Women's Store	Drugstore	Diner	Discount Store	Dog Run	Doner Restaurant	Donut Shop	Dumpling Restaurant
20	Etobicoke	West Deane Park	-79.554724	2	Bank	Women's Store	Drugstore	Diner	Discount Store	Dog Run	Doner Restaurant	Donut Shop	Dumpling Restaurant

As we see, Cluster 2 belongs to the borough of Etobicoke with all its neighborhoods. So, is better for Marcos to open his restaurant at his own borough. He could still open in other neighborhood.

6. Conclusion

In this project we used many different tools to solve Marcos problem. We need to purchase the data from two different sites, wikipedia and foursquare and use the cluster analysis to find the neighborhoods that are similar to Cloverdale. It was no surprise that the neighborhoods similar to Cloverdale are the ones in the same borough as Cloverdale. So, if Marcos want a neighborhood with the same characteristics as the one that he lives, it is preferable to install a restaurant at Etobicoke.

7. References

1. Wikipedia - Toronto, 2019. link: https://en.wikipedia.org/wiki/Toronto (https://en.wikipedia.org/wiki/Toronto)
2. Wikipedia - Cluster Analysis, 2019. link: https://en.wikipedia.org/wiki/Cluster_analysis (https://en.wikipedia.org/wiki/Cluster_analysis)