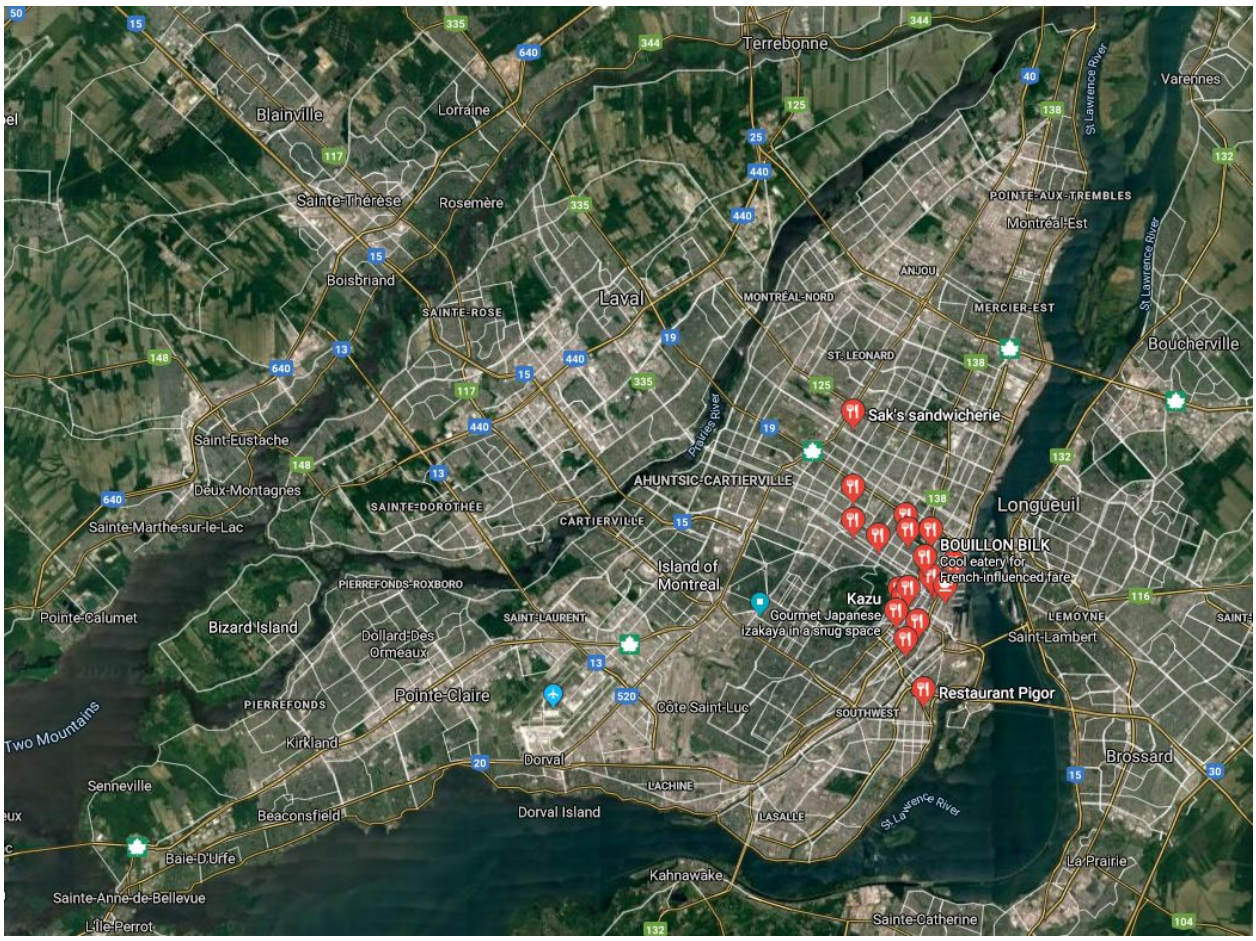


The Battle of Neighborhoods

01.23.2020

Your Name

Oliver Sobremonte Magno

Montreal, QC, Canada

Introduction: Business Problem

This is my Data Science Capstone project, I am setting the location to Montreal, Quebec, since this is where I am currently residing. I am creating a concept of building a small food business here and want to find where the best strategic place for Asian Restaurant. So if a businessman wants to ask me where is the best place, I can easily point to him where and why.

Problem

With so many Asian food businesses here in Montreal, it is hard to point out if the planned business will be profitable and not a liability. Using Data Science, we can predict the best place and situation, most of all, the chance of having a profit.

Data

- Montreal Postcode: https://en.wikipedia.org/wiki/List_of_postal_codes_of_Canada:_H
- Geolocation using googlemaps API

Install Necessary Modules

Since I will be using Python 3.6 and Jupyter Notebook, I need to install some necessay libraries to scrap all datas that I needed.

- Beautifulsoup4: https://pypi.org/project/beautifulsoup442/
- Lxml: https://pypi.org/project/lxml/
- Googlemaps: https://pypi.org/project/googlemaps/
- Folium: https://pypi.org/project/folium/

Necessary Modules for this Project

- Pandas
- Numpy
- Scikit-learn
- Matplotlib

Get Postcode, Borough, Neighborhood, Longitude and Latitude

We create a dictionary out of this process

We will build the factors that will affect our decisions:

- number of existing restaurants in Montreal
- number of and distance to any Asian Restaurant, or if none, we will investigate if it is the right place
- distance of each neighborhood

In this project we will:

- Scrape the data from 'https://en.wikipedia.org/wiki/List_of_postal_codes_of_Canada:_H'
- Then we will get the borough, neighborhood and postalcodes.
- Then we will use googlemap to get the longitude and latitude of each borough.

The code below is the one I use to put the Latitude and Longitude of each Neighborhood

```
key = "your_googlemap_key"

from collections import defaultdict

source = requests.get('https://en.wikipedia.org/wiki/List_of_postal_codes_of_Canada:_H').text
soup = BeautifulSoup(source, 'html.parser')
gmaps = googlemaps.Client(key = key)

postal_codes_dict = defaultdict(dict) # initialize an empty dictionary to save the data in
for table_cell in soup.find_all('td'):
    try:
        postal_code = table_cell.p.b.text # get the postal code
        postal_code_investigate = table_cell.span.text
        neighborhoods_data = table_cell.span.text # get the rest of the data in the cell
        borough = neighborhoods_data.split('(')[0] # get the borough in the cell

        # if the cell is not assigned then ignore it
        if neighborhoods_data == 'Not assigned':
            neighborhoods = []
        # else process the data and add it to the dictionary
        else:
            postal_codes_dict[postal_code] = {}

            try:
                neighborhoods = neighborhoods_data.split('(')[1]

                # remove parantheses from neighborhoods string
```

```
                neighborhoods = neighborhoods.replace('(', ' ')
                neighborhoods = neighborhoods.replace(')', ' ')

                neighborhoods_names = neighborhoods.split('/')
                neighborhoods_clean = ', '.join([name.strip() for name in neighborhoods_names])
                neighborhoods_clean = neighborhoods_clean.strip('\n')
            except:
                borough = borough.strip('\n')
                neighborhoods_clean = borough

            #googlemaps for longitude and latitude of borough
            geocode_result = gmaps.geocode('{}, QC'.format(neighborhoods_clean))
            latlong = geocode_result[0]['geometry']['location']
            latitude = latlong["lat"]
            longitude = latlong['lng']

            # add borough and neighborhood to dictionary
            postal_codes_dict[postal_code]['borough'] = borough
            postal_codes_dict[postal_code]['neighborhoods'] = neighborhoods_clean
            postal_codes_dict[postal_code]['latitude'] = latitude
            postal_codes_dict[postal_code]['longitude'] = longitude

    except:
        pass
```

Create the Data Frame

Below code is what will create the base dataframe that we need.

```
# create an empty dataframe
columns = ['PostalCode', 'Borough', 'Neighborhood', 'Latitude', 'Longitude']
montreal_data = pd.DataFrame(columns=columns)

# populate dataframe with data from dictionary
for key, val in postal_codes_dict.items():

    try:
        borough = val['borough']
        neighborhood = val['neighborhoods']
        latitude = val['latitude']
        longitude = val['longitude']


    except KeyError:

        borough = ''
        neighborhood = ''
        latitude = ''
        longitude = ''

    montreal_data = montreal_data.append({"PostalCode": key,
                                          "Borough": borough,
```

```
                                    "Neighborhood": neighborhood,
                                     "Latitude": latitude,
                                     "Longitude": longitude},
                                    ignore_index=True)
```

Then we save the data frame as csv, the reason for this is to avoid going back and forth creating the data frame.

```
montreal_data.to_csv(r'montrealData.csv')

#Preview the data frame

df_montreal = pd.read_csv('montrealData.csv')
df_montreal.head(10)
```

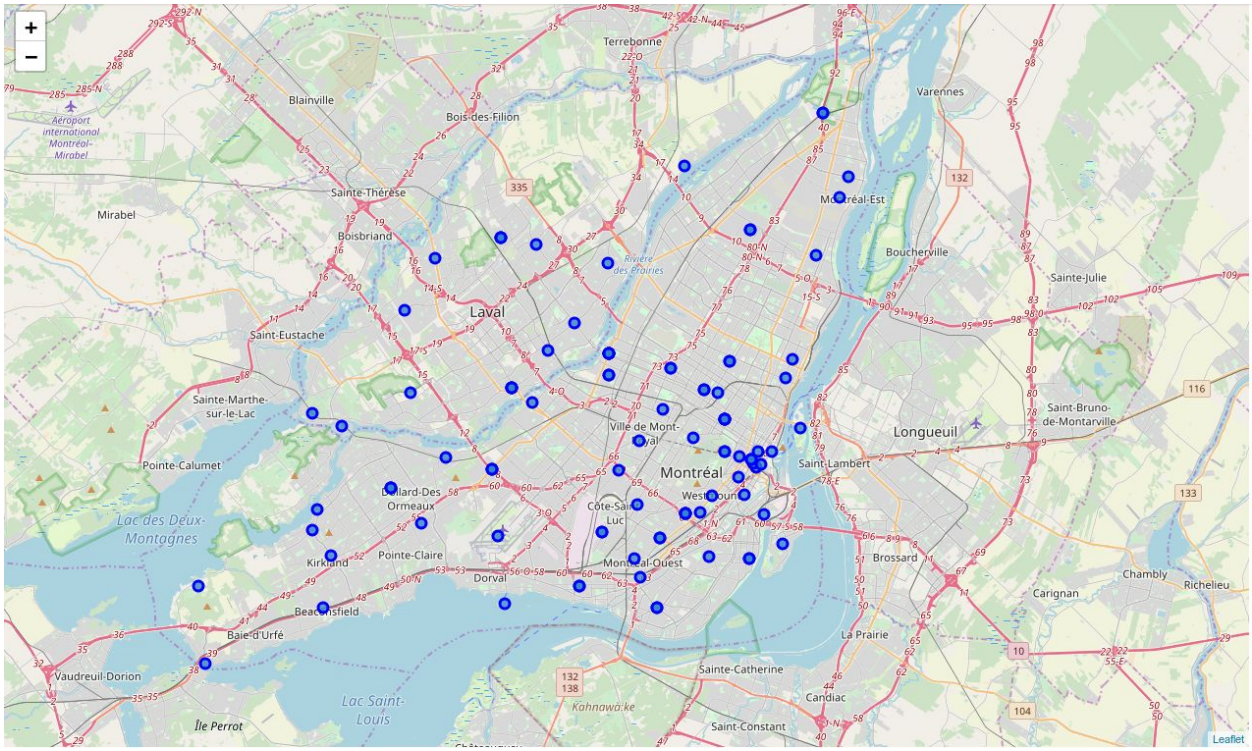
And here is a preview of the table:

	Unnamed: 0	PostalCode	Borough	Neighborhood	Latitude	Longitude
0	0	H1A	Pointe-aux-Trembles	Pointe-aux-Trembles	45.641666	-73.500401
1	1	H2A	Saint-MichelEast	Saint-MichelEast	52.939916	-73.549136
2	2	H3A	Downtown MontrealNorth	McGill University	45.504785	-73.577151
3	3	H4A	Notre-Dame-de-GrâceNortheast	Notre-Dame-de-GrâceNortheast	45.465174	-73.632219
4	4	H5A	Place Bonaventure	Place Bonaventure	45.499444	-73.565000
5	5	H7A	Duvernay-Est	Duvernay-Est	45.647219	-73.615238
6	6	H9A	Dollard-des-OrmeauxNorthwest	Dollard-des-OrmeauxNorthwest	45.489564	-73.820557
7	7	H1B	Montreal East	Montreal East	45.632000	-73.506698
8	8	H2B	AhuntsicNorth	AhuntsicNorth	45.555235	-73.668203
9	9	H3B	Downtown MontrealEast	Downtown MontrealEast	45.503480	-73.568489

Visualize the Data Frame

```
map_mont = folium.Map(location=[45.508888, -73.561668], zoom_start=11)

for lat,lng,borough,neighborhood in
zip(df_montreal['Latitude'],df_montreal['Longitude'],df_montreal['Borough'],df_montreal['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 = 0.75,
                        parse_html=False).add_to(map_mont)

map_mont
```



Foursquare

"*Foursquare* is the most trusted, independent location data platform for understanding how people move through the real world." - foursquare site

Through this we will get all the venues or restaurants/fast food in all the Neighborhoods.

```
CLIENT_ID = 'client_id' # your Foursquare ID
CLIENT_SECRET = 'client_secret' # your Foursquare Secret
VERSION = '20180604'
radius = 1000
LIMIT = 200
```

```
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={}&categoryId={}&ll={},{}
&radius={}&limit={}'.format(
            CLIENT_ID,
            CLIENT_SECRET,
            VERSION,
            '4d4b7105d754a06374d81259',
            lat,
            lng,
            radius,
            LIMIT)

        # make the GET request
        try:
            results = requests.get(url).json()["response"]['groups'][0]['items']
        except KeyError:
            continue

        # 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)

df_nearby_mon = getNearbyVenues(names = df_montreal['Neighborhood'],
                                latitudes = df_montreal['Latitude'],
                                longitudes = df_montreal['Longitude'])
```

Here we get the data frame and now have the venues for each Neighborhood.

```
df_nearby_mon.head(5)
```

And the head table.

Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
Pointe-aux-Trembles	45.641666	-73.500401	Subway	45.641076	-73.501334	Sandwich Place
Pointe-aux-Trembles	45.641666	-73.500401	PFK	45.642370	-73.503428	Fast Food Restaurant
Pointe-aux-Trembles	45.641666	-73.500401	Dic Ann's	45.641219	-73.501674	Fast Food Restaurant
Pointe-aux-Trembles	45.641666	-73.500401	Thaï mix	45.642294	-73.502373	Asian Restaurant
Pointe-aux-Trembles	45.641666	-73.500401	Pizza Hut	45.641550	-73.503546	Pizza Place

We save the data we got from Foursquare as CSV.

```
df_nearby_mon.to_csv('montrealVenue.csv')
```

Clustering Analysis

We will find Neighborhood with less Asian Restaurant, we will cluster the Neighborhoods base on the data we gathered from Foursquare using unsupervised Machine Learning.

Read the saved csv file.

```
df_mont_venue = pd.read_csv('montrealVenue.csv')
```

Check the table.

```
df_mont_venue.head()
```

	Unnamed: 0	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	0	Pointe-aux-Trembles	45.641666	-73.500401	Subway	45.641076	-73.501334	Sandwich Place
1	1	Pointe-aux-Trembles	45.641666	-73.500401	PFK	45.642370	-73.503428	Fast Food Restaurant
2	2	Pointe-aux-Trembles	45.641666	-73.500401	Dic Ann's	45.641219	-73.501674	Fast Food Restaurant
3	3	Pointe-aux-Trembles	45.641666	-73.500401	Thaï mix	45.642294	-73.502373	Asian Restaurant
4	4	Pointe-aux-Trembles	45.641666	-73.500401	Pizza Hut	45.641550	-73.503546	Pizza Place

One-Hot Encoding

By definition: "*One hot encoding is a process by which categorical variables are converted into a form that could be provided to ML algorithms to do a better job in prediction*."

"This encoding is needed for feeding categorical data to many scikit-learn estimators, notably linear models and SVMs with the standard kernels." - Scikit-learn doc

```
# one hot encoding
mont_onehot = pd.get_dummies(df_mont_venue[['Venue Category']], prefix="", prefix_sep="")
mont_onehot['Neighborhood'] = df_mont_venue['Neighborhood']
fixed_columns = [mont_onehot.columns[-1]] + list(mont_onehot.columns[:-1])
mont_onehot = mont_onehot[fixed_columns]
```

Display the head table.

```
mont_onehot.head()
```

	Neighborhood	American Restaurant	Argentinian Restaurant	Asian Restaurant	BBQ Joint	Bagel Shop	Bakery	Belgian Restaurant	Bistro	Breakfast Spot	...	Sushi Restaurant	Swiss Restaurant	Szechuan Restaurant	Taco Place	Tapas Restaurant	Tex-Mex Restaurant	Thai Restaurant	Turkish Restaurant	Vegetarian / Vegan Restaurant	Vietnamese Restaurant
0	Pointe-aux-Trembles	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	Pointe-aux-Trembles	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	Pointe-aux-Trembles	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	Pointe-aux-Trembles	0	0	1	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	Pointe-aux-Trembles	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

5 rows × 86 columns

Let us group it by Neighborhood.

```
mont_grp = mont_onehot.groupby('Neighborhood').mean().reset_index()
mont_grp.head()
```

	Neighborhood	American Restaurant	Argentinian Restaurant	Asian Restaurant	BBQ Joint	Bagel Shop	Bakery	Belgian Restaurant	Bistro	Breakfast Spot	...	Sushi Restaurant	Swiss Restaurant	Szechuan Restaurant	Taco Place	Tapas Restaurant	Tex-Mex Restaurant	Thai Restaurant	Turkish Restaurant	Vegetarian / Vegan Restaurant	Vietnamese Restaurant
0	Laval-sur-le-Lac	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	AhuntsicCentral	0.0	0.0	0.166667	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	AhuntsicEast	0.0	0.0	0.166667	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	AhuntsicNorth	0.0	0.0	0.166667	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	AhuntsicSoutheast	0.0	0.0	0.166667	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 86 columns

Now we will create a table for the 20 most common venue.

```
def return_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_venues = 20

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

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

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

for i in np.arange(mont_grp.shape[0]):
        n_venues_sorted.iloc[i, 1:] = return_common_venues(mont_grp.iloc[i, :], num_venues)
```

Display the head of the newly created data frame.

```
n_venues_sorted.head(5)
```

	Neighborhood	1th Most Common Venue	2th Most Common Venue	3th 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	15th Most Common Venue	16th Most Common Venue	17th Most Common Venue	18th Most Common Venue	19th Most Common Venue	20th Most Common Venue
0	Laval-sur-le-Lac	Middle Eastern Restaurant	Vietnamese Restaurant	Food	Diner	Donut Shop	Dumpling Restaurant	Eastern European Restaurant	Falafel Restaurant	Fast Food Restaurant	...	Fish & Chips Shop	Food Court	Deli / Bodega	Food Truck	French Restaurant	Fried Chicken Joint	Gastropub	Gluten-free Restaurant	Greek Restaurant	Hawaiian Restaurant
1	AhuntsicCentral	Italian Restaurant	Deli / Bodega	Fast Food Restaurant	Pizza Place	Restaurant	Asian Restaurant	Fish & Chips Shop	Diner	Donut Shop	...	Eastern European Restaurant	Falafel Restaurant	Argentinian Restaurant	Filipino Restaurant	Bagel Shop	Bakery	Food Court	Food Truck	French Restaurant	Fried Chicken Joint
2	AhuntsicEast	Italian Restaurant	Deli / Bodega	Fast Food Restaurant	Pizza Place	Restaurant	Asian Restaurant	Fish & Chips Shop	Diner	Donut Shop	...	Eastern European Restaurant	Falafel Restaurant	Argentinian Restaurant	Filipino Restaurant	Bagel Shop	Bakery	Food Court	Food Truck	French Restaurant	Fried Chicken Joint
3	AhuntsicNorth	Italian Restaurant	Deli / Bodega	Fast Food Restaurant	Pizza Place	Restaurant	Asian Restaurant	Fish & Chips Shop	Diner	Donut Shop	...	Eastern European Restaurant	Falafel Restaurant	Argentinian Restaurant	Filipino Restaurant	Bagel Shop	Bakery	Food Court	Food Truck	French Restaurant	Fried Chicken Joint
4	AhuntsicSoutheast	Italian Restaurant	Deli / Bodega	Fast Food Restaurant	Pizza Place	Restaurant	Asian Restaurant	Fish & Chips Shop	Diner	Donut Shop	...	Eastern European Restaurant	Falafel Restaurant	Argentinian Restaurant	Filipino Restaurant	Bagel Shop	Bakery	Food Court	Food Truck	French Restaurant	Fried Chicken Joint

5 rows × 21 columns

Apply the Clustering Algorithm

from sklearn.cluster import KMeans

```
# set number of clusters
kclusters = 5

mont_cluster = mont_grp.drop('Neighborhood', 1)
# run k-means clustering
kmeans = KMeans(n_clusters=kclusters, random_state=0).fit(mont_cluster)
# check cluster labels generated for each row in the dataframe
kmeans.labels_[0:10]
```

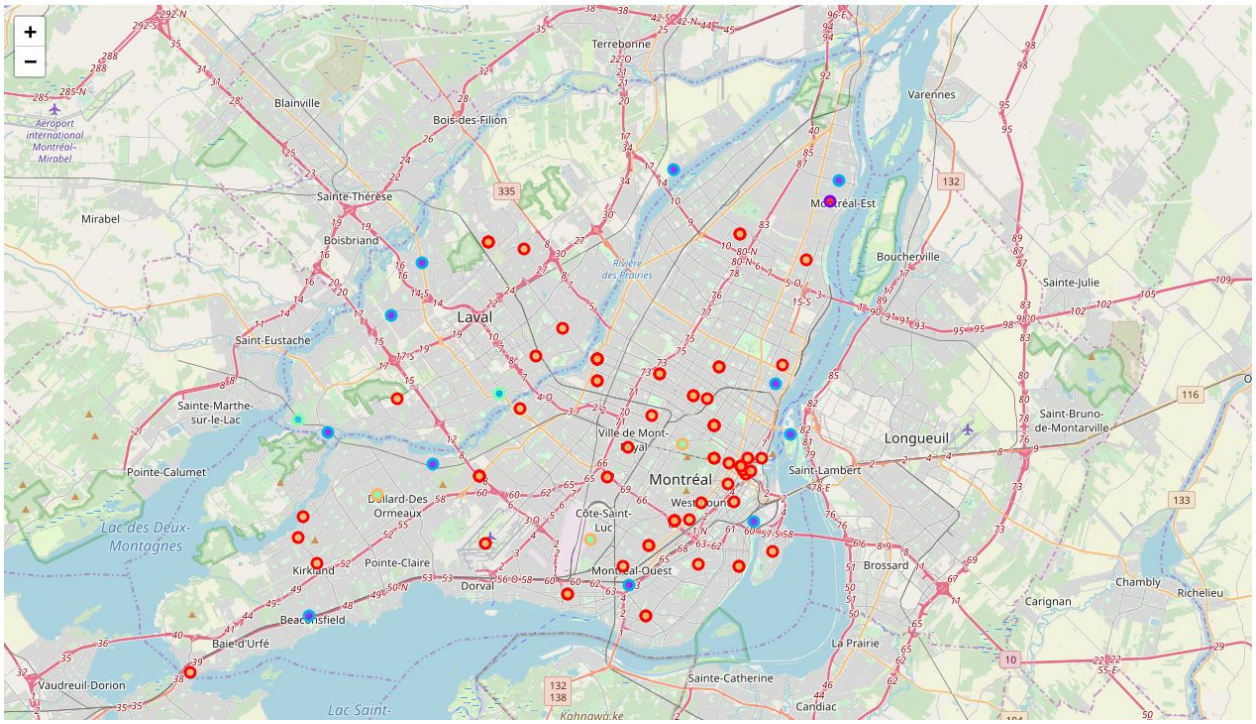
```
# add clustering labels
n_venues_sorted.insert(0, 'Cluster Labels', kmeans.labels_)

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

df_montreal_m.dropna(inplace=True)
df_montreal_m.head() # check the last columns!
```

	Unnamed: 0	PostalCode	Borough	Neighborhood	Latitude	Longitude	Cluster Labels	1th Most Common Venue	2th Most Common Venue	3th Most Common Venue	...	11th Most Common Venue	12th Most Common Venue	13th Most Common Venue	14th Most Common Venue	15th Most Common Venue	16th Most Common Venue	17th Most Common Venue	18th Most Common Venue	19th Most Common Venue	20th Most Common Venue
0	0	H1A	Pointe-aux-Trembles	Pointe-aux-Trembles	45.641666	-73.500401	2.0	Pizza Place	Fast Food Restaurant	Asian Restaurant	...	Eastern European Restaurant	Falafel Restaurant	Vietnamese Restaurant	Fish & Chips Shop	Deli / Bodega	Food Court	Food Truck	French Restaurant	Fried Chicken Joint	Gastropub
2	2	H3A	Downtown MontrealNorth	McGill University	45.504785	-73.577151	0.0	Sandwich Place	Café	Pizza Place	...	Dim Sum Restaurant	Diner	Donut Shop	Dumpling Restaurant	Eastern European Restaurant	Falafel Restaurant	Vietnamese Restaurant	Fast Food Restaurant	Filipino Restaurant	Deli / Bodega
3	3	H4A	Notre-Dame-de-GrâceNortheast	Notre-Dame-de-GrâceNortheast	45.465174	-73.632219	0.0	Pizza Place	Asian Restaurant	Japanese Restaurant	...	Falafel Restaurant	Fast Food Restaurant	Filipino Restaurant	Food Court	Food	Dim Sum Restaurant	Food Truck	French Restaurant	Fried Chicken Joint	Gastropub
4	4	H5A	Place Bonaventure	Place Bonaventure	45.499444	-73.565000	0.0	Café	Sandwich Place	Restaurant	...	Comfort Food Restaurant	Bistro	Cuban Restaurant	Deli / Bodega	Fast Food Restaurant	French Restaurant	Gastropub	BBQ Joint	Asian Restaurant	Vegetarian / Vegan Restaurant
5	5	H7A	Duvernay-Est	Duvernay-Est	45.647219	-73.615238	2.0	Pizza Place	Vietnamese Restaurant	Fish & Chips Shop	...	Food	Deli / Bodega	Food Court	Food Truck	French Restaurant	Fried Chicken Joint	Gastropub	Gluten-free Restaurant	Greek Restaurant	Hawaiian Restaurant

5 rows × 27 columns

Visualize the Cluster

```
# create map
map_mont_k = folium.Map(location=[45.508888, -73.561668], 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(df_montreal_m['Latitude'], df_montreal_m['Longitude'],
df_montreal_m['Neighborhood'], df_montreal_m['Cluster Labels']):
        label = folium.Popup(str(poi) + ' Cluster ' + str(cluster), parse_html=True)
        folium.CircleMarker(
        [lat, lon],
        radius=5,
        popup=label,
        color=rainbow[int(cluster)-1],
        fill=True,
        fill_color=rainbow[int(cluster)-2],
        fill_opacity=0.75).add_to(map_mont_k)

map_mont_k
```




```
clusterdata = pd.merge(mont_onehot.groupby('Neighborhood').sum(),
                       df_montreal_m[['Neighborhood','Cluster Labels']],
                       left_on='Neighborhood',
                       right_on='Neighborhood',how='inner')
clusterdata = clusterdata.iloc[:,1:].groupby('Cluster Labels').sum().transpose()
clusterdata.head(10)
```

Cluster Labels	0.0	1.0	2.0	3.0	4.0
American Restaurant	23	0	0	0	0
Argentinian Restaurant	1	0	0	0	0
Asian Restaurant	42	0	1	0	0
BBQ Joint	13	0	0	0	0
Bagel Shop	5	0	0	0	0
Bakery	51	0	0	0	6
Belgian Restaurant	1	0	0	0	0
Bistro	14	0	0	0	0
Breakfast Spot	56	0	0	0	0
Buffet	1	0	0	0	0

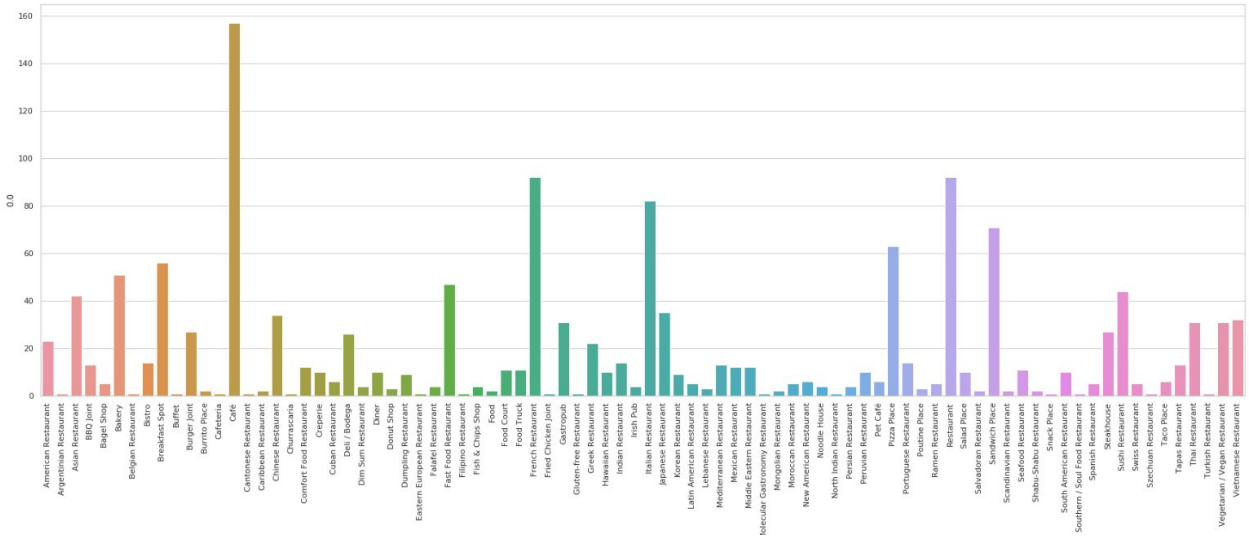
Analyze the Cluster

Here now we will plot the clusters using Seaborn and see the density of venues on each cluster.

```
import seaborn as sns

def plot_bar(clusternumber):
        sns.set(style="whitegrid",rc={'figure.figsize':(30,10)})
        df =
clusterdata[[clusternumber]].drop(clusterdata[[clusternumber]][clusterdata[clusternumber]==0].index)
        chart = sns.barplot(x=df.index, y=clusternumber, data=df)
        chart.set_xticklabels(chart.get_xticklabels(),rotation=90)
```

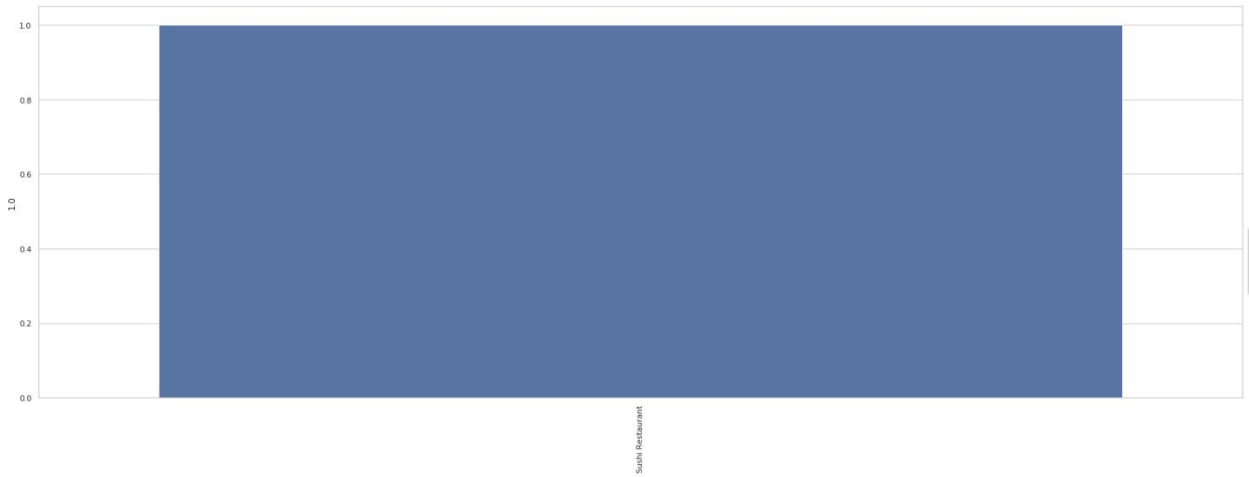
Let us plot one by one the 5 clusters that we made from the Clustering Algorithm.

```
plot_bar(0)
```



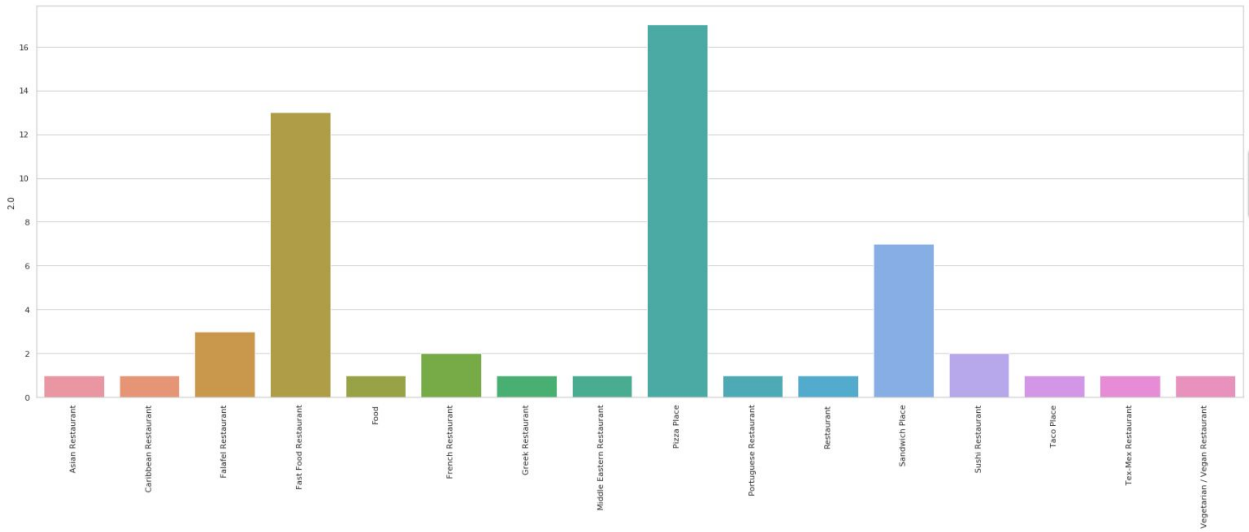
A lot of them as you can see these are the orange circles on the map.

```
plot_bar(1)
```



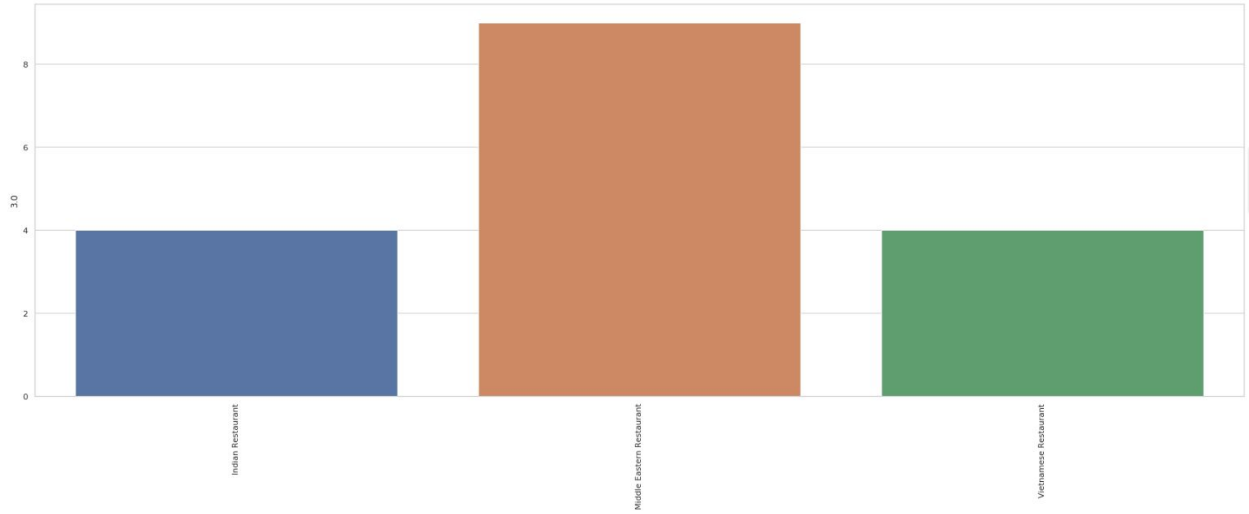
The only purple circle in our map.

```
plot_bar(2)
```



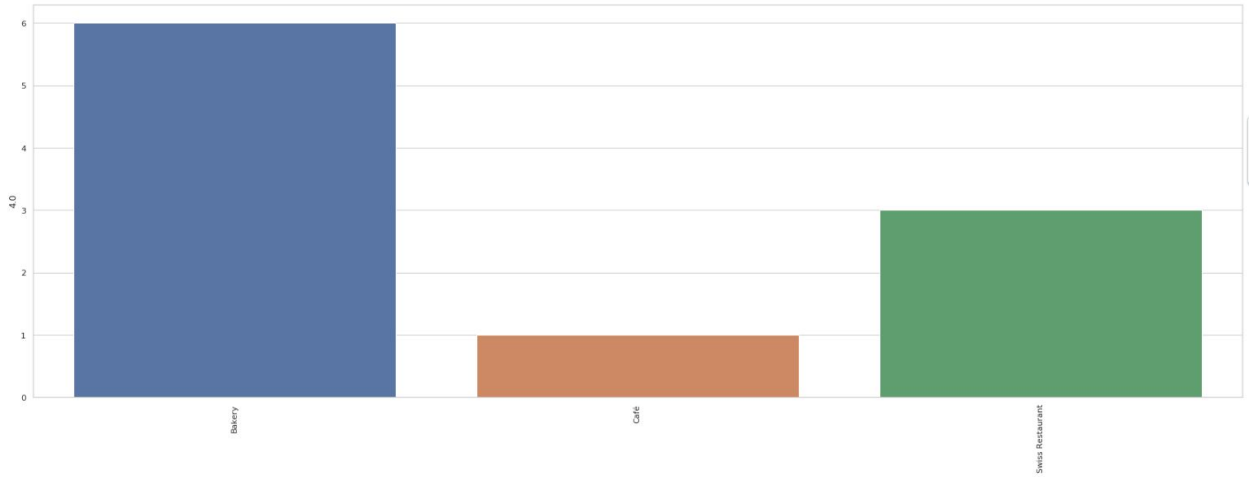
These are the blue circles in our map

```
plot_bar(3)
```



The orange circles.

```
plot_bar(4)
```



The green circles in our map.

So what is next?

Analyze the Result

We now analyze the result and from here we can recommend the place which is suitable for an Asian Restaurant.

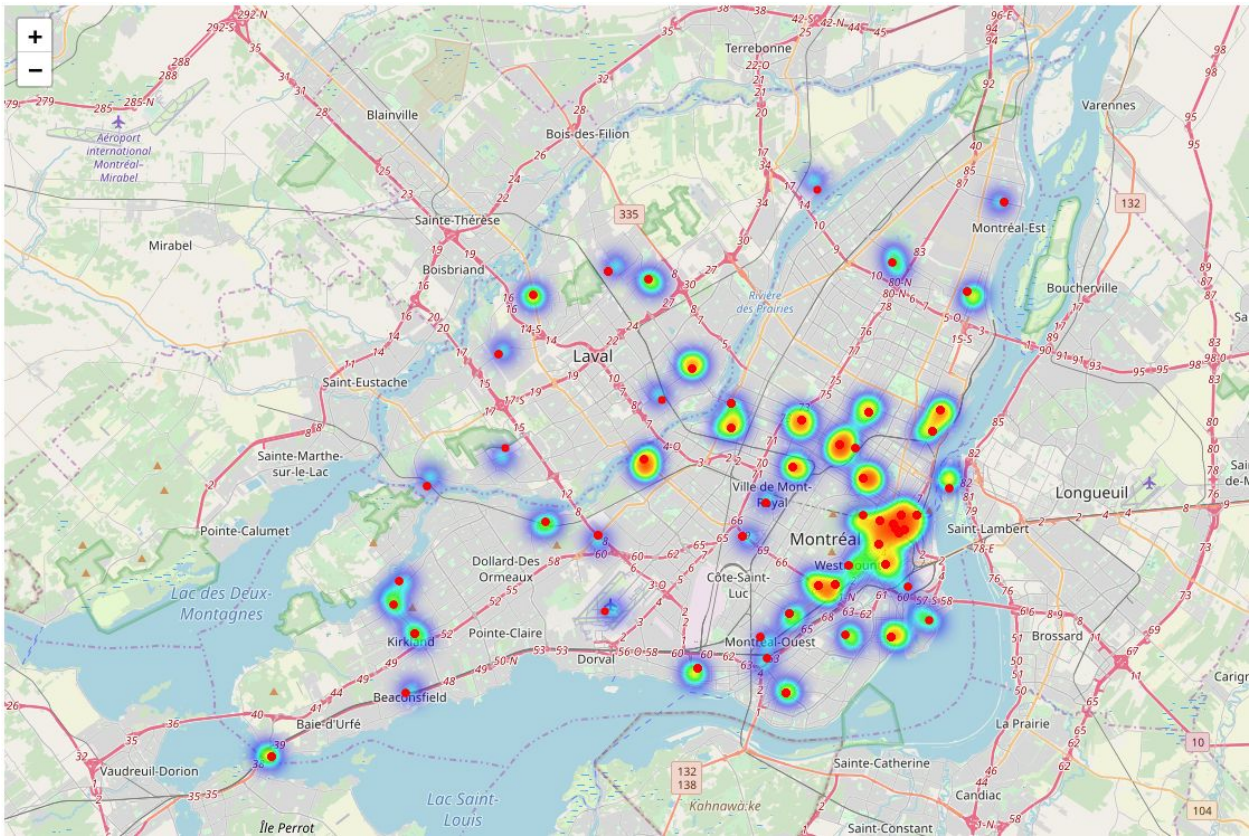
Let us create the heatmap of our result. According to our bar charts, we can use clusters 0 and 2.

```
mont_heatmap = df_mont_venue
mont_heatmap = pd.merge(mont_heatmap,df_montreal_m[['Neighborhood','Cluster
Labels']],left_on='Neighborhood', right_on='Neighborhood',how='inner')
mont_heatmap.drop(mont_heatmap[~mont_heatmap['Cluster Labels'].isin([0,2])].index, inplace=True)
mont_heatmap.head()
```

	Unnamed: 0	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category	Cluster Labels
0	0	Pointe-aux-Trembles	45.641666	-73.500401	Subway	45.641076	-73.501334	Sandwich Place	2.0
1	1	Pointe-aux-Trembles	45.641666	-73.500401	PFK	45.642370	-73.503428	Fast Food Restaurant	2.0
2	2	Pointe-aux-Trembles	45.641666	-73.500401	Dic Ann's	45.641219	-73.501674	Fast Food Restaurant	2.0
3	3	Pointe-aux-Trembles	45.641666	-73.500401	Thaï mix	45.642294	-73.502373	Asian Restaurant	2.0
4	4	Pointe-aux-Trembles	45.641666	-73.500401	Pizza Hut	45.641550	-73.503546	Pizza Place	2.0

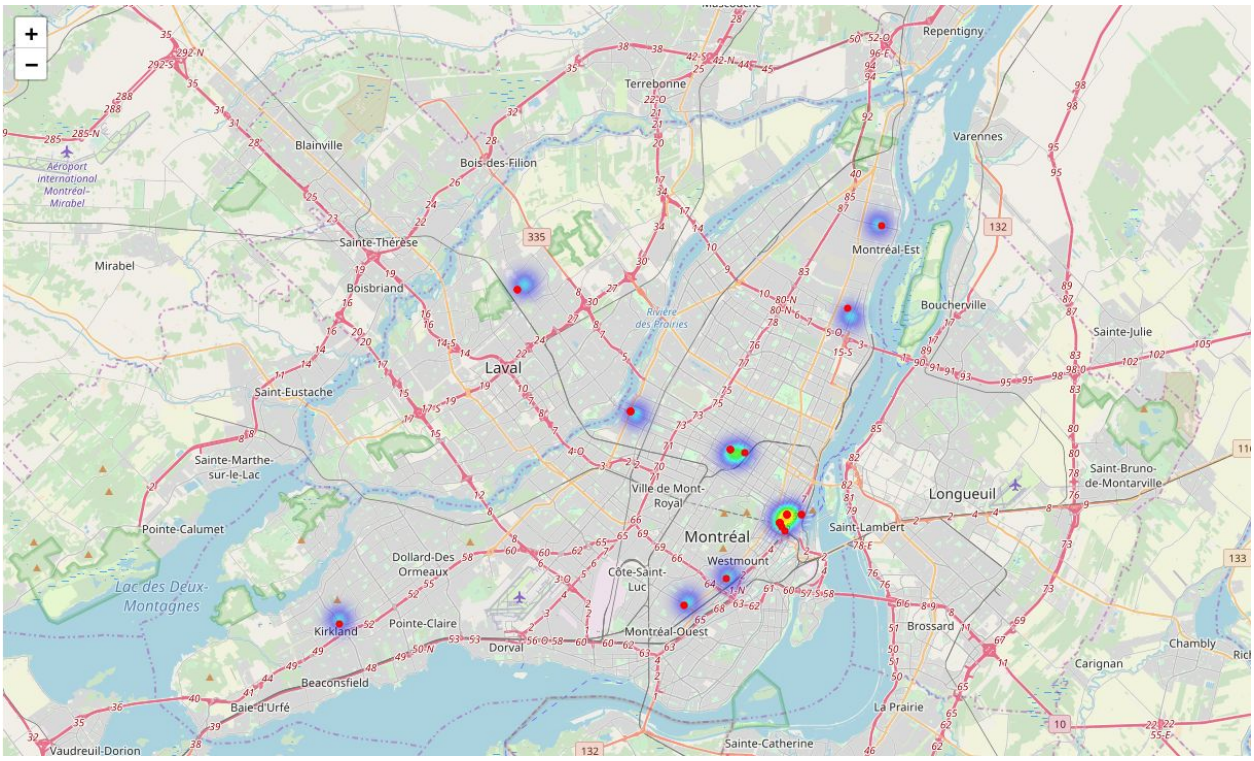
Heatmap of All Restaurants in Clusters 0 and 2

```
from folium.plugins import HeatMap

venue_heat = folium.Map(location=[45.508888, -73.561668], zoom_start=11)
HeatMap(list(zip(mont_heatmap['Venue Latitude'],mont_heatmap['Venue Longitude'])),
        min_opacity=0.2,
        radius=10, blur=15,
        max_zoom=1
        ).add_to(venue_heat)
for lat, lng, label in zip(mont_heatmap['Neighborhood Latitude'], mont_heatmap['Neighborhood
Longitude'], mont_heatmap['Neighborhood']):
        label = folium.Popup(label, parse_html=True)
        folium.CircleMarker(
        [lat, lng],
        radius=2,
        popup=label,
        color='red',
        fill=True,
        fill_color='#3186cc',
        fill_opacity=0.7,
        parse_html=False).add_to(venue_heat)
venue_heat
```



Heatmap of Asian Restaurants in Clusters 0 and 2

```
fil_res = mont_heatmap[mont_heatmap['Venue Category']=='Asian Restaurant']

fil_heat = folium.Map(location = [45.508888, -73.561668], zoom_start=11)
HeatMap(list(zip(fil_res['Venue Latitude'],fil_res['Venue Longitude'])),
        min_opacity=0.2,
        radius=10, blur=15,
        max_zoom=1
        ).add_to(fil_heat)
for lat, lng, label in zip(fil_res['Neighborhood Latitude'], fil_res['Neighborhood Longitude'],
fil_res['Neighborhood']):
        label = folium.Popup(label, parse_html=True)
        folium.CircleMarker(
        [lat, lng],
        radius=2,
        popup=label,
        color='red',
        fill=True,
        fill_color='#3186cc',
        fill_opacity=0.7,
        parse_html=False).add_to(fil_heat)
fil_heat
```



Now we remove neighborhood with this condition:

- Number of Asian restaurants >30%
- Number of all restaurants >60%

```
mont_all=mont_heatmap[['Neighborhood','Venue']].groupby('Neighborhood').count().sort_values(by='Venue')
target_count = int(0.6*len(mont_all))
print(mont_all.iloc[target_count])
mont_all.drop(mont_all[mont_all.Venue.values>7].index,inplace=True)
mont_all.columns=['all count']
mont_all.head()
```

```
Venue    9
Name: Côte-des-NeigesEast, dtype: int64
```

Neighborhood	all count
Duvernay-Est	1
Sainte-Dorothée	1
Laval-des-Rapides	1
Akwesasne Region1A0: Akwesasne	1
YUL	1

```
asian_count = mont_heatmap[mont_heatmap['Venue Category']=="Asian
Restaurant"][['Neighborhood','Venue']].groupby('Neighborhood').count().sort_values(by='Venue')
target_count = int(0.3*len(asian_count))
print(asian_count.iloc[target_count])
asian_count.drop(asian_count[asian_count.Venue.values>1].index,inplace=True)
asian_count.columns = ['asian count']
asian_count.head()
```

```
Venue    1
Name: Kirkland, dtype: int64
```

Neighborhood	asian count
AhuntsicCentral	1
RosemontCentral	1
Pointe-aux-Trembles	1
Place Bonaventure	1
Notre-Dame-de-GrâceSouthwest	1

```
lowdensity = asian_count.join(mont_all)
lowdensity.index.values
```

```
array(['AhuntsicCentral', 'RosemontCentral', 'Pointe-aux-Trembles',
       'Place Bonaventure', 'Notre-Dame-de-GrâceSouthwest',
       'Notre-Dame-de-GrâceNortheast', 'Montréal-NordNorth', 'Kirkland',
       'MercierWest', 'AuteuilWest', 'AuteuilSouth', 'AuteuilNortheast',
       'AhuntsicSouthwest', 'AhuntsicSoutheast', 'AhuntsicNorth',
       'AhuntsicEast', 'WestmountNorth'], dtype=object)
```

```
recommend = df_montreal
recommend.drop(recommend[~recommend['Neighborhood'].isin(lowdensity.index.values)].index, inplace=True)
recommend.head()
```

	Neighborhood	Latitude	Longitude	Place Bonaventure	Montréal-NordNorth	Kirkland	WestmountNorth
0	Pointe-aux-Trembles	45.641666	-73.500401	16.5997	16.4168	34.9444	19.9633
1	Notre-Dame-de-GrâceNortheast	45.465174	-73.632219	6.4816	6.4932	17.9814	2.5719
2	Place Bonaventure	45.499444	-73.565000	0.0000	0.3054	23.6871	3.9172
3	AhuntsicNorth	45.555235	-73.668203	10.1579	9.8665	18.7283	9.9593
4	Notre-Dame-de-GrâceSouthwest	45.465174	-73.632219	6.4816	6.4932	17.9814	2.5719

```
popular_neighbor=df_mont_venue[['Neighborhood','Venue']].groupby('Neighborhood').count().sort_values(by=
'Venue', ascending=False).head(30).index.values
popular_neighbor
```

```
array(['Saint-LaurentEast', 'Place Desjardins', 'Montréal-NordNorth',
       'Downtown MontrealEast', 'Downtown MontrealSouthwest',
       'Downtown MontrealNortheast', 'Reserved0H0: Santa Claus',
       'Concordia University', 'Old Montreal', 'Tour de la Bourse',
       'Plateau Mont-RoyalWest', 'Plateau Mont-RoyalNorth',
       'Plateau Mont-RoyalNorth Central',
       'Plateau Mont-RoyalSouth Central', 'Place Bonaventure',
       'Plateau Mont-RoyalSoutheast', 'VerdunNorth', 'VerdunSouth',
       'Petite-PatrieSouthwest', 'Petite-PatrieNortheast',
       'WestmountNorth', 'Petite-Bourgogne', 'Maisonneuve', 'Kirkland',
       'McGill University', 'VilleraySoutheast',
       'Sainte-Anne-De-Bellevue', 'Villeray Northeast', 'LaSalleSoutheast',
       'LaSalleNorthwest'], dtype=object)
```

```
toplatlng=df_montreal[['Neighborhood','Latitude','Longitude']][df_montreal['Neighborhood'].isin(popular_
neighbor)].reset_index()
toplatlng
```

	index	Neighborhood	Latitude	Longitude
0	4	Place Bonaventure	45.499444	-73.565000
1	26	Montréal-NordNorth	45.501689	-73.567256
2	44	Kirkland	45.456041	-73.862334
3	115	WestmountNorth	45.477638	-73.604446

```
from math import sin, cos, sqrt, atan2, radians

def distanceInKM(la1,lo1,la2,lo2):
        # approximate radius of earth in km
        R = 6373.0
        lat1 = radians(la1)
        lon1 = radians(lo1)
        lat2 = radians(la2)
        lon2 = radians(lo2)
        dlon = lon2 - lon1
        dlat = lat2 - lat1
        a = sin(dlat / 2)**2 + cos(lat1) * cos(lat2) * sin(dlon / 2)**2
        c = 2 * atan2(sqrt(a), sqrt(1 - a))
        dis = R * c
        return round(dis,4)

recommend.reset_index(inplace = True)
recommend.drop(columns=["index","Unnamed: 0","PostalCode", "Borough"], inplace=True)
for i in toplatlng.index:
        recommend[toplatlng.iloc[i]['Neighborhood']] = recommend.apply(lambda x :
distanceInKM(toplatlng.iloc[i]['Latitude'],toplatlng.iloc[i]['Longitude'],x['Latitude'],x['Longitude']),
axis=1)
recommend.head()
```

	Neighborhood	Latitude	Longitude	Place Bonaventure	Montréal-NordNorth	Kirkland	WestmountNorth
0	Pointe-aux-Trembles	45.641666	-73.500401	16.5997	16.4168	34.9444	19.9633
1	Notre-Dame-de-GrâceNortheast	45.465174	-73.632219	6.4816	6.4932	17.9814	2.5719
2	Place Bonaventure	45.499444	-73.565000	0.0000	0.3054	23.6871	3.9172
3	AhuntsicNorth	45.555235	-73.668203	10.1579	9.8665	18.7283	9.9593
4	Notre-Dame-de-GrâceSouthwest	45.465174	-73.632219	6.4816	6.4932	17.9814	2.5719

```
nearPlaceBonaventure = recommend.sort_values(by=['Place
Bonaventure']).iloc[:,:3].head().set_index('Neighborhood')
nearPlaceBonaventure
```

Neighborhood	Latitude	Longitude
Place Bonaventure	45.499444	-73.565000
Montréal-NordNorth	45.501689	-73.567256
WestmountNorth	45.477638	-73.604446
RosemontCentral	45.536080	-73.591702
Notre-Dame-de-GrâceNortheast	45.465174	-73.632219

```
nearMontrealNorth=recommend.sort_values(by=['Montréal-NordNorth']).iloc[:,:3].head().set_index('Neighbor
hood')
nearMontrealNorth
```

Neighborhood	Latitude	Longitude
Montréal-NordNorth	45.501689	-73.567256
Place Bonaventure	45.499444	-73.565000
WestmountNorth	45.477638	-73.604446
RosemontCentral	45.536080	-73.591702
Notre-Dame-de-GrâceNortheast	45.465174	-73.632219

```
nearKirkland = recommend.sort_values(by=['Kirkland']).iloc[:,:3].head().set_index('Neighborhood')
nearKirkland
```

Neighborhood	Latitude	Longitude
Kirkland	45.456041	-73.862334
Notre-Dame-de-GrâceNortheast	45.465174	-73.632219
Notre-Dame-de-GrâceSouthwest	45.465174	-73.632219
AhuntsicSoutheast	45.555235	-73.668203
AhuntsicNorth	45.555235	-73.668203

```
nearWestmountNorth=recommend.sort_values(by=['WestmountNorth']).iloc[:,:3].head().set_index('Neighborhood')
nearWestmountNorth
```

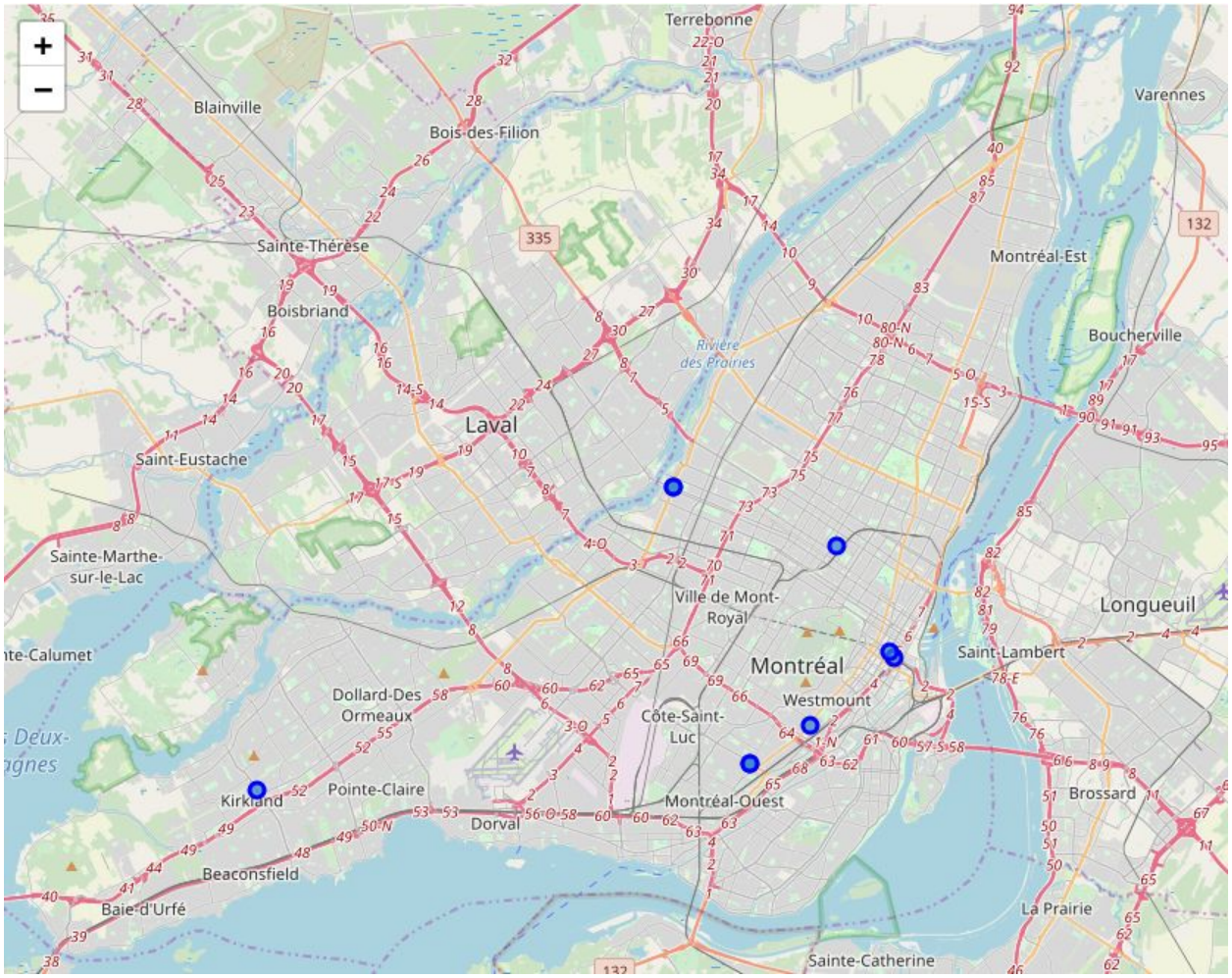
Neighborhood	Latitude	Longitude
WestmountNorth	45.477638	-73.604446
Notre-Dame-de-GrâceNortheast	45.465174	-73.632219
Notre-Dame-de-GrâceSouthwest	45.465174	-73.632219
Place Bonaventure	45.499444	-73.565000
Montréal-NordNorth	45.501689	-73.567256

Final Recommendation

```
final=nearPlaceBonaventure.append(nearMontrealNorth).append(nearKirkland).append(nearWestmountNorth).reset_index()
final.drop_duplicates(inplace=True)
final.reset_index(inplace=True)
final.drop(columns=['index'],inplace=True)
final
```

	Neighborhood	Latitude	Longitude
0	Place Bonaventure	45.499444	-73.565000
1	Montréal-NordNorth	45.501689	-73.567256
2	WestmountNorth	45.477638	-73.604446
3	RosemontCentral	45.536080	-73.591702
4	Notre-Dame-de-GrâceNortheast	45.465174	-73.632219
5	Kirkland	45.456041	-73.862334
6	Notre-Dame-de-GrâceSouthwest	45.465174	-73.632219
7	AhuntsicSoutheast	45.555235	-73.668203
8	AhuntsicNorth	45.555235	-73.668203

```
finalMap = folium.Map(location=[45.508888, -73.561668], zoom_start=11)

# add markers to map
for lat, lng, label in zip(final['Latitude'], final['Longitude'], final['Neighborhood']):
        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(finalMap)

finalMap
```



Conclusion

Number of Neighborhood = 124 Number of Restaurants or Venues = 1543 Number of Recommended Places = 9

Among the places, the top four are:

Neighborhood	Latitude	Longitude
Place Bonaventure	45.499444	73.565000
Montréal-NordNorth	45.501689	-73.567256
WestmountNorth	45.477638	-73.604446
RosemontCentral	45.536080	-73.591702

Though this is not really conclusive due to some factors missing like: population, accessibility, how expensive the rent on each top neighborhood and much more.