Capstone Project - The Battle of the Neighborhoods

Applied Data Science Capstone for the IBM Data Science Professional Certificate (Coursera)

Fanny Balestrat

June 2021

1. Introduction and Business Problem

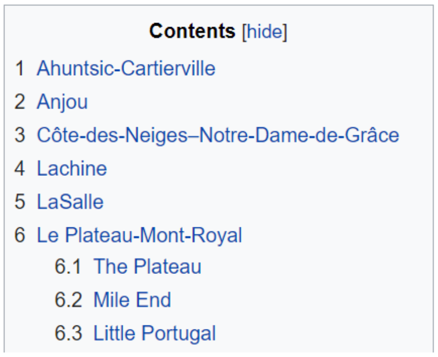
In this project, we are helping a friend who lives in Montreal to move to another neighborhood within the city. He asked us to suggest to him a neighborhood which is relatively similar to the one he lives in now in terms of population density and the proximity of restaurants, shops and venues. He has been living for several years in Côte des Neiges - Notre Dame de Grâce, a neighborhood located in the center of Montreal's island, but needs a change. The data science tools we learned will help us narrow down the optimal neighborhoods for him, by using two different clustering models.

2. Data Presentation

Two sets of data will be needed to solve the problem:

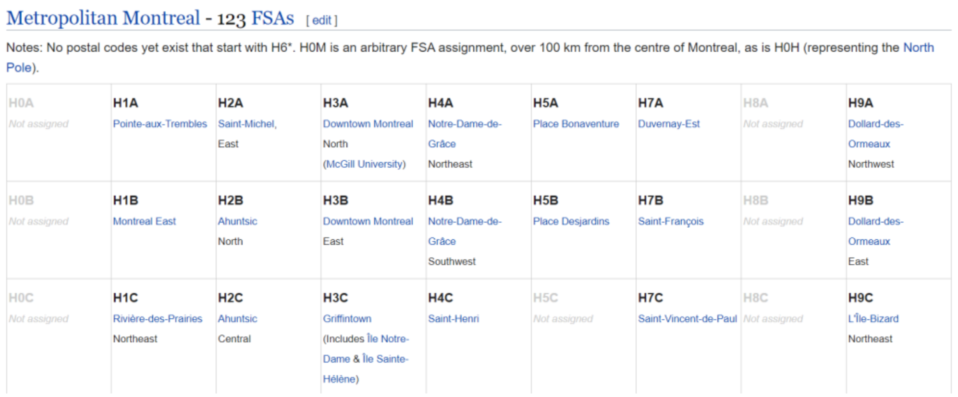
1. Demographic information about the population of Montreal by neighborhood
2. Proximity and density of venues in each neighborhood

The list of Montreal's neighborhoods is found from Wikipedia [1]. The required data will be scraped from the site using the Beautiful Soup package. The demographic information is found from the last Montreal Census data (from 2016) [2]. The data is already available as a .CSV file and provides the population density by neighborhood as well as each neighborhood area. The postal codes of each neighborhood are found from Wikipedia [3]. Finally, the proximity and density of venues by neighborhood is found using the Foursquare API [4]. Coordinates of each neighborhood are determined using the Geocoder API.


Contents [hide]

1 Ahuntsic-Cartierville
2 Anjou
3 Côte-des-Neiges–Notre-Dame-de-Grâce
4 Lachine
5 LaSalle
6 Le Plateau-Mont-Royal
   6.1 The Plateau
   6.2 Mile End
   6.3 Little Portugal

Figure 1: List of Montreal’s Neighborhoods from Wikipedia [1]

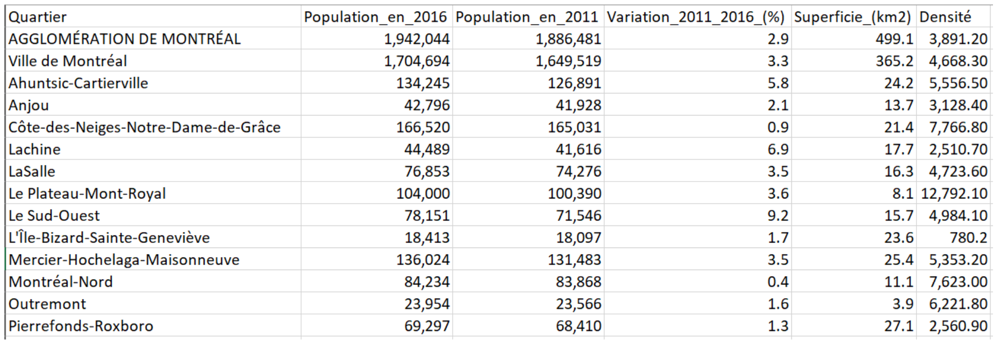


Metropolitan Montreal - 123 FSAs [edit]

Notes: No postal codes yet exist that start with H6*. H0M is an arbitrary FSA assignment, over 100 km from the centre of Montreal, as is H0H (representing the North Pole).

H0A Not assigned	H1A Pointe-aux-Trembles	H2A Saint-Michel, East	H3A Downtown Montreal North (McGill University)	H4A Notre-Dame-de-Grâce Northeast	H5A Place Bonaventure	H7A Duvernay-Est	H8A Not assigned	H9A Dollard-des-Ormeaux Northwest
H0B Not assigned	H1B Montreal East	H2B Ahuntsic North	H3B Downtown Montreal East	H4B Notre-Dame-de-Grâce Southwest	H5B Place Desjardins	H7B Saint-François	H8B Not assigned	H9B Dollard-des-Ormeaux East
H0C Not assigned	H1C Rivière-des-Prairies Northeast	H2C Ahuntsic Central	H3C Griffintown (Includes Île Notre-Dame & Île Sainte-Hélène)	H4C Saint-Henri	H5C Not assigned	H7C Saint-Vincent-de-Paul	H8C Not assigned	H9C L'Île-Bizard Northeast

Figure 2: List of Montreal's Postal Codes from Wikipedia [3]



Quartier	Population_en_2016	Population_en_2011	Variation_2011_2016_(%)	Superficie_(km2)	Densité
AGGLOMÉRATION DE MONTRÉAL	1,942,044	1,886,481	2.9	499.1	3,891.20
Ville de Montréal	1,704,694	1,649,519	3.3	365.2	4,668.30
Ahuntsic-Cartierville	134,245	126,891	5.8	24.2	5,556.50
Anjou	42,796	41,928	2.1	13.7	3,128.40
Côte-des-Neiges-Notre-Dame-de-Grâce	166,520	165,031	0.9	21.4	7,766.80
Lachine	44,489	41,616	6.9	17.7	2,510.70
LaSalle	76,853	74,276	3.5	16.3	4,723.60
Le Plateau-Mont-Royal	104,000	100,390	3.6	8.1	12,792.10
Le Sud-Ouest	78,151	71,546	9.2	15.7	4,984.10
L'Île-Bizard-Sainte-Geneviève	18,413	18,097	1.7	23.6	780.2
Mercier-Hochelaga-Maisonneuve	136,024	131,483	3.5	25.4	5,353.20
Montréal-Nord	84,234	83,868	0.4	11.1	7,623.00
Outremont	23,954	23,566	1.6	3.9	6,221.80
Pierrefonds-Roxboro	69,297	68,410	1.3	27.1	2,560.90

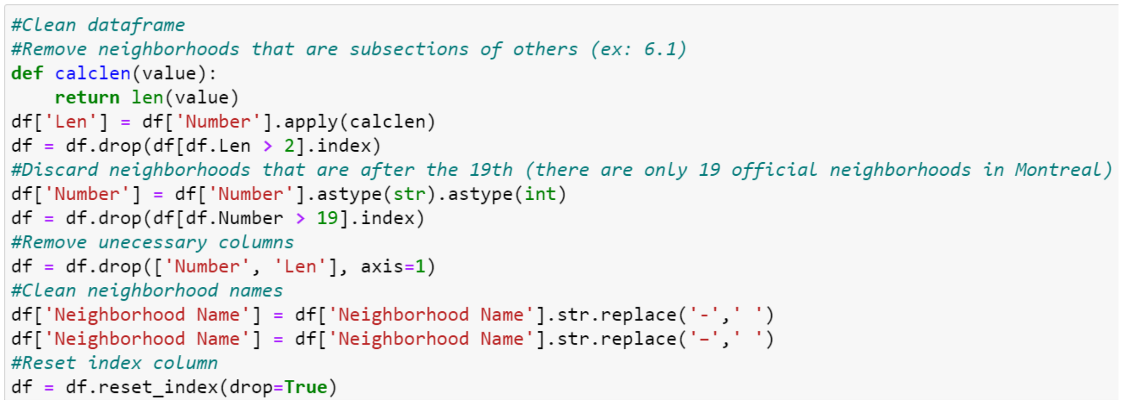
Figure 3: 2016 Montreal Census Information [2]

3. Methodology

3.1 Data Cleaning

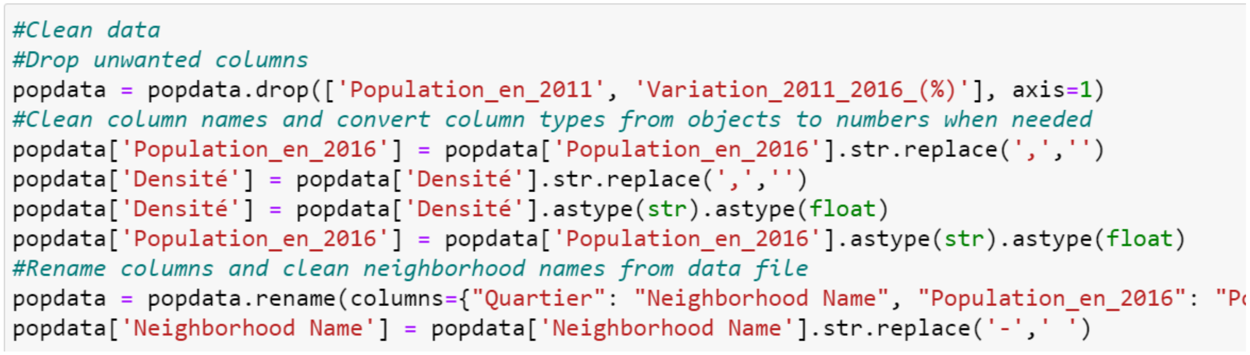
The data imported or scraped from different sources is cleaned and combined into one table. The unnecessary information is discarded, only to keep the required information for the further analysis to be done.

For the list of Montreal's neighborhoods, only the main 19 official neighborhoods are kept. The sub-neighborhoods are discarded as well as other cities or regions around Montreal that were also listed on the Wikipedia site. The name of each neighborhood is also cleaned, removing all dashes and replacing them with spaces.



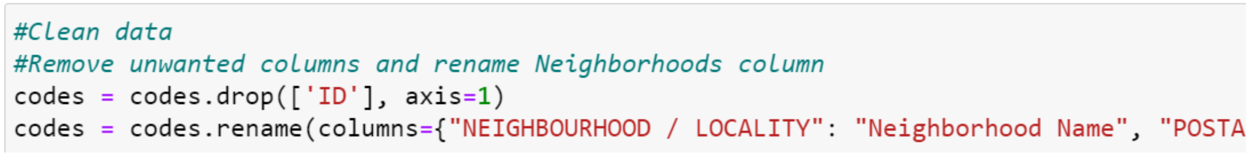
```python
#Clean dataframe
#Remove neighborhoods that are subsections of others (ex: 6.1)
def calclen(value):
    return len(value)
df['Len'] = df['Number'].apply(calclen)
df = df.drop(df[df.Len > 2].index)
#Discard neighborhoods that are after the 19th (there are only 19 official neighborhoods in Montreal)
df['Number'] = df['Number'].astype(str).astype(int)
df = df.drop(df[df.Number > 19].index)
#Remove unecessary columns
df = df.drop(['Number', 'Len'], axis=1)
#Clean neighborhood names
df['Neighborhood Name'] = df['Neighborhood Name'].str.replace('-',' ')
df['Neighborhood Name'] = df['Neighborhood Name'].str.replace('–',' ')
#Reset index column
df = df.reset_index(drop=True)
```

For the 2016 Montreal census information, the unnecessary columns regarding 2011 information and variation between 2011 and 2016 are dropped. Columns names are also modified from French to English and certain columns are casted as integer type instead of object type for further calculations on those columns. Finally, neighborhood names are cleaned, removing all dashes and replacing them with spaces.



```python
#Clean data
#Drop unwanted columns
popdata = popdata.drop(['Population_en_2011', 'Variation_2011_2016_(%)'], axis=1)
#Clean column names and convert column types from objects to numbers when needed
popdata['Population_en_2016'] = popdata['Population_en_2016'].str.replace(',','')
popdata['Densité'] = popdata['Densité'].str.replace(',','')
popdata['Densité'] = popdata['Densité'].astype(str).astype(float)
popdata['Population_en_2016'] = popdata['Population_en_2016'].astype(str).astype(float)
#Rename columns and clean neighborhood names from data file
popdata = popdata.rename(columns={"Quartier": "Neighborhood Name", "Population_en_2016": "Po
popdata['Neighborhood Name'] = popdata['Neighborhood Name'].str.replace('-',' ')
```

For the list of postal codes, a CSV file is imported with the required information. Unnecessary columns are dropped, and column names are modified to match the names of the columns in the previous tables imported.



```python
#Clean data
#Remove unwanted columns and rename Neighborhoods column
codes = codes.drop(['ID'], axis=1)
codes = codes.rename(columns={"NEIGHBOURHOOD / LOCALITY": "Neighborhood Name", "POSTA
```

3.2 Geocoder API

The Geocoder API is used to retrieve latitude and longitude coordinates of each neighborhood's center, based on each postal code. The name of each neighborhood as well as the retrieved postal code are passed to the API, and longitude and latitude values are returned.

```python
#Use the Geocoder API to retrieve coordinates
#Define function that returns latitude and longitude from a postal code and a neighborhood
def get_coordinates(postalcode, neighborhood):
    lat_lng_coords = None
    while(lat_lng_coords is None):
        g = geocoder.arcgis('{}, {}'.format(postalcode, neighborhood))
        lat_lng_coords = g.latlng
        latitude = lat_lng_coords[0]
        longitude = lat_lng_coords[1]
    return latitude, longitude
```

```python
#Obtain latitude and longitude coordinates for each neighborhood
for row in range(0,len(codes),1):
    postalcode = codes.loc[row, 'Postal Codes']
    neighborhood = codes.loc[row, 'Neighborhood Name']
    lat, lon = get_coordinates(postalcode, neighborhood)
    codes.loc[codes.index[row], 'Latitude'] = lat
    codes.loc[codes.index[row], 'Longitude'] = lon
```

3.3 Data Visualization

From the coordinates of each neighborhood retrieved using the Geocoder API, a map of Montreal is obtained using the Folium package, for better visualization of the neighborhoods within the city. Neighborhoods are indicated with a blue dot, and markers are added to every dot with the neighborhood name. 19 official neighborhoods are listed in Montreal; therefore 19 dots are visible on the map obtained.

```python
#Use Folium library to create Montreal map
#Create map of Montreal using latitude and longitude values
map_montreal = folium.Map(location=[latitude, longitude], zoom_start=10)

#Add markers to map
for lat, lng, neighborhood in zip(codes['Latitude'], codes['Longitude'], codes['Neighborhood Name']):
    label = '{}'.format(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(map_montreal)
#Show map
map_montreal
```

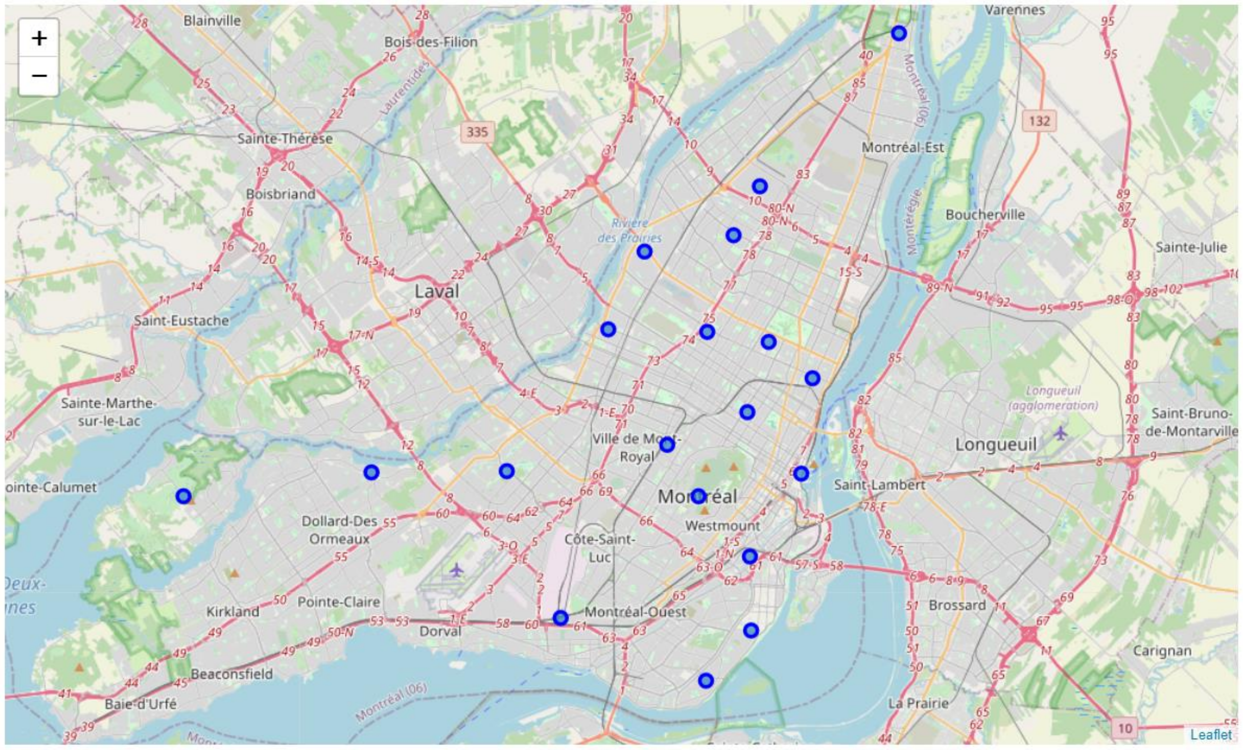


Figure 4: Map of Montreal and its 19 Neighborhoods obtained using Folium

3.4 Foursquare API

The Foursquare API is then used to obtain information about the venues in each neighborhood. First, the credentials (client ID, client secret, version and client token) are declared. A limit of 100 venues is first used as the maximum number of venues returned for every search. This limit will be verified later on, to make sure that it is high enough. The search radius for every neighborhood is then calculated. It is calculated using the area (in squared kilometers), available in the 2016 census information csv file. A function is defined to find the radius and add it to the data.

In order to make sure that venues are retrieved only once (only attributed to one neighborhood, not several), the radius returned is multiplied by a certain coefficient. The coefficient is first initialized as 1. Once the venues are returned for every neighborhood, it is verified if some venues are returned twice or more. If yes, the radius coefficient is reduced. This process is done iteratively until venues are only returned once and attributed to their main neighborhood only. The final coefficient found is 0.42. This coefficient introduces some errors in the analysis, since certain neighborhoods are larger than others, and not all venues would be returned in those cases. However, we can assume that we are interested in the main venues of each neighborhood, located near the center, and that the few venues that would not be retrieved would not have changed much the final results obtained.

```python
#Obtain radius of each Neighborhood based on their areas to specify search radius for Foursquare API
#Keep only a certain percentage of the radii obtained, so that neighborhood limits don't cross each others
radius_coeff = 0.42
def calcradius(value):
    radius = np.sqrt(value/np.pi)
    return (radius*1000*radius_coeff)
alldata['Radius(m)'] = alldata['Area (km2)'].apply(calcradius)
```

```python
#Define function that returns nerby venues for each Neighborhood center
def getNearbyVenues(names, latitudes, longitudes, radius):

    venues_list=[]
    for name, lat, lng, rad in zip(names, latitudes, longitudes, radius):

        #Create the API request URL
        url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={},{}&radius={}&limit={}'.form
            CLIENT_ID,
            CLIENT_SECRET,
            VERSION,
            lat,
            lng,
            rad,
            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 Name',
                  'Neighborhood Latitude',
                  'Neighborhood Longitude',
                  'Venue',
                  'Venue Latitude',
                  'Venue Longitude',
                  'Venue Category']

    return(nearby_venues)
```

```python
#Get all nearby venues for each neighborhood
montreal_venues = getNearbyVenues(names=alldata['Neighborhood Name'],
                                  latitudes=alldata['Latitude'], longitudes=alldata['Longitude'], radius=alldata['Radius(m)'])
```

```python
#Make sure that venues are not listed more than once (meaning that they would appear in two different
#neighborhoods)
#If they are listed twice (meaning that the function below returns False), reduce the radius of each
#neighborhood calculated from their respective areas, so that neighborhood limits don't cross each other
#From this iterative process, the optimal (largest radius coefficient that does not return
#duplicate venues is 0.42)
print(montreal_venues["Venue Latitude"].is_unique and montreal_venues["Venue Longitude"].is_unique)
```

Since we are interested only in the number of venues for each neighborhood, we group and count the venues returned by the API and keep only the total number by neighborhood.

```python
#Retrieve the number of venues in each neighborhood and put data into a dataframe
num_venues = montreal_venues.groupby('Neighborhood Name').count()['Venue']
num_venues = num_venues.to_frame()
num_venues = num_venues.rename(columns={"Venue": "Number of Venues"})
```

3.5 Merging All Data

All the data imported previously, scraped from websites or retrieves from the Geocoder or Foursquare API is merged into a single data frame. The data now contains each neighborhood name, postal code, longitude and latitude coordinates, population, area, population density, radius and number of venues. This data frame will be used for the clustering analysis.

	Neighborhood Name	Postal Codes	Latitude	Longitude	Population (2016)	Area (km2)	Population Density	Radius(m)	Number of Venues
0	Ahuntsic Cartierville	H2C	45.56094	-73.65921	134245.0	24.2	5556.5	1165.687225	62
1	Anjou	H1J	45.61588	-73.57606	42796.0	13.7	3128.4	877.070770	8
2	Côte des Neiges Notre Dame de Grâce	H3V	45.49737	-73.61008	166520.0	21.4	7766.8	1096.178402	44
3	Lachine	H8S	45.45041	-73.68545	44489.0	17.7	2510.7	996.921557	4
4	LaSalle	H8P	45.42628	-73.60552	76853.0	16.3	4723.6	956.683219	18
5	Le Plateau Mont Royal	H2J	45.52957	-73.58281	104000.0	8.1	12792.1	674.398916	100
6	Le Sud Ouest	H3H	45.47426	-73.58158	78151.0	15.7	4984.1	938.910466	95
7	L'Île Bizard Sainte Geneviève	H9C	45.49734	-73.89177	18413.0	23.6	780.2	1151.145859	5
8	Mercier Hochelaga Maisonneuve	H1W	45.54239	-73.54722	136024.0	25.4	5353.2	1194.238897	68
9	Montréal Nord	H1H	45.59120	-73.63897	84234.0	11.1	7623.0	789.470385	17

3.6 Clustering Analysis

Two different models are used for clustering the neighborhoods, K-Means model, and Hierarchical model. Using two different models increases the accuracy of the final predictions and allows for better results to be obtained. For each model, different numbers of clusters are used. This number is varied to analyze its impact on the final results obtained. The results shown at the end show the predictions of each model depending on the number of clusters used.

3.6.1 K-Means Model

To use the K-Means model, the data is first scaled, using a standard scaler, and then fit. The model is then initiated, and used to obtain a label for each neighborhood, depending on the number of clusters used. Each label is added to the data. An additional column is then added to indicate if the label of each neighborhood is similar to the label of our initial neighborhood, Côte des Neiges, to which we are comparing all others to.

```
#Scale data
x = kmeans_data.values[:,1:]
StandardScaler().fit_transform(x);
```

```
#Fit model
k_means = KMeans(init = "k-means++", n_clusters = clusterNum, n_init = 12)
k_means.fit(x)
labels = k_means.labels_
```

```
#Add labels to data for each neighborhood
kmeans_data["Label"] = labels
kmeans_data.head(19)
```

```
#Add column to indicate if neighborhood is in the same cluster as Côte des Neiges
cond = []
for idx in range(0, len(kmeans_data), 1):
    diff = kmeans_data.iloc[idx]['Label'] == kmeans_data.iloc[2]['Label']
    #print(diff)
    if diff:
        cond.append('1')
    else:
        cond.append('0')
kmeans_data['Similar on Kmeans?'] = pd.DataFrame(cond)
kmeans_data.head()
```

We then plot a graph of Population Density VS Number of venues (our two main independent variables) for each neighborhood, and color their respective dots based on their cluster.

```
#Plot Population Density VS Number of Venues for by Neighborhood and their respective cluster
plt.scatter(x[:, 0], x[:, 1], c=labels, alpha=0.5)
plt.xlabel('Population Density', fontsize=18)
plt.ylabel('Number of Venues', fontsize=16)
plt.title('Population Density VS Number of Venues for by Neighborhood', fontsize=16)
plt.show()
```

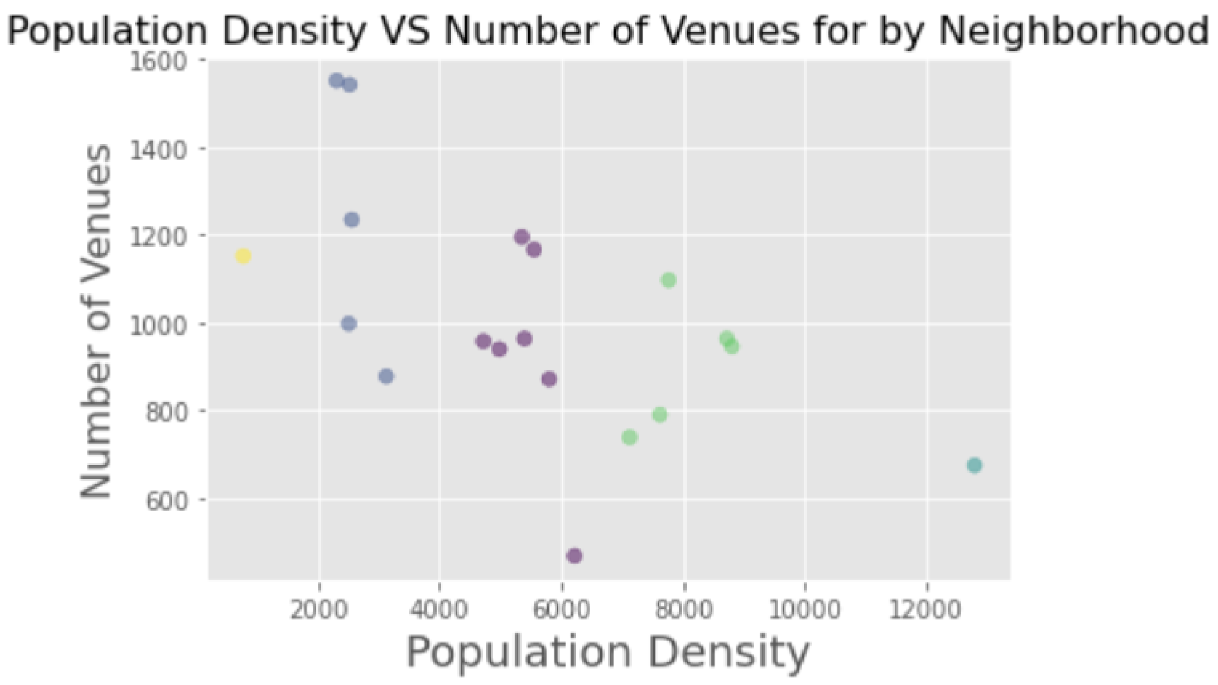
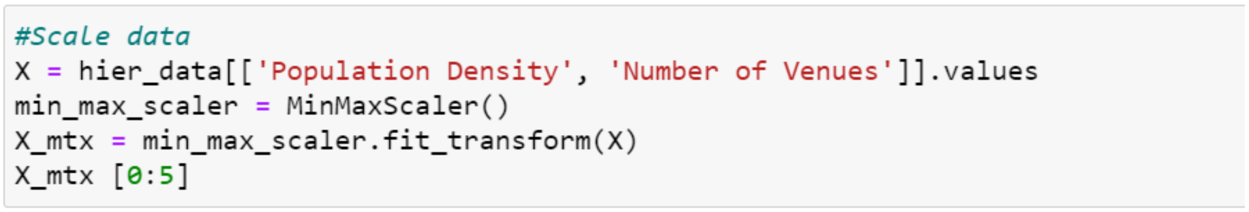


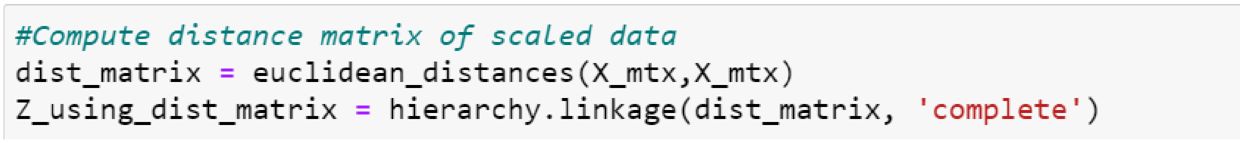
Figure 5: Population Density VS Number of Venues using K-Means Clustering Results

3.6.2 Hierarchical Model

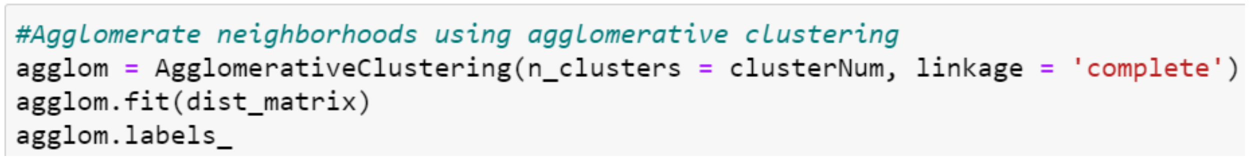
To use the Hierarchical model, the data is first scaled, using a Min Max scaler, and then fit. The distance matrix is also computed. The model is then initiated, and used to obtain a label for each neighborhood, depending on the number of clusters used. Each label is added to the data. An additional column is then added to indicate if the label of each neighborhood is similar to the label of our initial neighborhood, Côte des Neiges, to which we are comparing all others to. We also plot the hierarchical tree to visualize how the model clustering works.



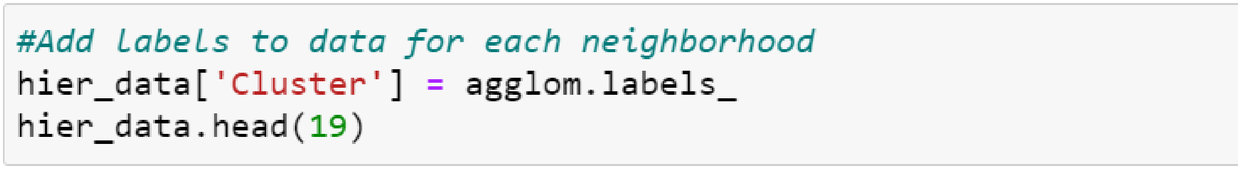
```
#Scale data
X = hier_data[['Population Density', 'Number of Venues']].values
min_max_scaler = MinMaxScaler()
X_mtx = min_max_scaler.fit_transform(X)
X_mtx [0:5]
```



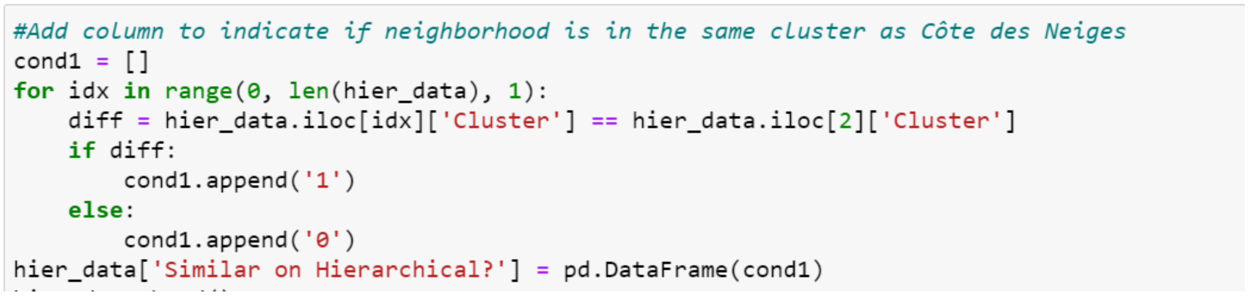
```
#Compute distance matrix of scaled data
dist_matrix = euclidean_distances(X_mtx,X_mtx)
Z_using_dist_matrix = hierarchy.linkage(dist_matrix, 'complete')
```



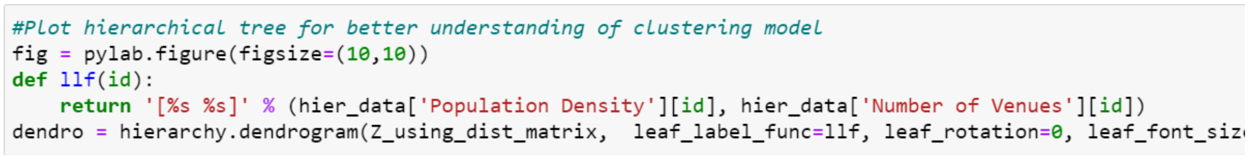
```
#Agglomerate neighborhoods using agglomerative clustering
agglom = AgglomerativeClustering(n_clusters = clusterNum, linkage = 'complete')
agglom.fit(dist_matrix)
agglom.labels_
```



```
#Add labels to data for each neighborhood
hier_data['Cluster'] = agglom.labels_
hier_data.head(19)
```



```
#Add column to indicate if neighborhood is in the same cluster as Côte des Neiges
cond1 = []
for idx in range(0, len(hier_data), 1):
    diff = hier_data.iloc[idx]['Cluster'] == hier_data.iloc[2]['Cluster']
    if diff:
        cond1.append('1')
    else:
        cond1.append('0')
hier_data['Similar on Hierarchical?'] = pd.DataFrame(cond1)
```



```
#Plot hierarchical tree for better understanding of clustering model
fig = pylab.figure(figsize=(10,10))
def llf(id):
    return '[%s %s]' % (hier_data['Population Density'][id], hier_data['Number of Venues'][id])
dendro = hierarchy.dendrogram(Z_using_dist_matrix,  leaf_label_func=llf, leaf_rotation=0, leaf_font_siz
```

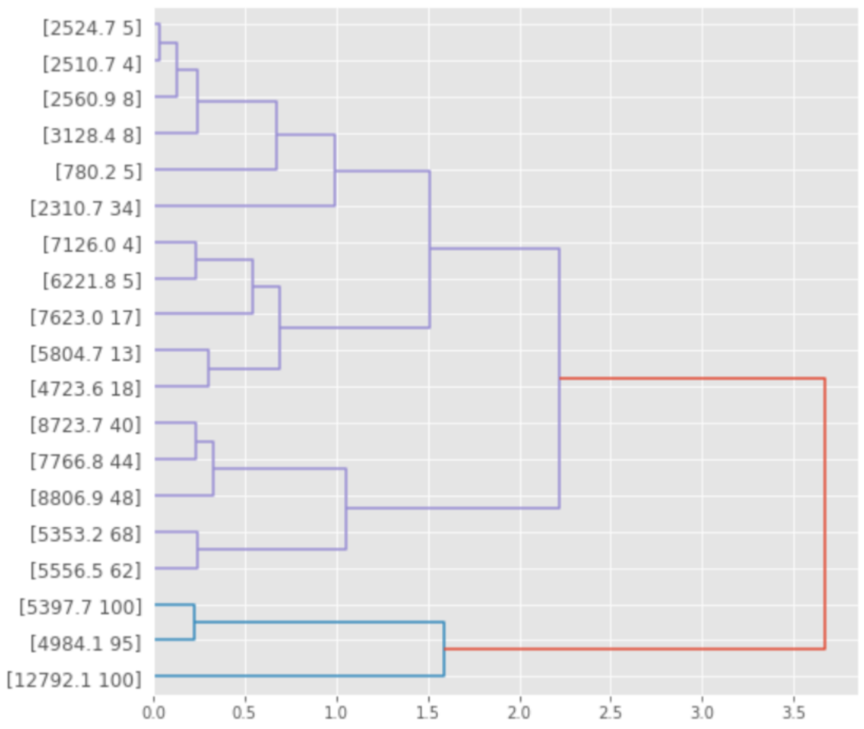
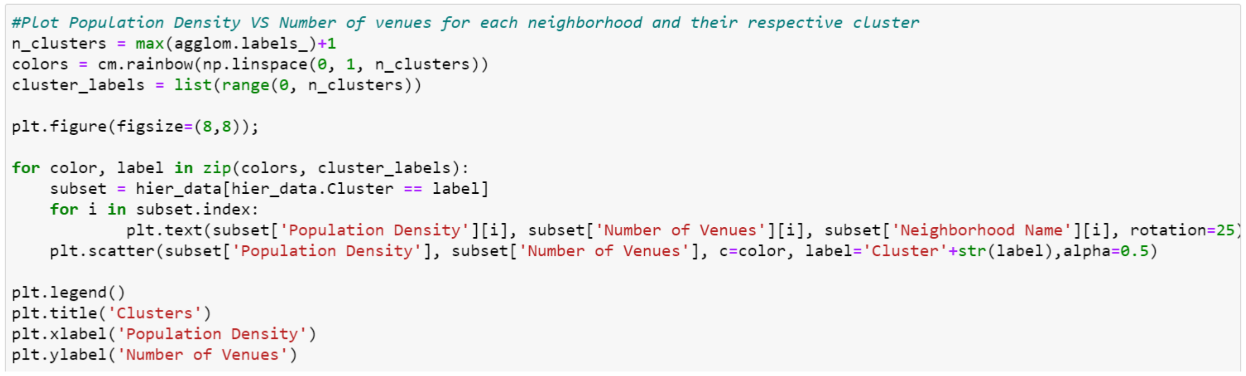


Figure 6: Hierarchical Tree with Results from the Hierarchical Model

We then plot a graph of Population Density VS Number of venues (our two main independent variables) for each neighborhood, and color their respective dots based on their cluster.



```
#Plot Population Density VS Number of venues for each neighborhood and their respective cluster
n_clusters = max(agglom.labels_)+1
colors = cm.rainbow(np.linspace(0, 1, n_clusters))
cluster_labels = list(range(0, n_clusters))

plt.figure(figsize=(8,8));

for color, label in zip(colors, cluster_labels):
    subset = hier_data[hier_data.Cluster == label]
    for i in subset.index:
            plt.text(subset['Population Density'][i], subset['Number of Venues'][i], subset['Neighborhood Name'][i], rotation=25)
    plt.scatter(subset['Population Density'], subset['Number of Venues'], c=color, label='Cluster'+str(label),alpha=0.5)

plt.legend()
plt.title('Clusters')
plt.xlabel('Population Density')
plt.ylabel('Number of Venues')
```

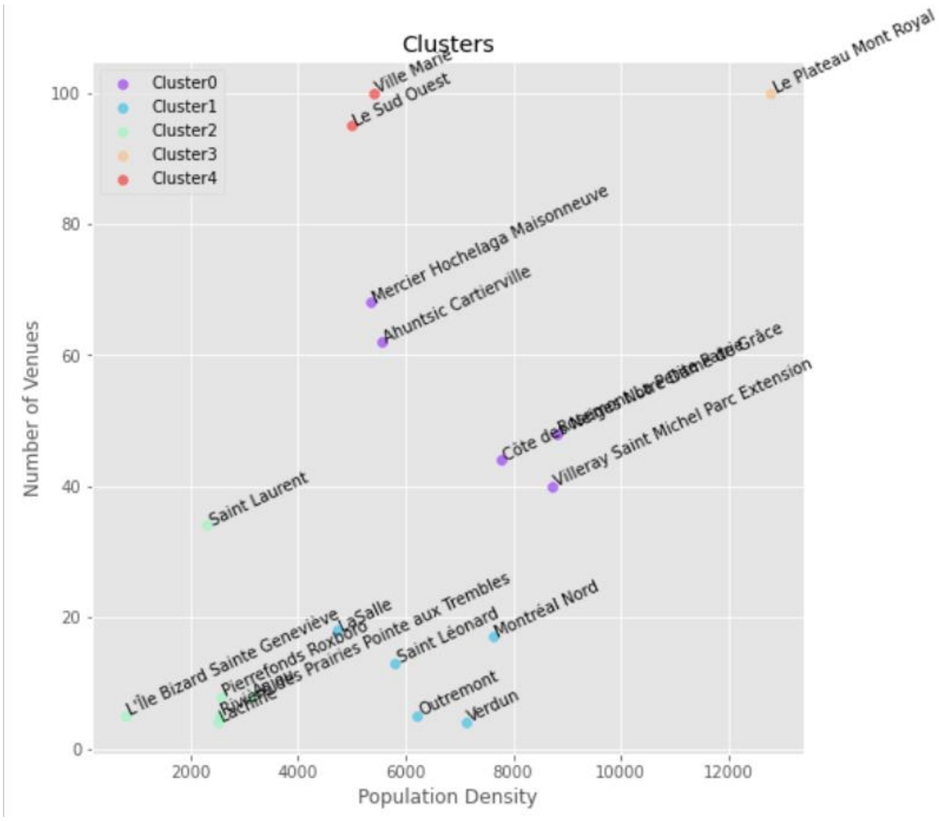
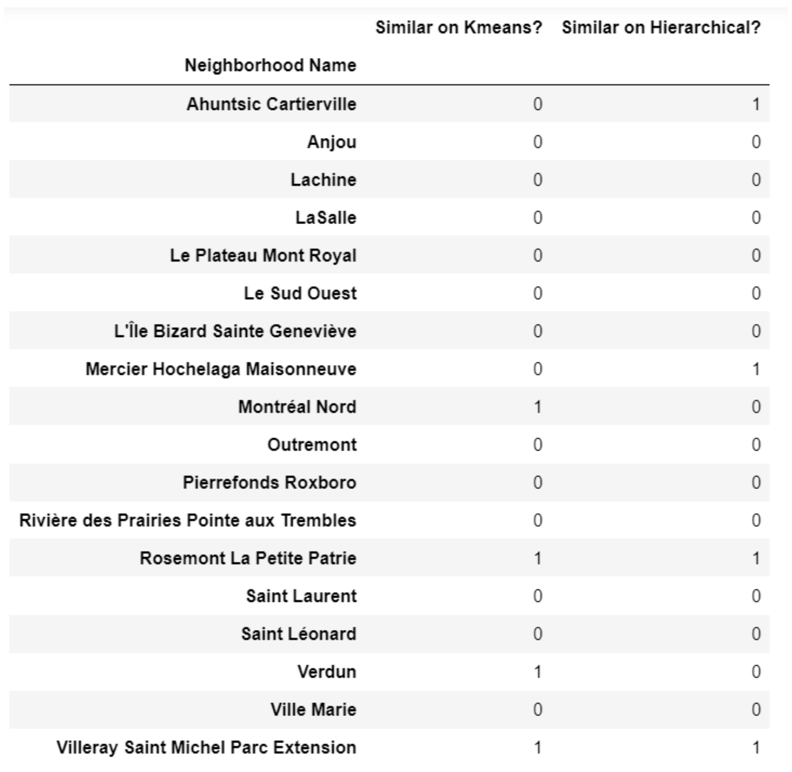


Figure 7: Population Density VS Number of Venues using Hierarchical Clustering Results

4. Results

The results obtained from both clustering models are merged in a single data frame. We drop the row representing the Côte des Neiges neighborhood, since we want to compare only other neighborhoods to it.

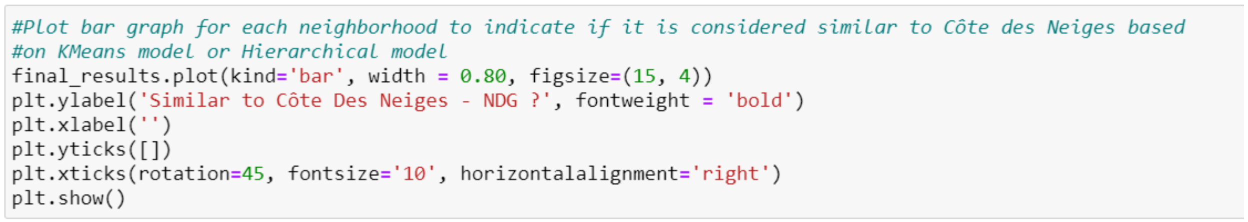
```
#Merge data from KMeans model and from Hierarchical model
results = pd.merge(kmeans_data, hier_data, on='Neighborhood Name', how='outer')
final_results = results[['Similar on Kmeans?', 'Similar on Hierarchical?']].astype(int)
final_results['Neighborhood Name'] = results['Neighborhood Name']
final_results = final_results.drop(final_results[final_results.index == 2].index)
final_results = final_results.set_index('Neighborhood Name')
final_results.head(final_results.shape[0])
```



Neighborhood Name	Similar on Kmeans?	Similar on Hierarchical?
Ahuntsic Cartierville	0	1
Anjou	0	0
Lachine	0	0
LaSalle	0	0
Le Plateau Mont Royal	0	0
Le Sud Ouest	0	0
L'Île Bizard Sainte Geneviève	0	0
Mercier Hochelaga Maisonneuve	0	1
Montréal Nord	1	0
Outremont	0	0
Pierrefonds Roxboro	0	0
Rivière des Prairies Pointe aux Trembles	0	0
Rosemont La Petite Patrie	1	1
Saint Laurent	0	0
Saint Léonard	0	0
Verdun	1	0
Ville Marie	0	0
Villeray Saint Michel Parc Extension	1	1

Figure 8: Resulting Data Frame with results from both Clustering Models

We plot a bar chart of the results. Different colors are used for each clustering model. Neighborhoods with two bars represent neighborhoods that are listed similar to our initial neighborhood from both models. The results shown in the bar graph vary as the number of clusters used changes. Using less clusters increases the number of neighborhoods similar to Côte des Neiges, while decreasing the number of clusters decreases it, as expected. The maximum number of clusters for which similar neighborhoods are found both from the K-Means model and the hierarchical model is 6. The results for other numbers are shown in the discussion section.



```
#Plot bar graph for each neighborhood to indicate if it is considered similar to Côte des Neiges based
#on KMeans model or Hierarchical model
final_results.plot(kind='bar', width = 0.80, figsize=(15, 4))
plt.ylabel('Similar to Côte Des Neiges - NDG ?', fontweight = 'bold')
plt.xlabel('')
plt.yticks([])
plt.xticks(rotation=45, fontsize='10', horizontalalignment='right')
plt.show()
```

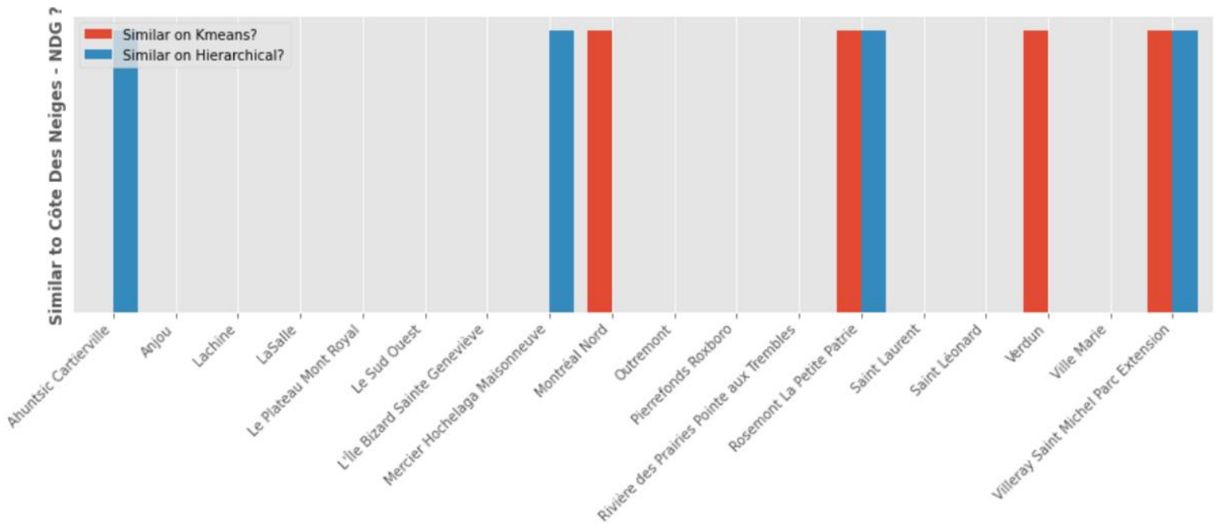


Figure 9: Results of both clustering models shown in a Bar Chart

5. Discussion and Conclusion

Similar neighborhoods to Cote des Neiges are visible in the graph above, shown in the Results section. The best neighborhoods are chosen to be the ones that are indicated similar both by the KMeans model and the Hierarchical model. Several numbers of clusters are used, to see the variation in the results.

- For 2 clusters, similar neighborhoods are: Montréal-Nord, Rosemont, Verdun and Villeray/St-Michel
- For 3, 4 and 5 clusters, similar neighborhoods are: Rosemont and Villeray/St-Michel
- For 6 or more clusters, no neighborhoods are similar on both KMeans and Hierarchical

It can be seen that, as expected, more neighborhoods are considered similar to the one we're comparing them to when less clusters are used. Increasing the number of clusters to 8 gives us only one final neighborhood. It can be concluded that Rosemont and Villeray/St-Michel are the optimal neighborhoods, since they are the most similar to Côte des Neiges, when the number of clusters is increased. They are therefore the best neighborhoods for our friend to choose from, since they are similar in terms of population density and in terms of numbers of venues, which were the two criteria he specified.

In conclusion, data science tools and methodology were used to solve a simple clustering problem, where similar neighborhoods were clustered to suggest to our friend the best neighborhood in Montreal in which to move, based on population density and venues density. Data analysis, preparation and cleaning was done in order for the required data to be easily processed by the machine learning algorithms. Two clustering models were then used to segment and cluster the neighborhoods based on two independent variables. The final results were obtained by taking the common results from the two models. Our friend now has to make a choice betwen Rosemont and Villeray/St-Michel, the two optimal neighborhoods we suggested him, and we can go on helping other friends with their problems by using the machine learning and data science tools we learned in this course!

6. References

[1] https://en.wikipedia.org/wiki/List_of_neighbourhoods_in_Montreal

[2] http://ville.montreal.qc.ca/portal/page?_pageid=6897,68149701&_dad=portal&_schema=PORTAL

[3] https://en.wikipedia.org/wiki/List_of_postal_codes_of_Canada:_H

[4] https://foursquare.com/