Applied Data-Science (Machine Learning)

Capstone Project on Neighborhoods

Submission Report

Content

1) Introduction and business problem

Objective:

The idea of this machine learning project is to create a neighborhood comparison model.

The model should support you to compare neighborhoods of different cities and find the most suitable place for yourself. So for example you are about to relocate to another city and would like to find a flat in a similar neighborhood.

In a way it is similar to a recommendation tool, which recommends similar neighborhoods in other cities.

Target Audience:

This tool can be very useful if you're moving to a different city and not sure what neighborhood to choose. So in the tool you would be able to enter the name of the neighborhood of your current city and city you would like to move or relocate. The tool will provide a list of similar neighborhoods in the new city and show the location on a map.

So the target audience can be :

1. People relocating to another city and looking for similar neighborhoods

2. Business people who are on a business trip and want to stay within a certain neighborhood for some time

3. Owners of shops that want to open a shop in another city within a special area

Success Criteria:

The objective of this project is not to develop the application itself – this could be a simple web application. The idea is rather to provide the algorithm/ model behind the application.

In order to keep it more simple we take the example, that somebody would like to move from Toronto (e.g. neighborhood : St. James Town in Toronto) to London or New York and would like to find similar neighborhoods in London or New York.

Success criteria of the project are :

- define common cluster/class values for similar neighborhoods in London / New York
- deliver optimized model for these classes
- provide a list of similar neighborhoods within the chosen cities
- show the recommended neighborhood on a map

2 Data understanding and data preparation

In order to be able to segment and compare different cities we need borough and neighborhood data from these cities as well as latitude and longitude (coordinates) of each neighborhood.

1) Extract data from the websites

We decided to compare neighborhoods between Toronto, London and New York. Therefore we need to get the relevant data for all those cities.

We get the basic neighborhood data from this websites :

https://en.wikipedia.org/wiki/List_of_areas_of_London

https://en.wikipedia.org/wiki/List_of_postal_codes_of_Canada:_M

https://geo.nyu.edu/catalog/nyu_2451_34572

The data can be extracted from the websites with the web scraping tool "Beautiful Soup".

Beautiful Soup is a Python package for parsing HTML and XML documents (incl. having malformed markup, i.e. non-closed tags, so named after tag soup). It creates a parse tree for parsed pages that can be used to extract data from HTML.

Beautiful Soup can be installed using the Python package manager pip or the anaconda package manager.

2) Data cleaning and pre-processing

Date need to be cleaned (e.g. remove whitespaces). Data cleaning will be done using Python Panda.

A big part of the pre-processing is encoding. This means representing each piece of data in a way that the computer can understand it, hence the name encode. There are different ways of encoding such as Label Encoding or One Hot Encoding. One Hot Encoding will also be used in this project.

3) Geocoding

Geocoding is needed to get latitude and longitude for each neighborhood and display these on a map.

Geocoding can be done with Python geocoder library. The geocoder will call ArcGIS World Geocoding Service which is a REST API provided by ESRI.

4) Get venue data for the neighborhoods for clustering

Foursquare API is used to get venue data for each neighborhood in London and Toronto.

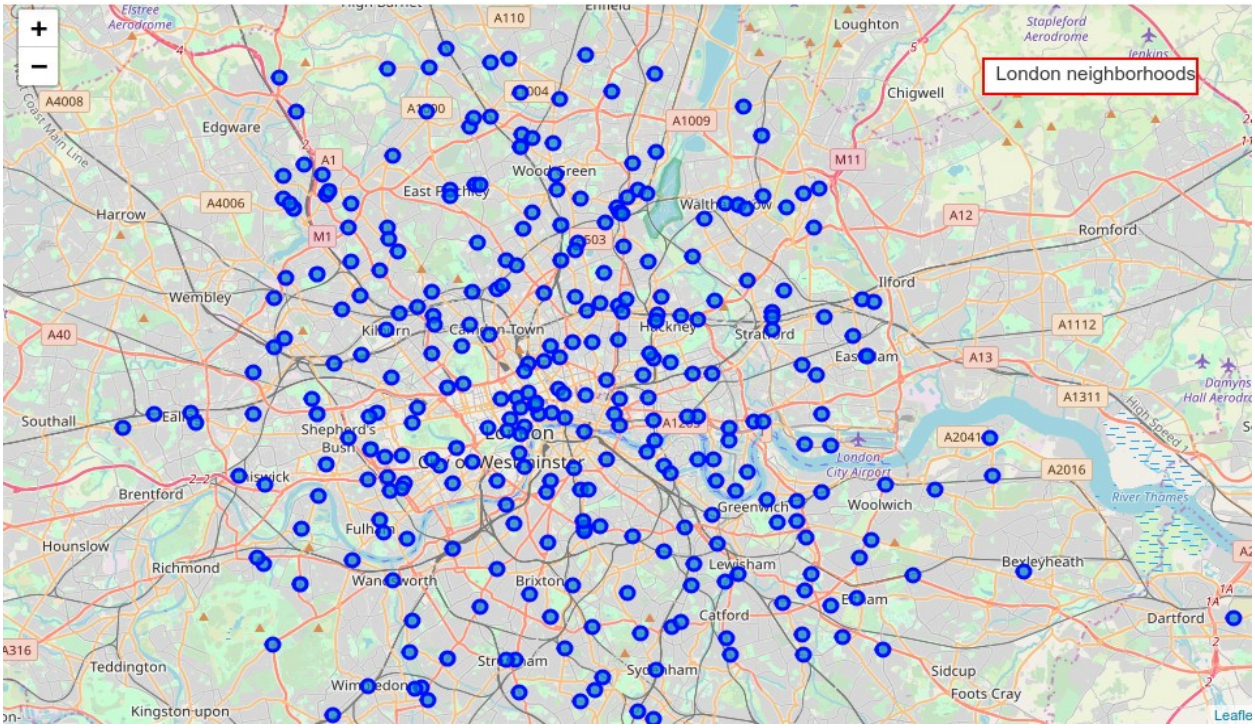
Foursquare API can give the full details about a venue including location, tips, and categories. Important for this project are mainly the categories of venues (e.g. Hotels, Bars, Coffee Shops).

For this the explore function will be used to finally get the most common venue categories in each neighborhood.

5) Data understanding

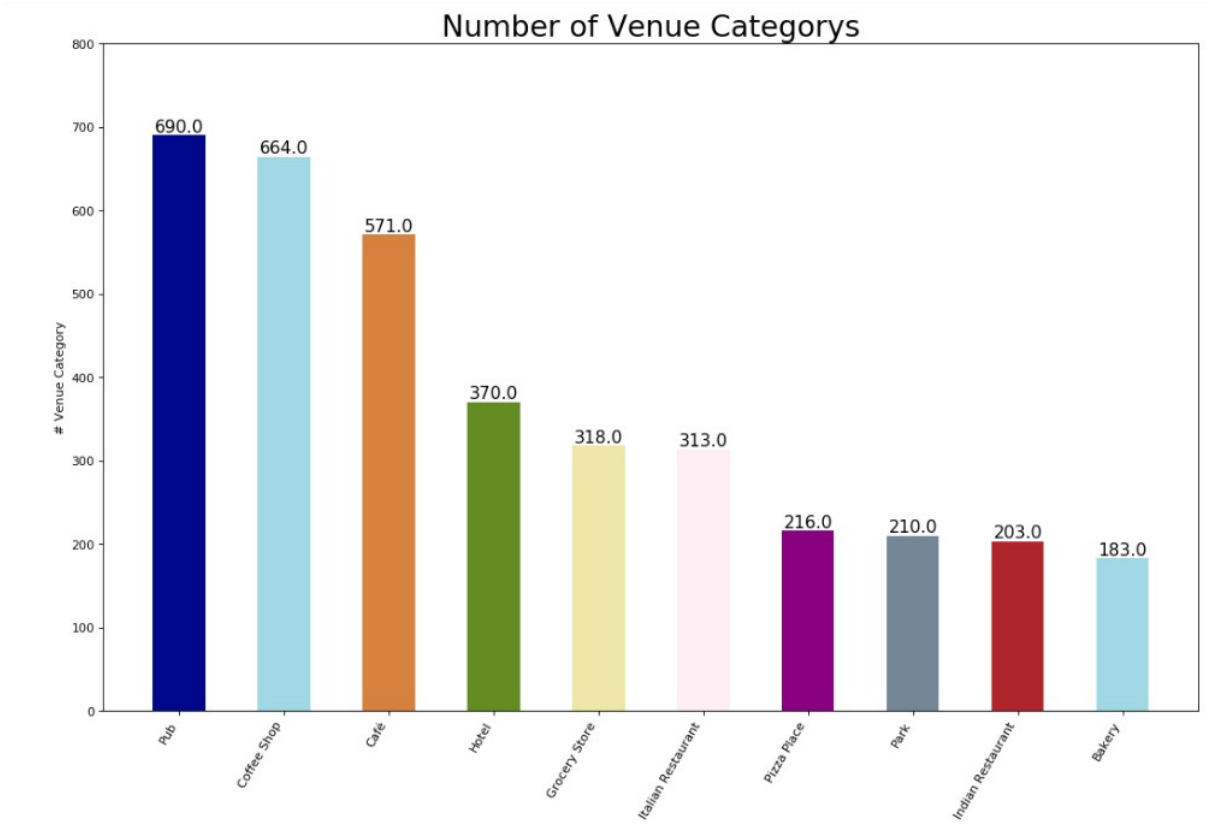
For a better understanding of data it is useful to visualize them.

For example it is useful to create a map of London with neighborhoods superimposed on top :



Also histograms are helpful for a better understanding.

The most important venues of London are shown below (according to their frequency) :



3) Modeling

We want to find neighborhoods with a similar profile in other cities.

This means we want to be able to predict the class of a neighborhood.

So in this project we first need to define the classes for all neighborhoods in London and New York.

For an initial classification we can use the venue feature from the Foursquare API, on which basis the neighborhoods where grouped into clusters.

First we need to analyze how much cluster we should have.

To deteremin the optimal value of K for the dataset we will be using the Silhouette Coefficient Method.

“The silhouette value is a measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation).

The silhouette ranges from −1 to +1, where a high value indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters” (souce:Wikipedia).

According to this method for London we should have 5 clusters :

```
In [39]: # determine the optimal value of K for dataset using the Silhouette Coefficient Method

         L_grouped_clustering = LGeo_grouped.drop('Neighborhood', 1)

         for n_cluster in range(2, 10):
             kmeans = KMeans(n_clusters=n_cluster).fit(L_grouped_clustering)
             label = kmeans.labels_
             sil_coeff = silhouette_score(L_grouped_clustering, label, metric='euclidean')
             print("For n_clusters={}, The Silhouette Coefficient is {}".format(n_cluster, sil_coeff))

         For n_clusters=2, The Silhouette Coefficient is 0.16298240547314097
         For n_clusters=3, The Silhouette Coefficient is 0.10406519487291023
         For n_clusters=4, The Silhouette Coefficient is 0.10811257833771752
         For n_clusters=5, The Silhouette Coefficient is 0.12033284442183052
         For n_clusters=6, The Silhouette Coefficient is 0.011286888354966997
         For n_clusters=7, The Silhouette Coefficient is 0.011707427557378289
         For n_clusters=8, The Silhouette Coefficient is 0.015135853339281677
         For n_clusters=9, The Silhouette Coefficient is 0.003996208706922669
```

n_clusters=5 has highest Silhouette Coefficient.

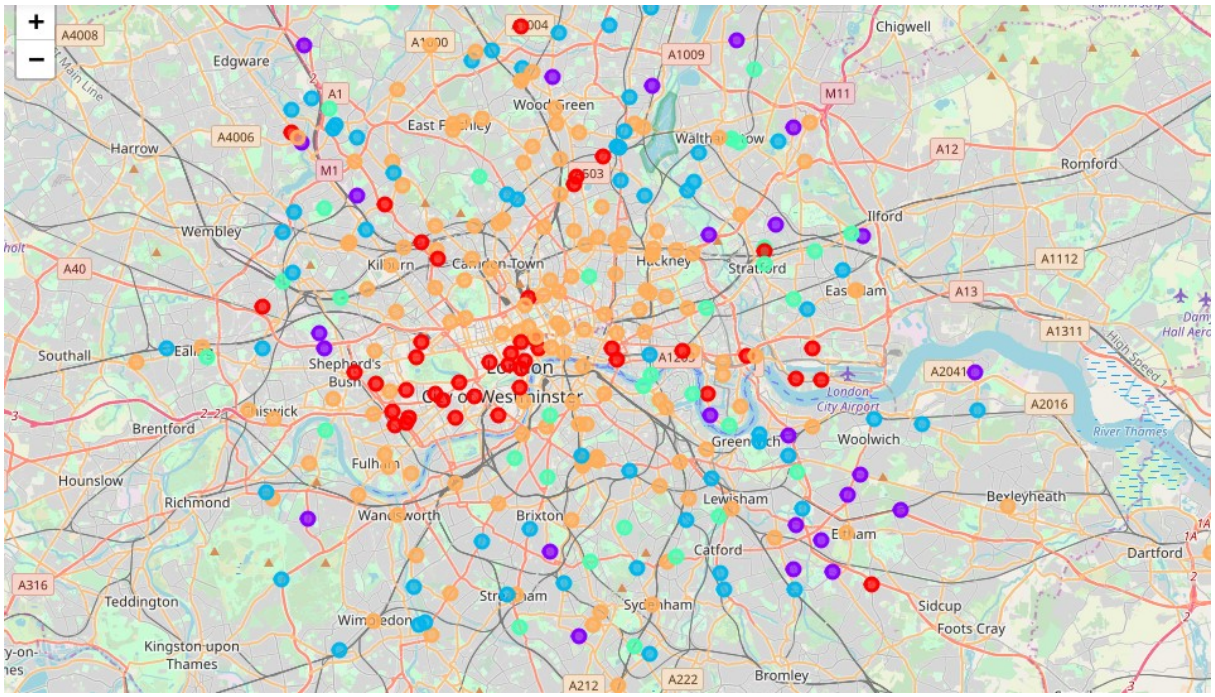
The k-means clustering algorithm is used to complete this task.

This gives us the clusters and profiles for each neighborhood.

	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	Abbey Wood	Supermarket	Coffee Shop	Convenience Store	Historic Site	Train Station	Forest	Food Truck	Ethiopian Restaurant	Event Space	Exhibit
1	Acton	Breakfast Spot	Indian Restaurant	Grocery Store	Park	Gas Station	Train Station	Convenience Store	Food Stand	Film Studio	Event Space
2	Aldgate	Hotel	Coffee Shop	Cocktail Bar	Salad Place	Restaurant	Gym / Fitness Center	Asian Restaurant	Pizza Place	Pub	Hotel Bar
3	Aldwych	Theater	Coffee Shop	Restaurant	Hotel	Dessert Shop	Cocktail Bar	Pub	History Museum	Burger Joint	Italian Restaurant
4	Anerley	Hardware Store	Gas Station	Pub	Music Store	Park	Fish & Chips Shop	Event Space	Exhibit	Falafel Restaurant	Farm

We start with the clusters of London and check the allocation of clusters.

Please find the example of clusters of London shown in the map below :



In order to be able to compare the clusters/classes between cities we need to create a model where we can enter the data from another city (e.g.Toronto) and get the adequate class in London or New York.

For this we will use k-Nearest-Neighbors (k-NN) model.

k-NN is a supervised machine learning model.

Supervised learning is when a model learns from data that is already labeled (as in our case from the clusters already available). For the labels we will use the clusters derived from the k-mean algorithm above.

A supervised learning model takes in a set of input objects and output values.

The model then train on that data to learn how to map the inputs to the desired output so it can learn to make predictions on other data.

The algorithm uses ‘feature similarity’ to predict values of any new data points. This means that the new point is assigned to a value based on how closely it resembles the points in the training set.

In general k-NN is suited for lower dimensional data. On high dimensional data (hundreds or thousands of input variables) it may not perform as well as other techniques.

k-NN can benefit from feature selection that reduces the dimensions of the input feature space.

Based on the k-nearest neighbor-method we can predict the cluster of any new neighborhood. Of course we need to first train and test the model.

Scikit-learn is a machine learning library for Python. In this project we will build a k-NN model using Scikit-learn to predict the class of a location.

Feature engineering and feature selection often provide a good return on time invested in a machine learning problem. Therefore we will do both, feature selection and feature engineering :

1) Feature selection

In order to improve accuracy of the model we will do feature selection. This is the process of choosing the most relevant features in the data. In feature selection, we remove features to help the model generalize better to new data and create a more interpretable model.

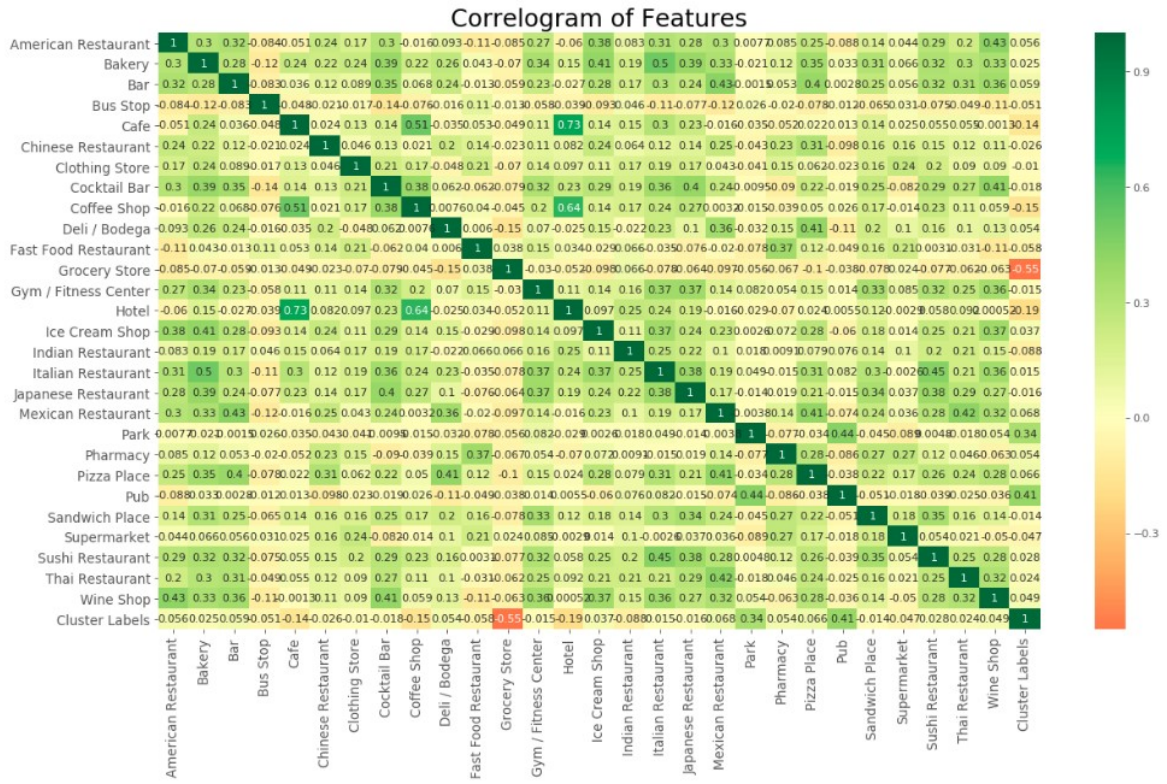
This can be easily done when features are redundant because they are highly correlated with each other.

In our case only few data are correlated (e.g. number of Hotels and Cafes).

To quantify relationships between variables, we can use the Pearson Correlation Coefficient.

This is a measure of the strength and direction of a linear relationship between two variables. A score of +1 is a perfectly linear positive relationship and a score of -1 is a perfectly negative linear relationship.

All values of the correlation coefficient of the available features are shown below:



We can see that some of the features have only a very small impact :

Train Station	-0.031855
Grocery Store	-0.026305
Asian Restaurant	-0.016174
Tapas Restaurant	-0.005104
Indian Restaurant	0.009746
Greek Restaurant	0.012736
Japanese Restaurant	0.027825

So we will be subtracting features and left with only those that are most important.

```
In [56]: #Lets define feature sets, X:
         Lfinalm.columns.tolist()

Out[56]: ['Neighborhood',
          'American Restaurant',
          'Bar',
          'Bus Stop',
          'Cafe',
          'Chinese Restaurant',
          'Clothing Store',
          'Cocktail Bar',
          'Coffee Shop',
          'Deli / Bodega',
          'Fast Food Restaurant',
          'Grocery Store',
          'Gym / Fitness Center',
          'Hotel',
          'Ice Cream Shop',
          'Indian Restaurant',
          'Mexican Restaurant',
          'Park',
          'Pharmacy',
          'Pizza Place',
          'Pub',
          'Sandwich Place',
          'Supermarket',
          'Wine Shop',
          'Cluster Labels']
```

B) Feature engineering:

This is the process of taking raw data and extracting or creating new features. Also it might mean taking transformations of variables, such as a natural log, square root or one-hot encoding categorical variables so they can be used in a model.

In this project we will normalize data. Data standardization is good practice, especially for algorithms such as KNN which is based on distance of cases.

KNN performs much better if all of the data has the same scale.

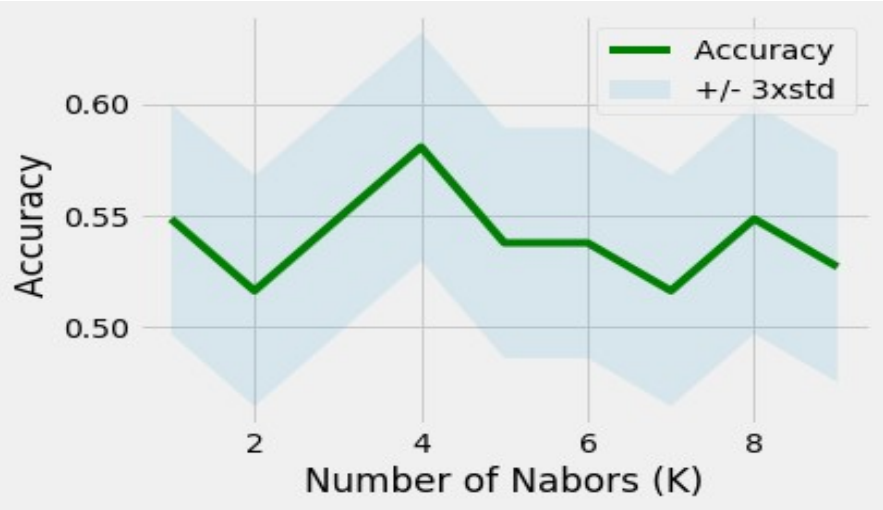
So we will be normalizing data to the range 0-1.

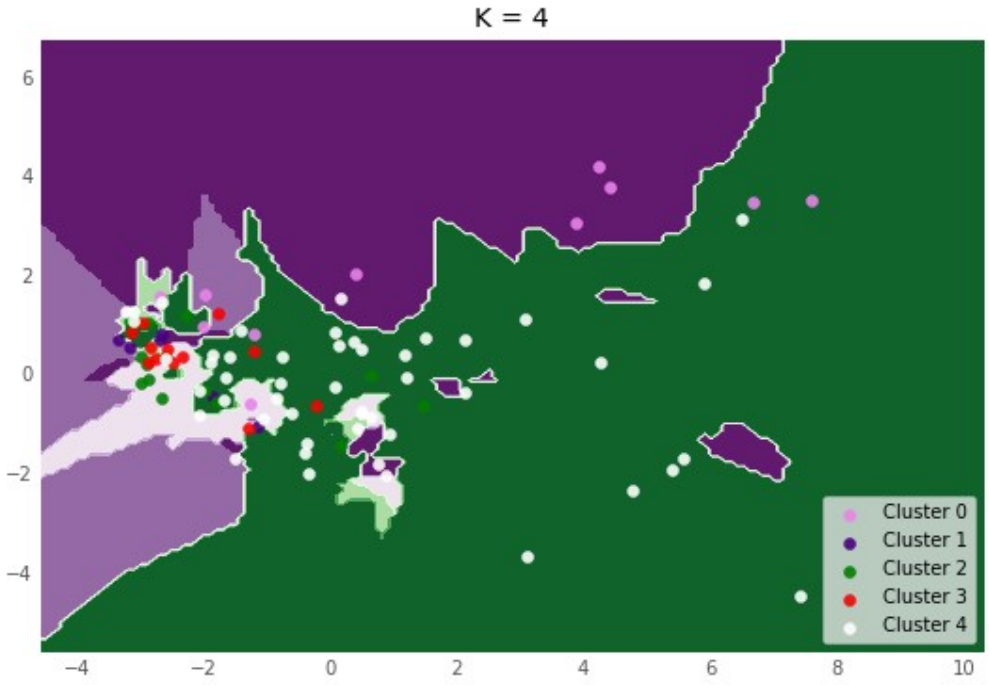
```
In [11]: # Normalize Data
         # Data Standardization give data zero mean and unit variance, it is good practice,
         # especially for algorithms such as KNN which is based on distance of cases:

         X = preprocessing.StandardScaler().fit(X).transform(X.astype(float))
         X[0:5]

Out[11]: array([[-0.454255  , -0.62466871, -0.49371472, -0.32511532, -0.48248985,
                 -0.16908747, -0.45637798, -0.26003804, -0.55653119, -0.90010015,
                 -0.50049228, -0.40253526, -0.38411249, -0.49406718,  1.05138021,
                 -0.58542845, -0.40428181, -0.39598167, -0.3673122 , -0.45397424,
                 -0.34639573, -0.30277167, -0.9531045 , -0.42851806, -0.65130109,
                 -0.50261479, -0.65354536, -0.67621363, -0.4419276 , -0.28362375,
                 -0.46161811, -0.81279167, -0.49229678, -0.73540311, -0.4169103 ,
                 -1.00246046, -0.54941141,  2.36425477, -0.40770749, -0.36234973,
                 -0.43876669,  1.89716919, -0.52147841, -0.27525397, -0.3772761 ,
                 -0.33136942],
```

Now we will determine the optimal value of K for the dataset.

First use the method for London only :

The best accuracy was 0.5806451612903226 with k= 4





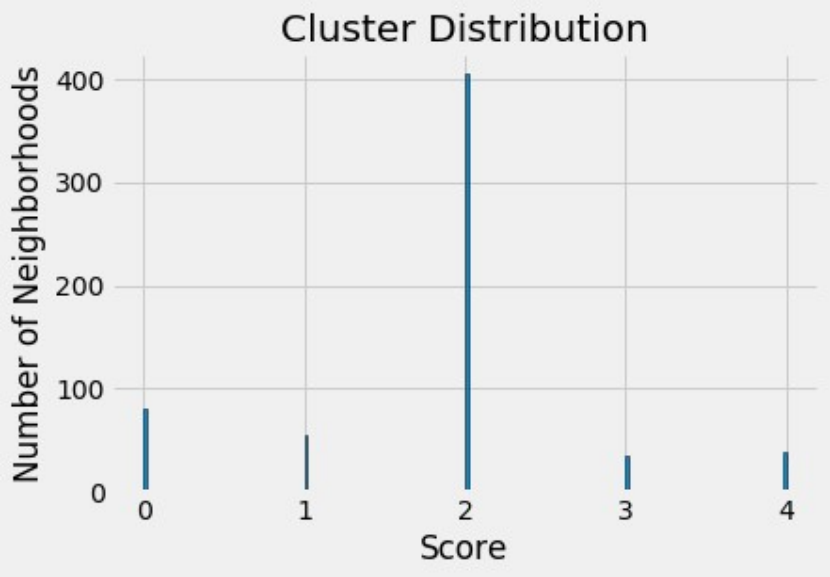
This is not really a good value. So we need to further improve the model.

First we will incorporate the data from New York. With this we will also have more training data.

But there is major difference between London and New York.

We can see that the main cluster in London is cluster 4 :

Whereas in New York the main cluster is cluster 2 :

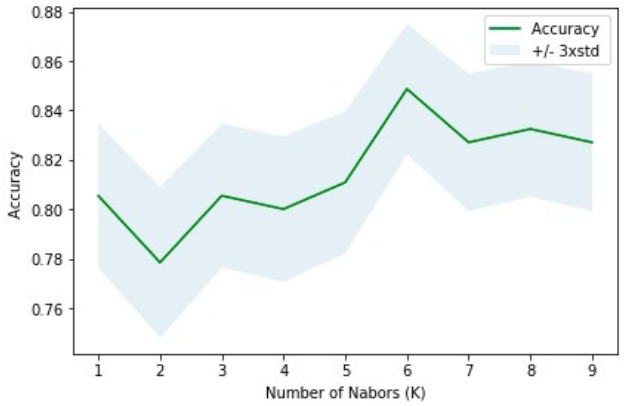


So first we need to harmonize the clusters of London and New York and have a simultaneous feature selection.

In order to further improve the model, we will do feature weighting. In feature weighting, each feature is multiplied by a weight value proportional to the ability of the feature to distinguish pattern classes.

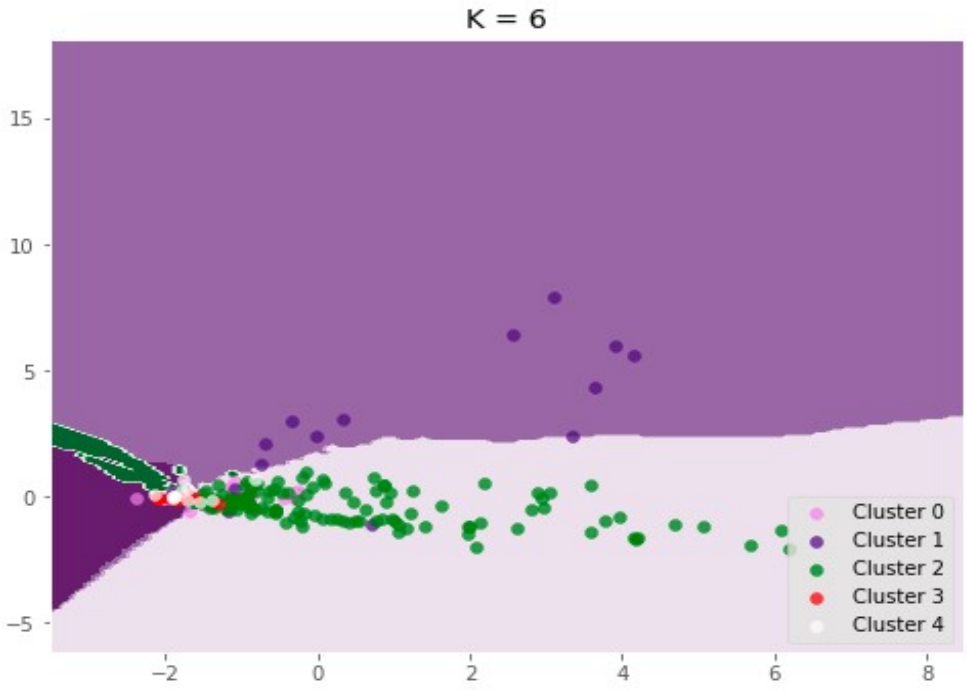
See also : https://www.jstage.jst.go.jp/article/tjsai/17/3/17_3_209/_pdf

After feature weighting accuracy has significantly improved :




```
In [18]: print( "The best accuracy was with", mean_acc.max(), "with k=", mean_acc.argmax()+1)
```

```
The best accuracy was with 0.8486486486486486 with k= 6
```



With this model we will now predict the class for our neighborhood of the “home city” - in this case St. James Town in Toronto.

The prediction is class 1

In [96]:
```
# make a prediction for an out-of-sample observation

predicted= neigh.predict([[0,  0,  0, 18,  0,  2,  3, 18,  0,  1,  1,  0, 15,  0,  1,  0,  2,
        0,  2,  0,  0,  0,  0]])


print("St. James Town in Toronto is in Cluster"), print(predicted)
```

```
St. James Town in Toronto is in Cluster
[1]
```

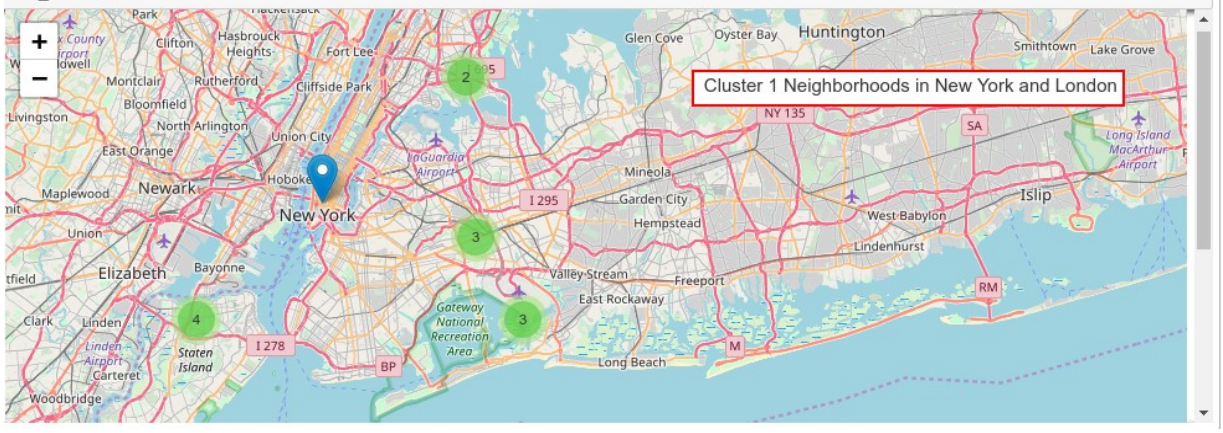
Now we will select all neighborhoods in New York and London with the same value (=1).

The result can also be shown on a folium map with drill-down functionality :

Drill down to city of London

Drill down to city of New York :

```
MarkerCluster(data).add_to(map_clusters).get_root().html.add_child(folium.Element(legend_html))

map_clusters
```

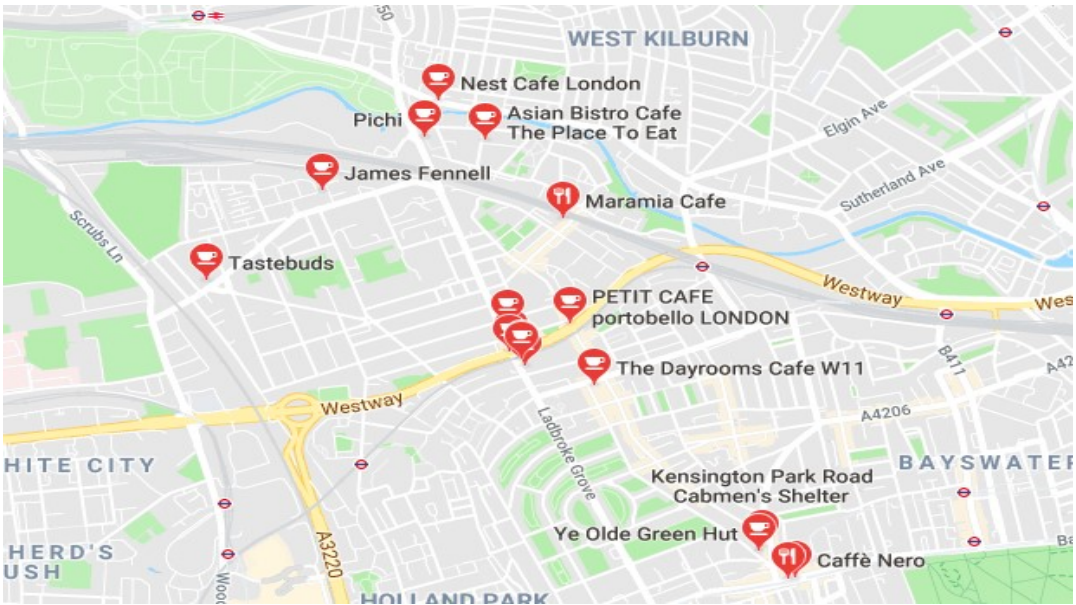


4 Results

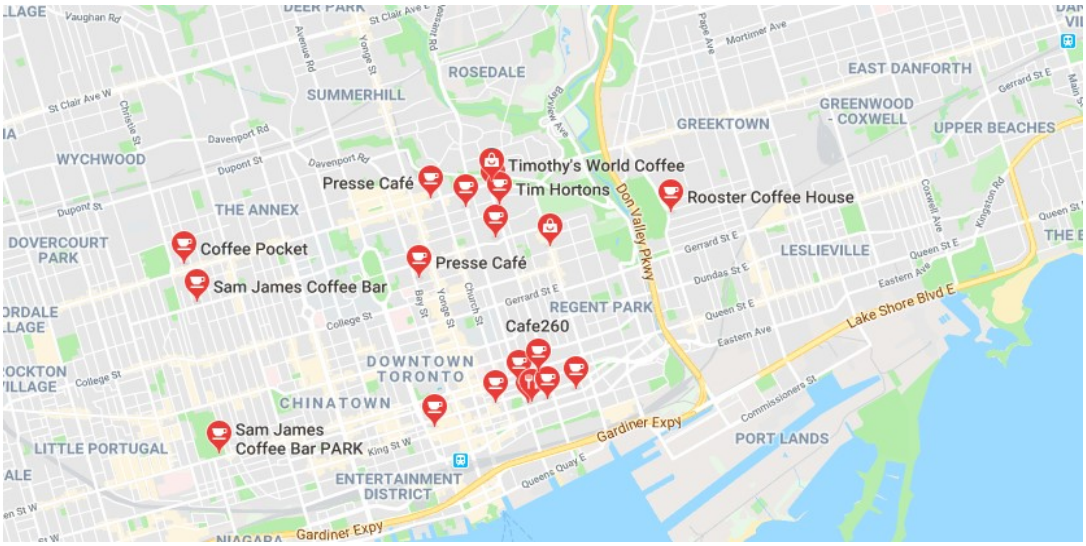
If we look at the results the neighborhoods look quite similar. An important category in this class is for example cafes.

We can also cross-check this in Google maps - there are a lot of cafes

and also in North Kensington in London :



and in St. James Town in Toronto



So overall in this case it is quite a good result and shows the neighborhoods are really similar with regards to the locations / location categories.

5 Discussion

As we saw it is important to have a structured approach in this kind or projects. A recommendation would be to us the CRISP-DM phases (Cross-Industry Process for Data Mining). The phases defined in this approach are :

1) Business understanding
2) Data understanding
3) Data preparation
4) Modeling
5) Evaluation
6) Deployment

Very important steps are exploring data to gain a better understanding of the specific characteristics of the data. For example we have discovered during the exploring data step the correlation of features and that some features can be removed without negatively affecting the analysis results.

Visualization techniques provide a fast and effective way to look at data in this step.

6 Conclusion

We have solved the problem with an classification approach. To be more specific : with a multi-class classification approach as the target variable (class) has more than two possible values. Multi-class classification is also referred to as multi-label classification. We have build and adjusted a model with the k□nearest neighbors algorithm and in this case it worked well as this is a simple yet effective data mining technique especially for small data sets.