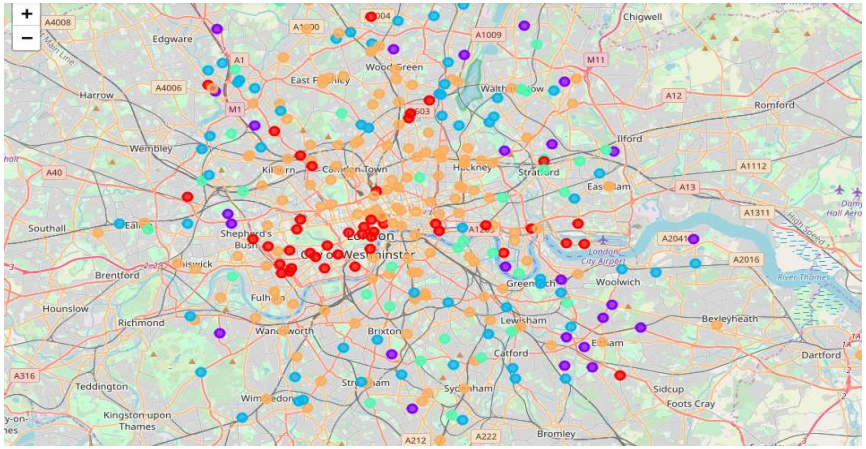
Applied Data-Science (Machine Learning) Capstone Project on Neighborhoods Submission Report

Marcus Kirchherr – September 2019



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.

Target Audience:

- The model can be very useful if you're moving to a different city and not sure what neighborhood to choose.
- 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
    3. Owners of shops that want to open a shop in another city within a similar neighborhood to the city the live in

2 Data understanding and data preparation

Extract data from the websites

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".

```
# specify which URL/web page to be scraping
url = "https://en.wikipedia.org/wiki/List_of_areas_of_London"

# open the url using urllib.request and put the HTML into the page variable
page = urllib.request.urlopen(url)

# import the BeautifulSoup library to parse HTML and XML documents
from bs4 import BeautifulSoup

# parse the HTML from our URL into the BeautifulSoup parse tree format
soup = BeautifulSoup(page, "lxml")
```

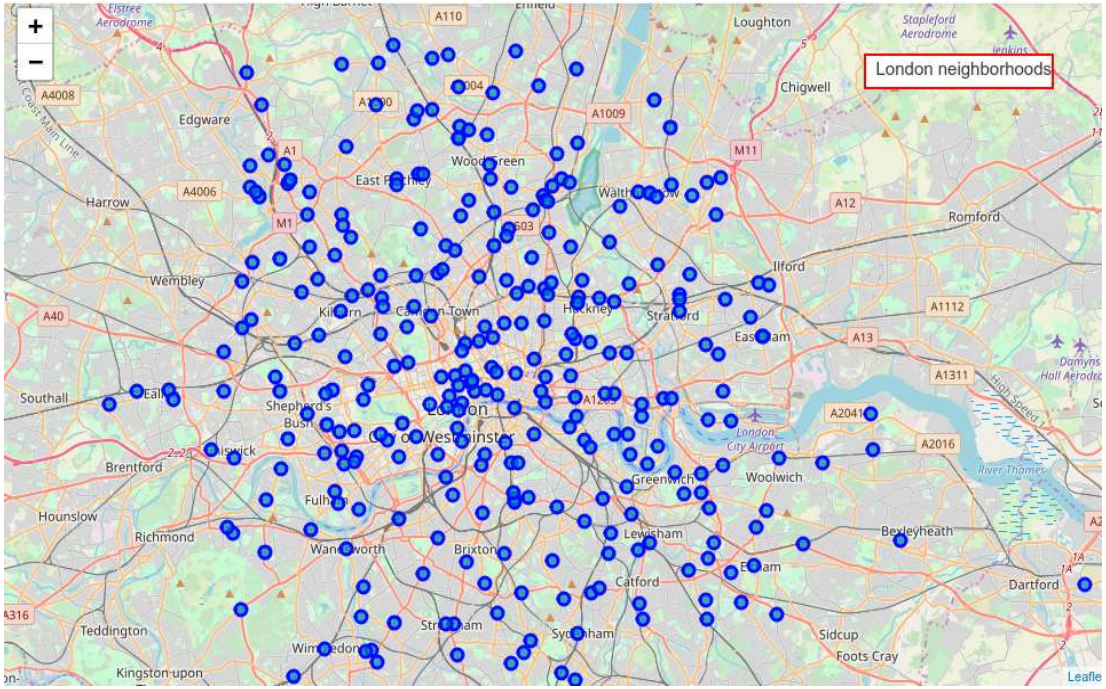
2 Data understanding and data preparation

Geocoding

- Geocoding is needed to get latitude and longitude for each neighborhood and can be done with Python geocoder library.
- The geocoder will call ArcGIS Geocoding Service which is a REST API provided by ESRI.

Create a map

- Folium is a powerful data visualization library in Python that was built primarily to visualize geospatial data



2 Data understanding and data preparation

Get venue data for the neighborhoods for clustering

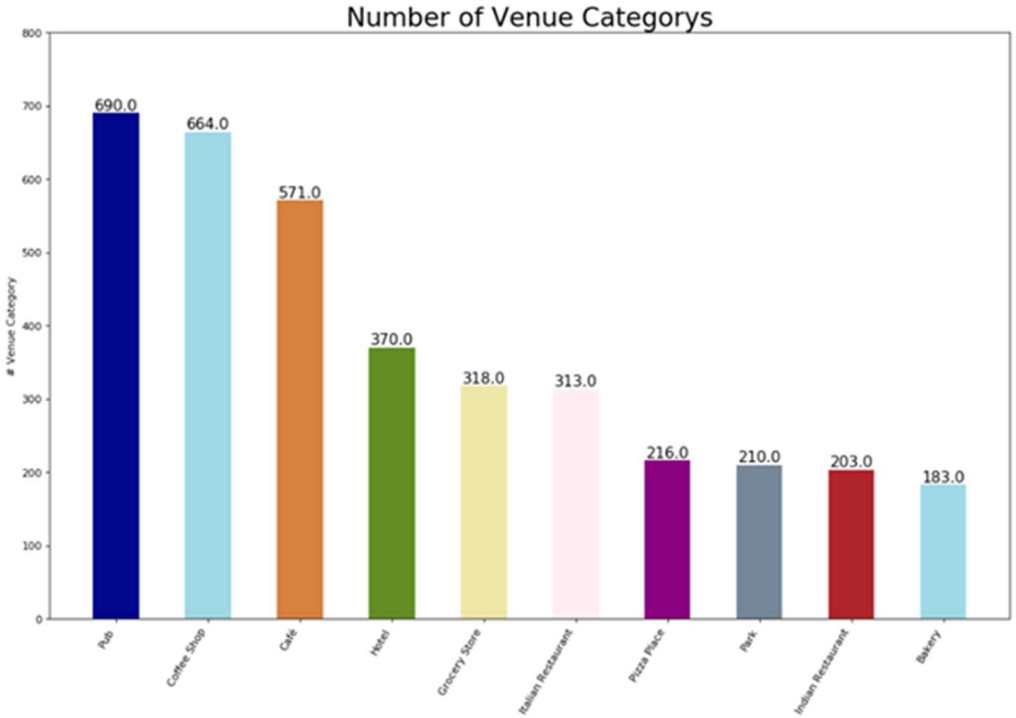
Foursquare API is used to get venue data for.

https://developer.foursquare.com/docs/api/venues/details

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.

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



3) Modeling

Neighborhood Clustering

- For an initial classification we can use the venue feature from the Foursquare API, on which basis the neighborhoods where grouped into clusters.
- Here the **k-means clustering algorithm** is used - to determine the optimal value of K for the dataset we will be using the Silhouette Coefficient Method.

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
```

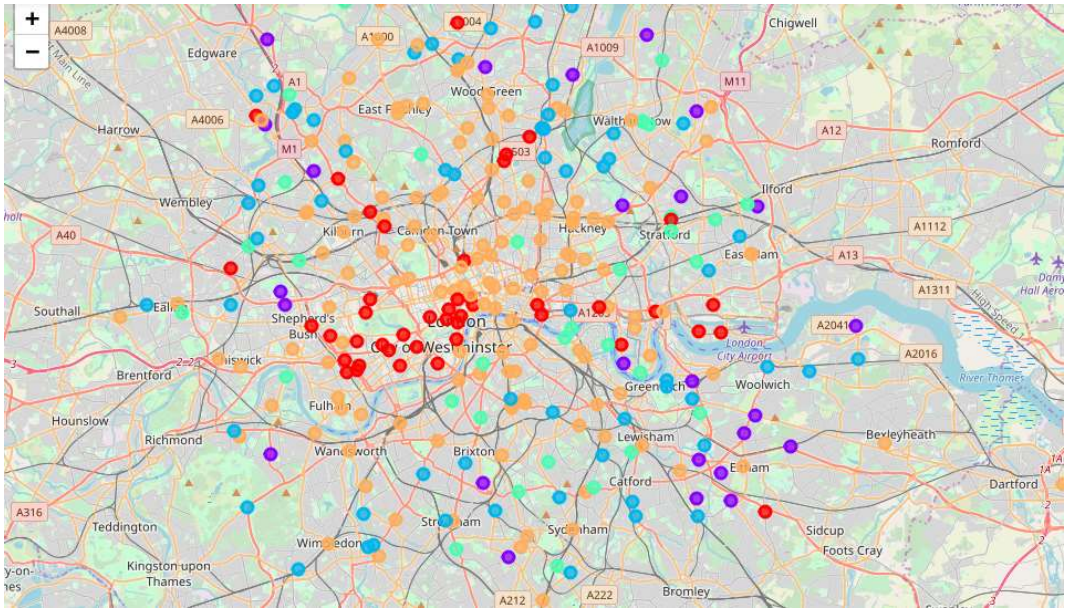
n_clusters=5 has highest Silhouette Coefficient.

3) Modeling

Neighborhood Clustering

- 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



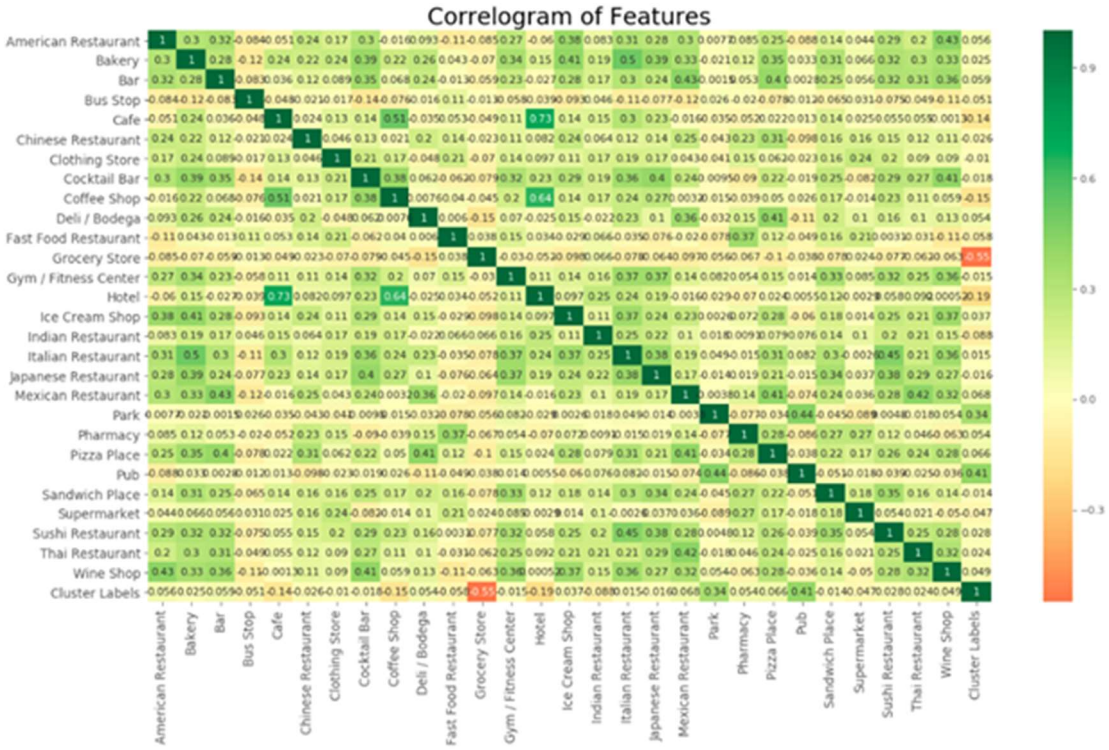
3) Modeling

k-Nearest-Neighbors (k-NN) model

- 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.

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



3) Modeling

Feature engineering:

- This is the process of taking raw data and extracting or creating new features.
- 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],
```

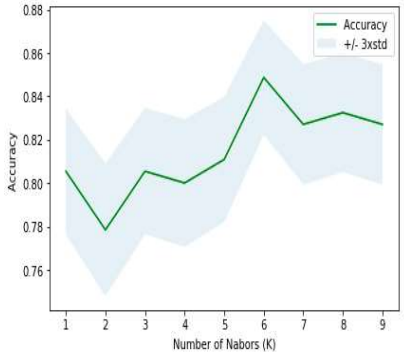
3) Modeling

Feature weighting

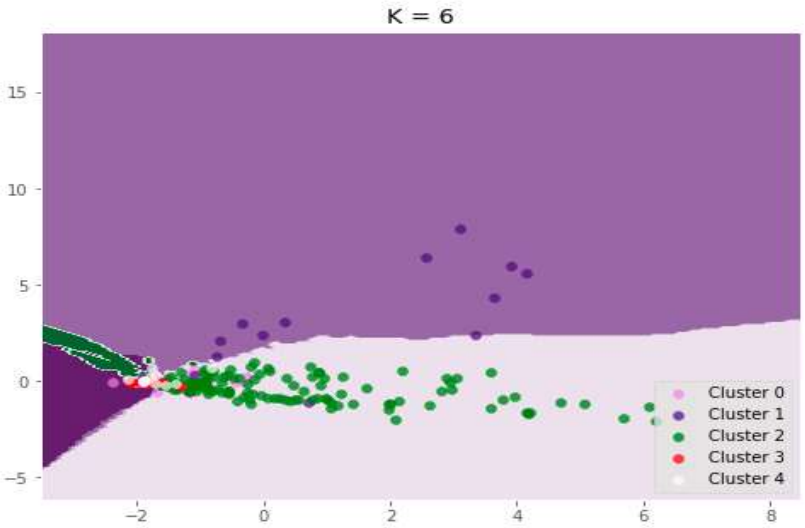
Now we will determine the optimal value of K for the dataset.

- 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.
- 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:
- 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
```



4) Results

Prediction

- 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).

4) Results

- Neighborhoods in New York and London within the same cluster (value =1).

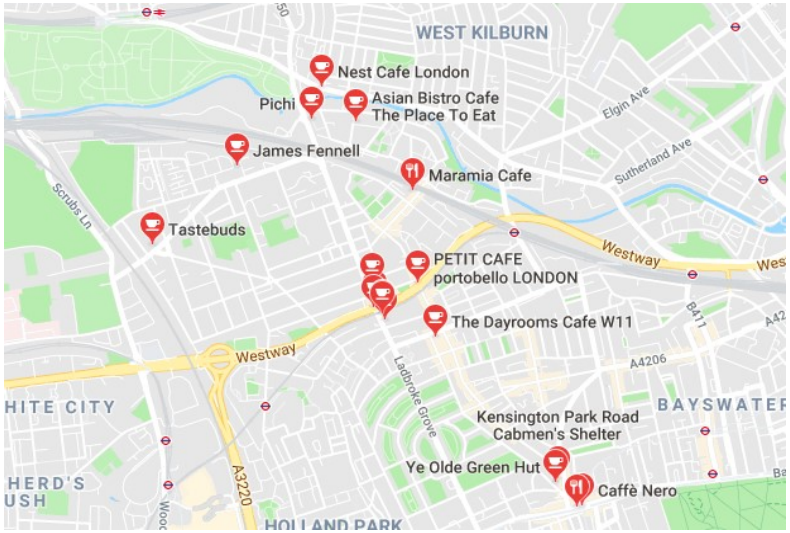
- City of London

- City of New York

4) Results

- It is quite a good result and shows the neighborhoods are really similar with regards to the locations / location categories. One important category in this cluster are cafes. This can also be seen in Google Maps:

Cafes in North Kensington in London



and in St. James Town in Toronto