IBM Data Science Capstone Report
"The Big Apple vs Hollywood"

October 17, 2020

The Big Apple vs Hollywood

Written by A.ABDELSALAM

1 Introduction

This report could have so many names, “The class of the titans”, “East vs West”, “El classico”, but the fact is this fight over which of New York city or Los Angeles is the heart of the United States has been going on, and will most likely go on, for many years. Any of these cities alone has a tremendous influence on the rest of the world, maybe more so than any other city on this planet. From the huge impact that Wall Street has on world economics to the influence of Hollywood movies on the everyday lives of each of us, we could speak hours on end about them.

Aside from this, New York city and Los Angeles remain on the top of the lists of tourists wanting to visit for the first time the US, or for the regular tourists wanting to spend the holidays. Indeed, both are diverse and multicultural and offer a wide palette of experiences that is widely sought after by tourists. In the project we will try to group the neighbourhoods of NYC and LA respectively and draw conclusions on what they both have to offer and if one city stands out more than the other.

2 Business Problem

From a business standpoint the aim is to help tourists in choosing their destinations depending on the type of experience proposed by one or the other city. This could also help them not only decide which city to go to but what neighborhood to stay in. This could also be useful for the

locals, looking to move from one city to the other, or even from any city to those two. Our analysis will hoefully provide them with a better breakdown of neighborhood compositions, in terms of culture, cuisine and facilitations. Finally, this could be the start of a more serious project for buisness owners looking to open up or branch their business.

3 Data Description

We will require geographical location data for both New York City and Los Angeles. The major difference is that New York city is split into boroughs and Los Angeles isn't but that should not be a problem, we will ignore the Boroughs and focus only on all the neighborhoods within New York city rather than focus on one Borough. We will essentially need a dataset that contains all the neighborhoods that exist in each city as well as the the latitude and logitude coordinates of each neighborhood.

3.1 New York city

To retrieve our data we will download it for free from this url : https://geo.nyu.edu/catalog/nyu_2451_34572

The JSON file contains a lot of unecessary information and will need to be wrangled appropriately in order to extract the pertinent data. The dataset has information on the Borough as well as Neighborhood, as well as the latitude and longitude for all the neighborhoods, we will limit ouselves to only: 1. *['features]['properties']['name']* : Name of Neighborhoods in NYC 2. *['features]['geometry']['coordinates']* : which is a vector that contains the Latitude and Longitude of the Neighborhoods

3.2 Los Angeles

Similarily, we will download a csv readily available for free at this url : https://usc.data.socrata.com/dataset/Los-Angeles-Neighborhood-Map/r8qd-yxsr

The CSV has many unecessary information but we will only retain the information we need about the neighborhoods of Los Angeles county:

1. *name* : Name of Neighbourhoods in LA
2. *latitude* : Latitude of Neighbourhoods in LA
3. *longitude* : Longitude of Neighbourhoods in LA

3.3 Foursquare API Data

We will need data about different venues in the different neighbourhoods. To get that that information we will leverage the Foursquare API locational information. Foursquare is a location data provider containing information about all venues and events within an area of interest. Those information can be in the form of venue names, locations, menus, customer ratings and comments and even pictures of the venue. This makes our job easier as we can find all the information centralized in one place. Consequently we will use the foursquare location platform as our only data source since all the required information can be obtained through their API directly from the notebook.

Once we have established a list of neighbourhoods, we can then connect to the Foursquare API to query for information about venues inside each and every neighbourhood. The data returned from Foursquare query contains information of venues within a specified distance of the longitude and latitude of each neighborhood. The information foe each venue is organized as follows:

1. *Neighbourhood* : Name of the Neighbourhood
2. *Neighbourhood Latitude* : Latitude of the Neighbourhood
3. *Neighbourhood Longitude* : Longitude of the Neighbourhood
4. *Venue* : Name of the Venue
5. *Venue Latitude* : Latitude of Venue
6. *Venue Longitude* : Longitude of Venue
7. *Venue Category* : Category of Venue

For each neighbourhood, we have chosen the radius to be 500 meters and we have limited the number of returned results to 100. Indeed, with a personal free account we are limited to 99500 basic calls a day and 500 premium. This is something to take into condiseration as we will not be able to get ALL venues, and as such our data and thefore analysis will be slightly biased.

Once the data for the venues of each neighborhood in NYC and LA have been gathered we will build our cluser model to group the neghborhoods according to their similarities. The rest of the report will over the methodology, data collection, wrangling, visualization and model construction. We will present and discuss our results and finish off with conclusions.

4 Methodology

Since we will be working from within the notebook we need to set up our environment my loading all the necessary packages as follows

[ ]:
```
import numpy as np # library to handle data in a vectorized manner

import pandas as pd # library for data analsysis
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)

import json # library to handle JSON files

#!conda install -c conda-forge geopy --yes # installs geopy if not already␣
 ↪installed
from geopy.geocoders import Nominatim # convert an address into latitude and␣
 ↪longitude values

import requests # library to handle requests
from pandas.io.json import json_normalize # tranform JSON file into a pandas␣
 ↪dataframe

# Matplotlib and associated plotting modules
import matplotlib.cm as cm
import matplotlib.colors as colors
```

```
# import k-means from clustering stage
from sklearn.cluster import KMeans

#!conda install -c conda-forge folium=0.5.0 --yes # uncomment this line if you␣
 ↪haven't completed the Foursquare API lab
import folium # map rendering library

print('Libraries imported.')
```

4.1 Data fetching

The entry point of the whole project is getting the data we need from somewhere. As mentionned we will need data for both New York city and Los Angeles: those datasets need to contain a minimum of neighborhood names, which is really the bare minimum, as in fact with only this information we would be able to extract the latitudes and longitudes using other tools within Python. However it implies more work, so it is best if we initially try to get our hands on the most complete dataset in order to facilitate our coding work.

For New York we will get the data in JSON format from the following URL https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DS0701EN-SkillsNetwork/labs/newyork_data.json as shown below. It's worth mentionning that this is not the only method to handle JSON files, in fact you could directly read it into a Pandas frame, however with such dense files, you have little control over what Pandas does. In the method below we take it step by step, we take a general look at the JSON result and look for the information we need and then only put those in a dataframe.

```
# New York data
!curl -o newyork_data.json https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IB
print('Data downloaded!')

with open('newyork_data.json') as json_data:
    newyork_data = json.load(json_data)
```

This is what the raw data looks like :

```
[6]: newyork_data
```

```
[6]: {'type': 'FeatureCollection',
      'totalFeatures': 306,
      'features': [{'type': 'Feature',
        'id': 'nyu_2451_34572.1',
        'geometry': {'type': 'Point',
         'coordinates': [-73.84720052054902, 40.89470517661]},
        'geometry_name': 'geom',
        'properties': {'name': 'Wakefield',
         'stacked': 1,
         'annoline1': 'Wakefield',
         'annoline2': None,
         'annoline3': None,
         'annoangle': 0.0,
         'borough': 'Bronx',
         'bbox': [-73.84720052054902,
          40.89470517661,
          -73.84720052054902,
          40.89470517661]}},
       {'type': 'Feature',
        'id': 'nyu_2451_34572.2',
        'geometry': {'type': 'Point',
         'coordinates': [-73.82993910812398, 40.87429419303012]},
        'geometry_name': 'geom',
        'properties': {'name': 'Co-op City',
```

For Los Angeles we can manually download the CSV file and place it in our working folder. Then we can import it directly into a datafram as its structure is fairly simple.

```
losangeles_data = pd.read_csv('la_neighborhoods.csv')
losangeles_data.head()
```

This what the imported data looks like in a dataframe:

```
[55]: losangeles_data = pd.read_csv('la_neighborhoods.csv')
      losangeles_data.head()
```

[55]:

	set	slug	the_geom	kind	external_i	name	display_na	sqmi	type	r
0	L.A. County Neighborhoods (Current)	acton	MULTIPOLYGON (((-118.20261747920541 34.5389897...	L.A. County Neighborhood (Current)	acton	Acton	Acton L.A. County Neighborhood (Current)	39.339109	unincorporated-area	
1	L.A. County Neighborhoods (Current)	adams-normandie	MULTIPOLYGON (((-118.30900800000012 34.0374109...	L.A. County Neighborhood (Current)	adams-normandie	Adams-Normandie	Adams-Normandie L.A. County Neighborhood (Curr...	0.805350	segment-of-a-city	
2	L.A. County Neighborhoods (Current)	agoura-hills	MULTIPOLYGON (((-118.76192500000009 34.1682029...	L.A. County Neighborhood (Current)	agoura-hills	Agoura Hills	Agoura Hills L.A. County Neighborhood (Current)	8.146760	standalone-city	
3	L.A. County Neighborhoods (Current)	agua-dulce	MULTIPOLYGON (((-118.2546773959221 34.55830403...	L.A. County Neighborhood (Current)	agua-dulce	Agua Dulce	Agua Dulce L.A. County Neighborhood (Current)	31.462632	unincorporated-area	
4	L.A. County Neighborhoods (Current)	alhambra	MULTIPOLYGON (((-118.12174700000014 34.1050399...	L.A. County Neighborhood (Current)	alhambra	Alhambra	Alhambra L.A. County Neighborhood (Current)	7.623814	standalone-city	

4.2 Data pre-processing

Once we have imported the datasets we need to do some clean-up, also known as data wrangling in the data science world. This step is crucial inevitable if we want the rest of the project to be painless. This step can also be longer in some cases.

4.2.1 New York city

For New York, we notice that all the information we need is stored under the *features* key, which is basically a list of all the neighborhoods. Hence, we will define a new variable to extract allt the *features*

```
ny_neighborhoods_data = newyork_data['features']
```

We will then loop over all the *features*, extract the name of the neighborhoods and their locations, and store them into a dataframe that we will initialize before hand with the appropriate column names. It goes as folows:

```
# define the dataframe columns
column_names = ['Neighborhood', 'Latitude', 'Longitude']

# instantiate the dataframe
ny_neighborhoods = pd.DataFrame(columns=column_names)

# loop over all the features key and store the relevant data in the df
for data in ny_neighborhoods_data:
    neighborhood_name = data['properties']['name']

    neighborhood_latlon = data['geometry']['coordinates']
```

```
    neighborhood_lat = neighborhood_latlon[1]
    neighborhood_lon = neighborhood_latlon[0]

    ny_neighborhoods = ny_neighborhoods.append({
                                            'Neighborhood': neighborhood_name,
                                            'Latitude': neighborhood_lat,
                                            'Longitude': neighborhood_lon}, ignore_index=True)
```

The result is the following dataframe: much cleaner and easier to read !

	Neighborhood	Latitude	Longitude
0	Wakefield	40.894705	-73.847201
1	Co-op City	40.874294	-73.829939
2	Eastchester	40.887556	-73.827806
3	Fieldston	40.895437	-73.905643
4	Riverdale	40.890834	-73.912585

Furthermore we can drop dupicates from the dataframe and return the number of total neighborhoods. It also very important at this early stage that we make sure there are not any null values in the latitudes and longitudes.

```
# Getting rid of duplicates
ny_neighborhoods = ny_neighborhoods.drop_duplicates(subset=['Neighborhood'])
print('The dataframe has {} neighborhoods.'.format(
        len(ny_neighborhoods['Neighborhood'].unique()),
    )
)

# Making sure there are no null entries
print('There are {} and {} null entries in Latitude and Longitude'.format(ny_neighborhoods['Lati
```

4.2.2 Los Angeles

For Los Angeles we have already imported the dataset into a dataframe however there are plenty of useless information that we need to get rid of and lighten up the dataframe.

We will select only the columns of interest to us and rename them so they match what wehave previously done with New York.

We will slect only the columns with the name of the neighborhoods, their latitudes and their longitudes. We will also drop duplicates and null entries if there are any.

```
columns_of_interest = ['name','longitude','latitude']
la_neigborhoods = losangeles_data[columns_of_interest]
```

```
la_neigborhoods.columns=['Neighborhood','Latitude','Longitude']
# Droping duplicates
la_neigborhoods = la_neigborhoods.drop_duplicates(subset=['Neighborhood'])

# Checking for NaN
la_neigborhoods.isnull().sum()
```

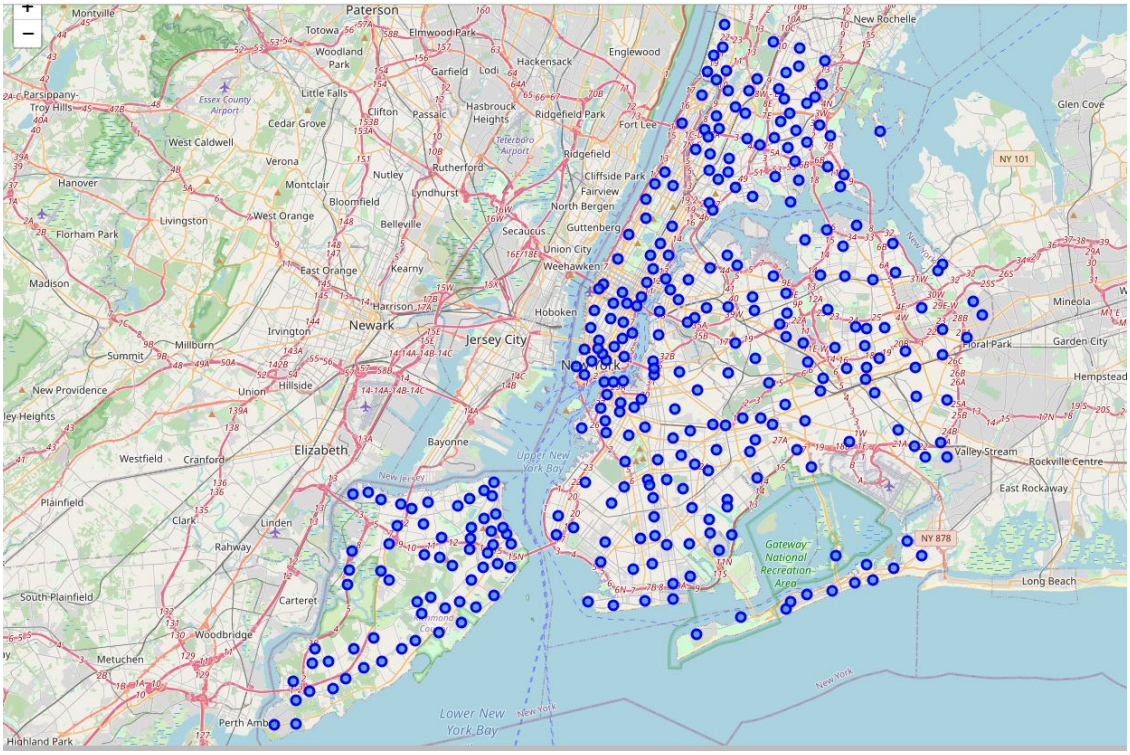
The result is the following dataframe

	Neighborhood	Latitude	Longitude
0	Acton	34.497355	-118.169810
1	Adams-Normandie	34.031461	-118.300208
2	Agoura Hills	34.146736	-118.759885
3	Agua Dulce	34.504927	-118.317104
4	Alhambra	34.085539	-118.136512

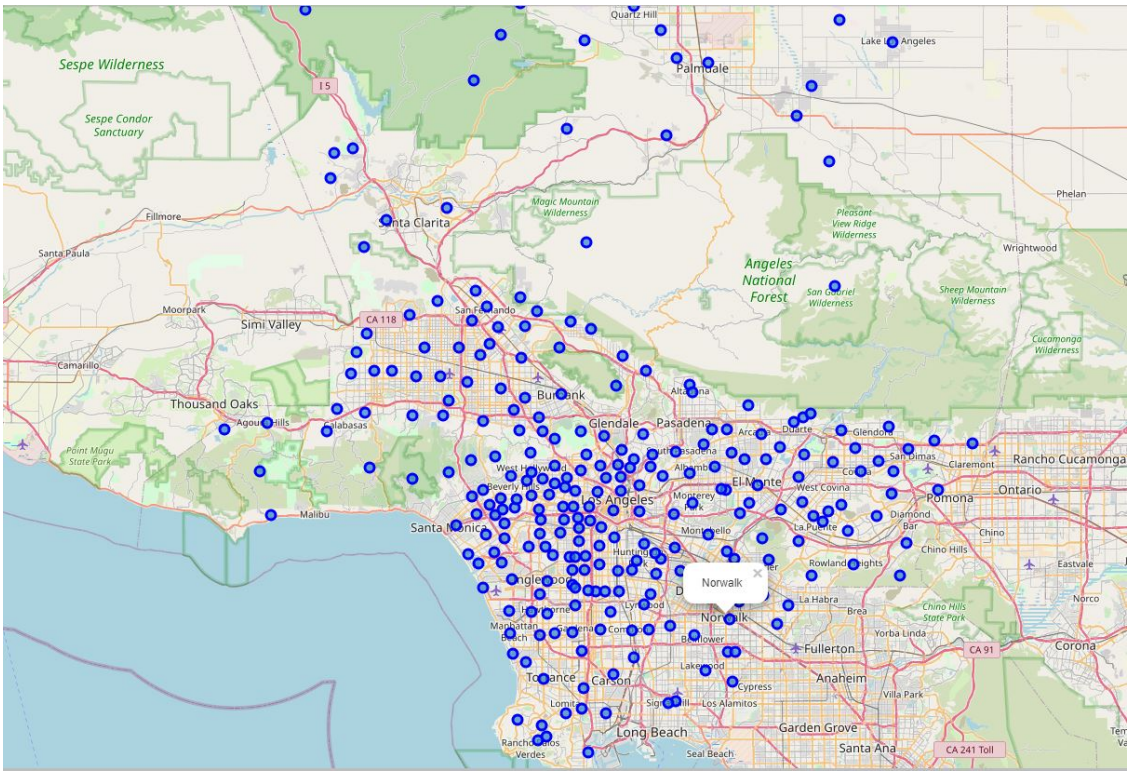
4.3 Data visualization

Now that our dataframes are ready to go, we can visualize the maps of New York and Los Angeles with their respective neighbourhoods that we collected. For this we will use the `Folium` package in Python. It's a tool that allows us to overlay pin points over specified latitudes and longitudes with a clickable popup bubble that indicates to us the name of the neighborhood.

Below, the maps of New York city and its neighnorhoods



and that of Los Angeles:



4.4 Collecting venues information

The newt step is now to explore the neighborhoods and find out what are the common venues. To accomplish this task we will use `Foursquare` as mentionned earlier. What we want to know are the types of venues, and their location for a given neighborhood. For this we will query the `Foursquare` API giving the geographical coordinates of each neighborhood, a radius of 500 meters around the neihborhood within wich we are interested in getting results, and a limit of 100 venues returned to us because we have a personal free ad therefore limited account.

The function below will take in a list of neighborhood and its coordinate, will query `Foursquare` and return to us dataframe containing a list of all the venues returned by `Foursquare` for all the neighborhoods. The function is as follows :

```
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={}&
            CLIENT_ID,
            CLIENT_SECRET,
            VERSION,
            lat,
            lng,
            radius,
            LIMIT)

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

        # return only relevant information for each nearby venue
        venues_list.append([(
            name,
            lat,
            lng,
            v['venue']['name'],
            v['venue']['location']['lat'],
            v['venue']['location']['lng'],
            v['venue']['categories'][0]['name']) for v in results])

    nearby_venues = pd.DataFrame([item for venue_list in venues_list for item in venue_list])
    nearby_venues.columns = ['Neighborhood',
                  'Neighborhood Latitude',
                  'Neighborhood Longitude',
                  'Venue',
                  'Venue Latitude',
                  'Venue Longitude',
```

```
                  'Venue Category']

    return(nearby_venues)
```

We will then call the function to return to us the venues in NYC and LA. Below are two snapshots of the venues dataframes.

New York city venues dataframe :

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Wakefield	40.894705	-73.847201	Lollipops Gelato	40.894123	-73.845892	Dessert Shop
1	Wakefield	40.894705	-73.847201	Rite Aid	40.896649	-73.844846	Pharmacy
2	Wakefield	40.894705	-73.847201	Carvel Ice Cream	40.890487	-73.848568	Ice Cream Shop
3	Wakefield	40.894705	-73.847201	Walgreens	40.896528	-73.844700	Pharmacy
4	Wakefield	40.894705	-73.847201	Dunkin'	40.890459	-73.849089	Donut Shop

Los Angeles venues dataframe :

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Acton	34.497355	-118.169810	Epik Engineering	34.498718	-118.168046	Construction & Landscaping
1	Acton	34.497355	-118.169810	Alma Gardening Co.	34.494762	-118.172550	Construction & Landscaping
2	Adams-Normandie	34.031461	-118.300208	Orange Door Sushi	34.032485	-118.299368	Sushi Restaurant
3	Adams-Normandie	34.031461	-118.300208	Shell	34.033095	-118.300025	Gas Station
4	Adams-Normandie	34.031461	-118.300208	Little Xian	34.032292	-118.299465	Sushi Restaurant

We can see that we have 428 unique venue categories in New York city and 316 in Los Angeles. One could think there are many more types of venues in New York. We will only briefly discuss the difference in number, but with a simple function we can quickly have a look at the categories that are in NY but not in LA and vice versa as follows :

```
# Define function that returns items in List 1 that are not in List 2 or vice versa
def Diff(li1, li2):
    return (list(set(li1)-set(li2)) )

# Categories in NY and LA
ny_cat = ny_venues['Venue Category'].unique()
la_cat = la_venues['Venue Category'].unique()


ny_diff = Diff(ny_cat, la_cat)
ny_diff
```

Here is a quick glance at a few things that are in NY but not in LA "according" to Foursquare

```
['Pet Café',
 'Israeli Restaurant',
 'Print Shop',
 'Waste Facility',
 'Dosa Place',
 'Mini Golf',
 'Bike Shop',
 'Memorial Site',
 'Newsstand',
 'Swiss Restaurant',
 'Caucasian Restaurant',
 'Herbs & Spices Store',
 'Auto Garage',
 'Tennis Stadium',
 'Whisky Bar',
 'Gluten-free Restaurant',
 'Beer Garden',
 'Fruit & Vegetable Store',
 'Bike Trail',
 'Climbing Gym',
 'Outlet Store',
 'Lebanese Restaurant',
 'Soccer Field',
 'Piercing Parlor',
 'History Museum',
 'Scandinavian Restaurant',
 'Turkish Restaurant',
 'Toll Plaza',
```

As you can see, without even going to either city we can immediately know that most of those venues are most definitely present in Los Angeles, and even oter cities. What we are seeing here is a different way of calling the same thing : for example, 'Bike shop' might appear in Los Angeles as 'Bicycle shop', that does not mean there aren't any bike shops in LA, just that it might be listed differently in the Los Angeles data.

4.5 One hot encoding

We have now gathered the venues withing 500 meters of each neighborhood but remember that we are interested in the most common venues let's say the top 10 venues, not just any random venue. For this we will use the one-hot-encoding method. This will allow us to convert, in some sense, the categorical data contained in the venue category into a numerical data, something that our future model can work with. Then we group the venue categories by neighborhood and take the mean of the occurence. Think of it as the probability of having a given venue in that neighborhood

The process is as follows :

```
# one hot encoding
ny_onehot = pd.get_dummies(ny_venues[['Venue Category']], prefix="", prefix_sep="")

ny_onehot.drop(columns=['Neighborhood'], inplace=True)
```

```
# add neighborhood column back to dataframe
ny_onehot['Neighborhood'] = ny_venues['Neighborhood']

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

# Grouping by neighborhoods and calculating the mean for each venue type
ny_grouped = ny_onehot.groupby('Neighborhood').mean().reset_index()
```

The result is the following table for New York (the same is done for Los Angeles):

	Neighborhood	Accessories Store	Adult Boutique	Afghan Restaurant	African Restaurant	Airport Terminal	American Restaurant	Antique Shop	Arcade	Arepa Restaurant	Argentinian Restaurant	Gal
0	Allerton	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	0.00	0.0	0.0	
1	Annadale	0.000000	0.0	0.0	0.0	0.0	0.181818	0.0	0.00	0.0	0.0	
2	Arden Heights	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	0.00	0.0	0.0	
3	Arlington	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	0.00	0.0	0.0	
4	Arrochar	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	0.00	0.0	0.0	
5	Arverne	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	0.00	0.0	0.0	

As explained, for each neighorhood we can see the likelihood of there being a given type of venue. With this we can now move on to the next step where we can find the top 10 venues for each neighborhood

4.6 Finding the top venues

Using the one hot encoded data we can now pull out the top ten venues for each neighborhood. Remember we mentionned the numerical value under each venue category is equivalent to the probability of finding this type of venue in that neighborhood. Therefore, to find the top 10 venues for each neighborhood we need to find the 10 highest venue 'probabilities'. Thereafter we want to organize that in dataframe to facilitate our analysis later on. The process is as follows:

First, let's write a function to sort the venues in descending order. We will use the functions previously given in the labs

```
def return_most_common_venues(row, num_top_venues):
    row_categories = row.iloc[1:]
    row_categories_sorted = row_categories.sort_values(ascending=False)

    return row_categories_sorted.index.values[0:num_top_venues]
```

Now let's create the new dataframe and display the top 10 venues for each neighborhood.

```
num_top_venues = 10

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

# create columns according to number of top venues
columns = ['Neighborhood']
for ind in np.arange(num_top_venues):
```

```
    try:
        columns.append('{}{} Most Common Venue'.format(ind+1, indicators[ind]))
    except:
        columns.append('{}th Most Common Venue'.format(ind+1))

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

for ind in np.arange(ny_grouped.shape[0]):
    ny_neighborhoods_venues_sorted.iloc[ind, 1:] = return_most_common_venues(ny_grouped.iloc[ind

ny_neighborhoods_venues_sorted.head()
```

The result is as follows for New York(the same applies to Los Angeles of course) :

	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	Allerton	Pizza Place	Chinese Restaurant	Discount Store	Deli / Bodega	Spa	Supermarket	Fried Chicken Joint	Donut Shop	Bakery	Pharmacy
1	Annadale	Pizza Place	American Restaurant	Dance Studio	Restaurant	Food	Train Station	Diner	Park	Deli / Bodega	Fish & Chips Shop
2	Arden Heights	Pharmacy	Deli / Bodega	Coffee Shop	Bus Stop	Pizza Place	Flea Market	Factory	Falafel Restaurant	Farm	Farmers Market
3	Arlington	Deli / Bodega	Coffee Shop	Bus Stop	Home Service	Boat or Ferry	Grocery Store	Yoga Studio	Flea Market	Falafel Restaurant	Farm
4	Arrochar	Bus Stop	Italian Restaurant	Deli / Bodega	Hotel	Supermarket	Liquor Store	Sandwich Place	Outdoors & Recreation	Pharmacy	Pizza Place

4.7 Machine learning model : kMeans Clustering

We now move on to building our clustering model. The idea is to hopefully group the similar neighborhoods together. We will first need to determine the number of clusters for each city. We could make a guess but it would be better to find the optimum number of clusters by evaluating some sort of metric score for each number of clusters. This score is the ***Silhouette Coefficient*** , defined for each cluster. The silhouette plot displays a measure of how close each point in one cluster is to points in the neighboring clusters and thus provides a way to assess parameters like number of clusters visually. This measure has a range of [-1, 1] similarily to the linear regression score.

Silhouette coefficients (as these values are referred to as) near +1 indicate that the sample is far away from the neighboring clusters. A value of 0 indicates that the sample is on or very close to the decision boundary between two neighboring clusters and negative values indicate that those samples might have been assigned to the wrong cluster.

The Silhouette Coefficient is defined as such for each sample:

$$SC = \frac{b - a}{max(a, b)} \tag{1}$$

(2)

where *a* and *b* are defined as follows :

a: The mean distance between a sample and all other points in the same class.

b: The mean distance between a sample and all other points in the next nearest cluster.

Now, to find the optimal value of *k* clusters we will loop through *1..n* for *n_clusters* in KMeans and calculate the average Silhouette Coefficient for each sample, we will retain the *k* with the highest average Silhouette Coefficient.

The algorithm for finding the optimal number of clusters is quite complex and will not be displayed in this report so please refer to the notebook to understand what is done behind the scene. We will only display a few samples of the Silhouette Coefficient plots and briefly comment on them.

As it turned out, the optimal number of cluster is ***3***, for both New York and Los Angeles. We will therefore build our two models with that in mind as follows:

```
# Let's cluster new york in 3
n_clusters = 3

#Build our model with n_clusters = 3
kmeans = KMeans(n_clusters=n_clusters, random_state=42).fit_predict(ny_grouped_clustering)
```

Once we have run our model, each neighborhood is assigned a cluster label that we will add in our dataframe and display the clusters on their respective maps.

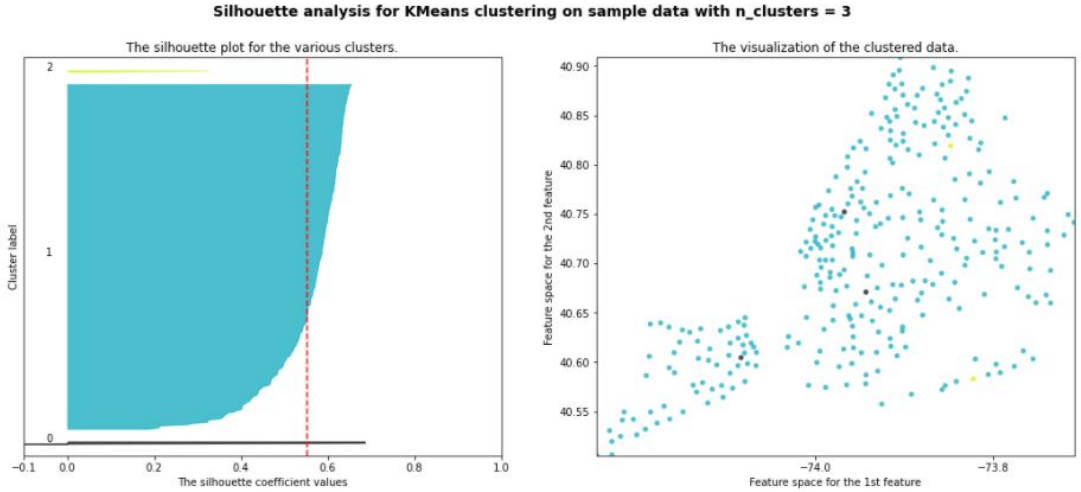
This is now what the Los Angeles dataframe looks like:

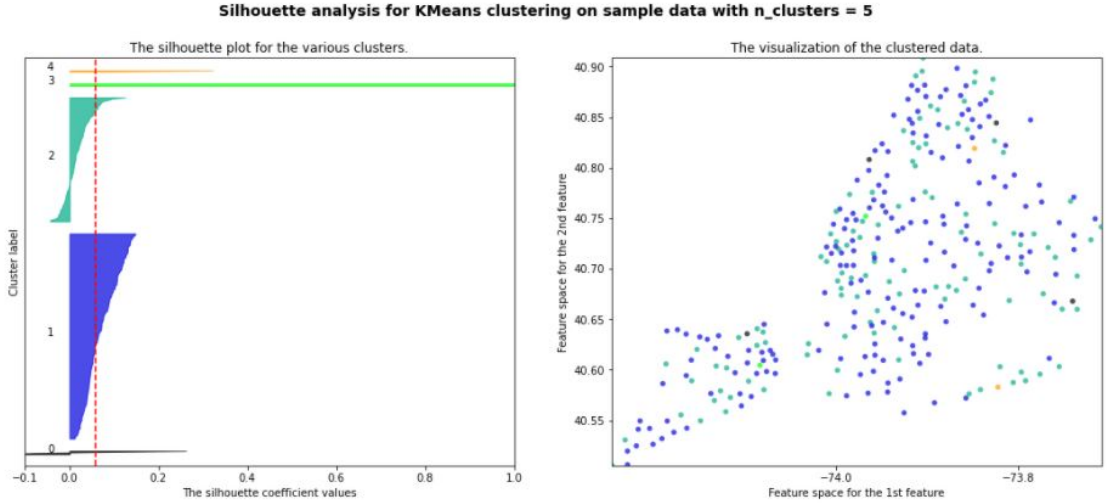
	Cluster Labels	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue	9th Most Common Venue	10th Most Common Venue	Lati
0	2	Acton	Construction & Landscaping	Yoga Studio	Fast Food Restaurant	English Restaurant	Escape Room	Ethiopian Restaurant	Fabric Shop	Falafel Restaurant	Farm	Farmers Market	34.49
1	0	Adams-Normandie	Sushi Restaurant	Park	Home Service	Gas Station	Playground	Grocery Store	Taco Place	Falafel Restaurant	Empanada Restaurant	English Restaurant	34.03
2	0	Agoura Hills	Fast Food Restaurant	Chinese Restaurant	Breakfast Spot	Restaurant	Thai Restaurant	Liquor Store	Lounge	Sushi Restaurant	Bakery	Shipping Store	34.14
3	0	Agua Dulce	Airport	Yoga Studio	Filipino Restaurant	Escape Room	Ethiopian Restaurant	Fabric Shop	Falafel Restaurant	Farm	Farmers Market	Fast Food Restaurant	34.50
4	0	Alhambra	Convenience Store	Bagel Shop	Sporting Goods Shop	Fast Food Restaurant	Pet Store	Hardware Store	Video Store	Mexican Restaurant	Business Service	Pizza Place	34.08

5 Results and Discussions

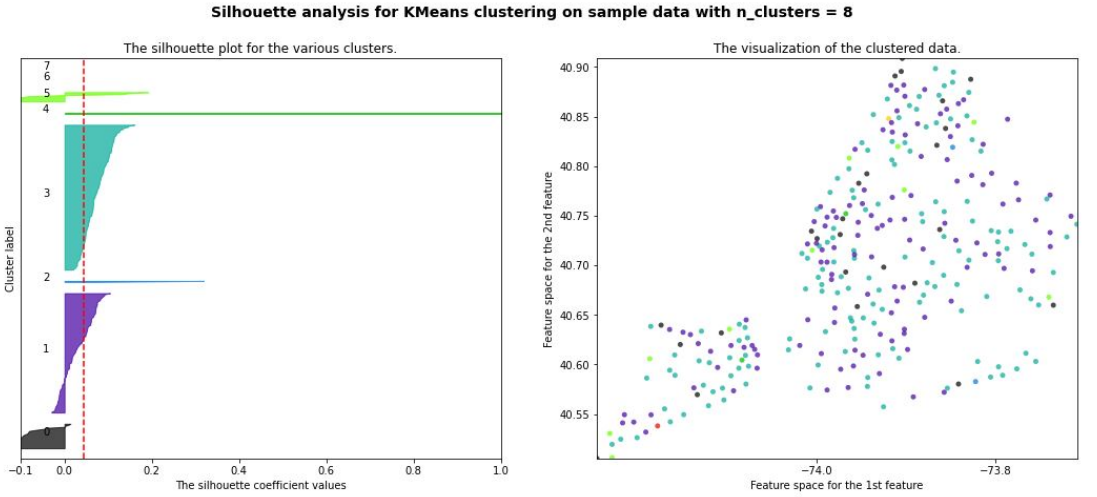
Before diving into our analysis we will first go back to the way we computed the optimal number of clusters. In the figures below, we have on the left the Silhouette coefficient plot for each cluster within a city and on the right the clusters displayed according to their geographic position, for *k=3* and *k=5* . What we observe is that for *k=3* we have most of the neighborhoods clustered in one big cluster whose silhouette coefficient is rather high with most of the silhouette curves ahead of the average silhouette coefficient represented by the dashed red line. On the other hand for *k=5* not only is the average much lower, but there are plenty of negative coefficients. The ideal plot should

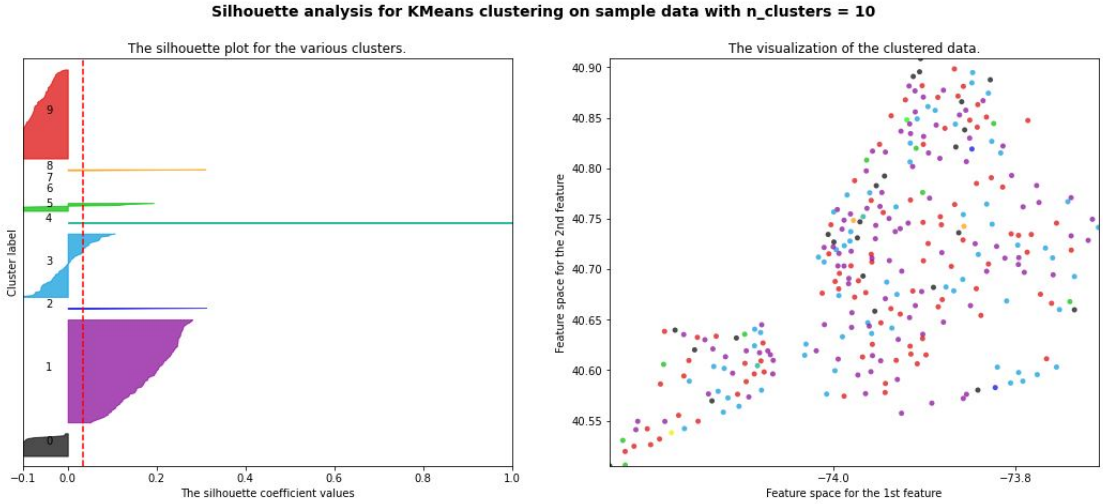
have a high average coefficient and with most curves ahead of the average coefficient. Although the *k=3* plot does not look like the ideal plot, it is the least 'bad' of all.





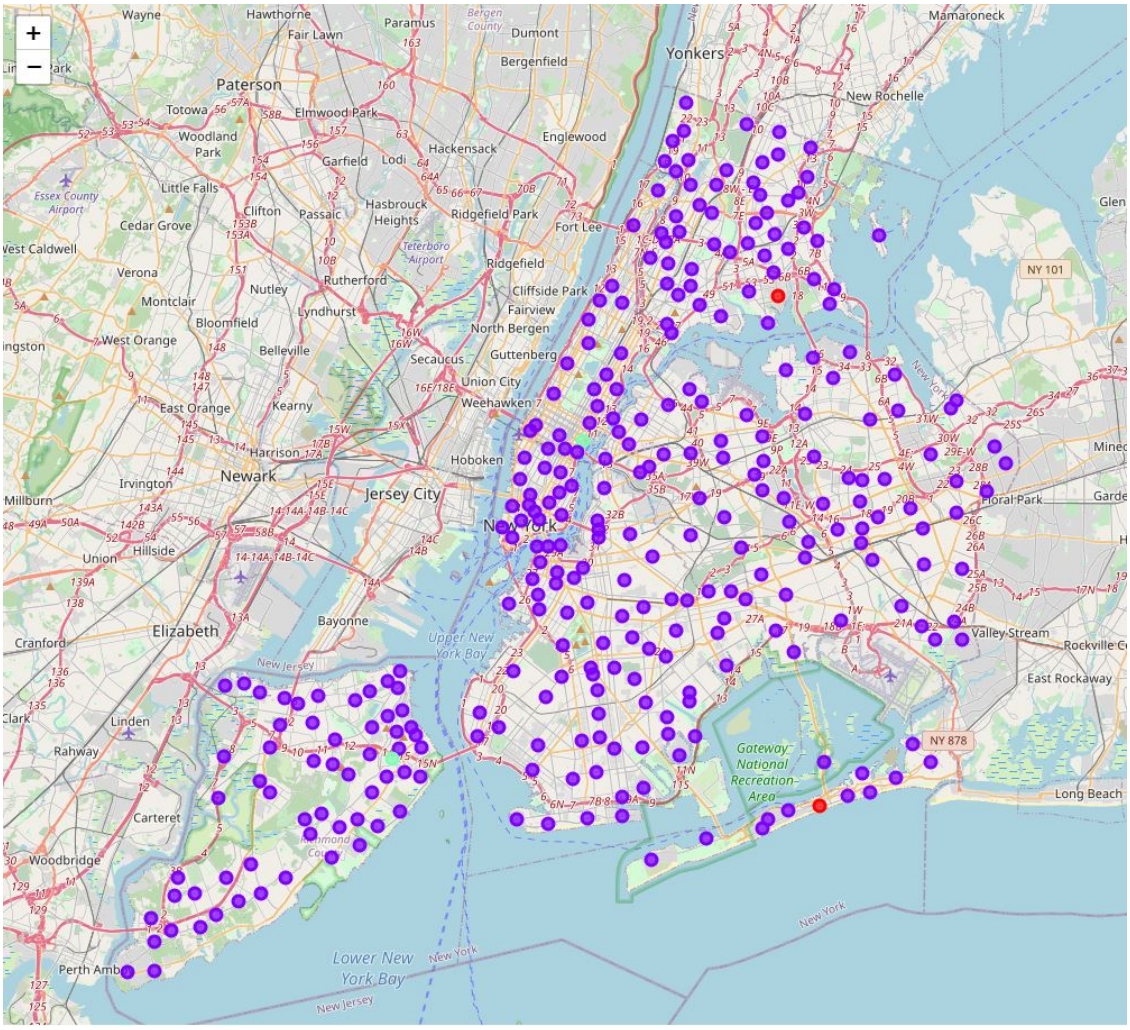
Here are a few more Silhouette plots for *k=8* and *k=10* for you to look at :



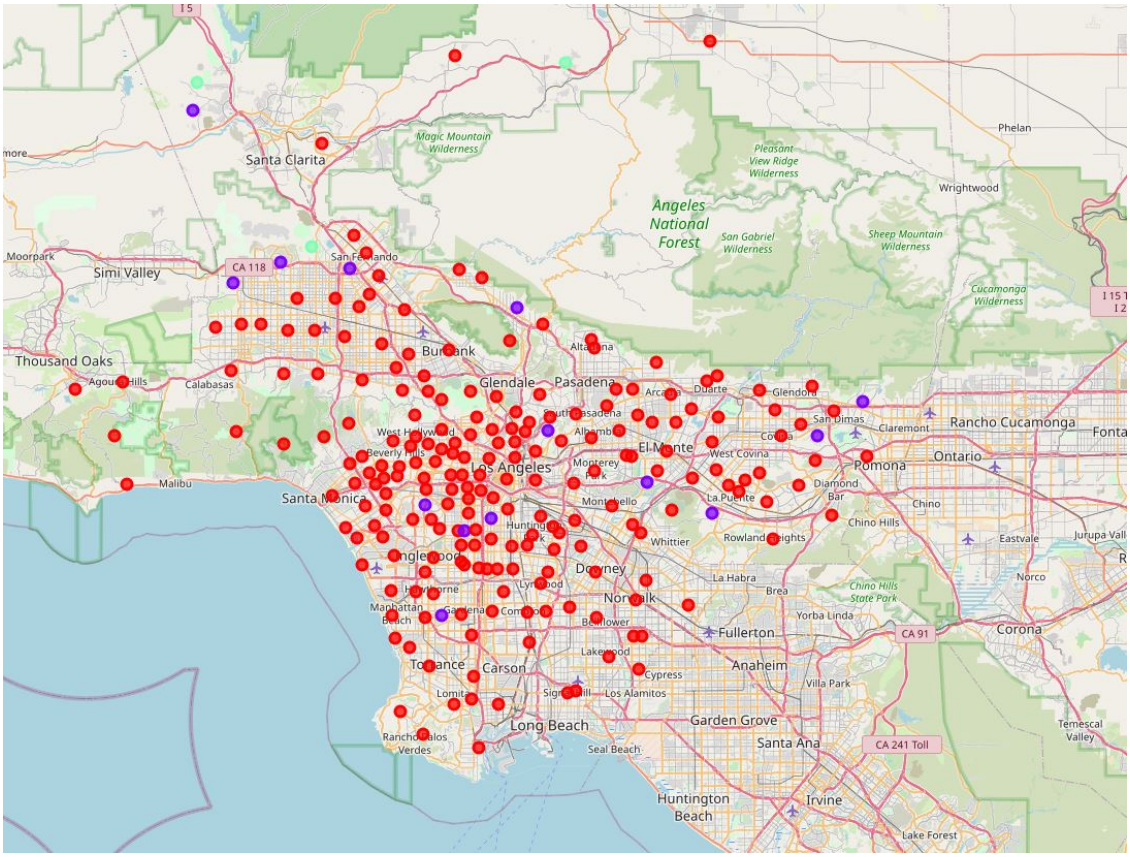


Let's now visualize the clusters in both cities :

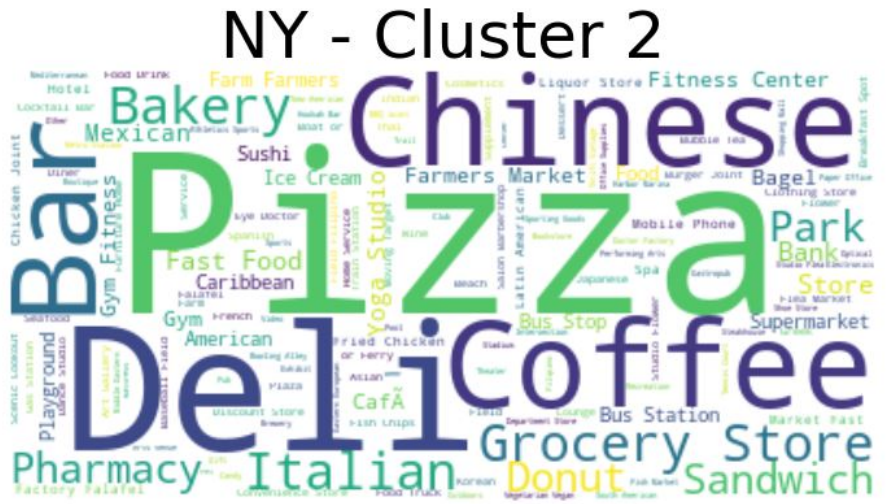
New York city:

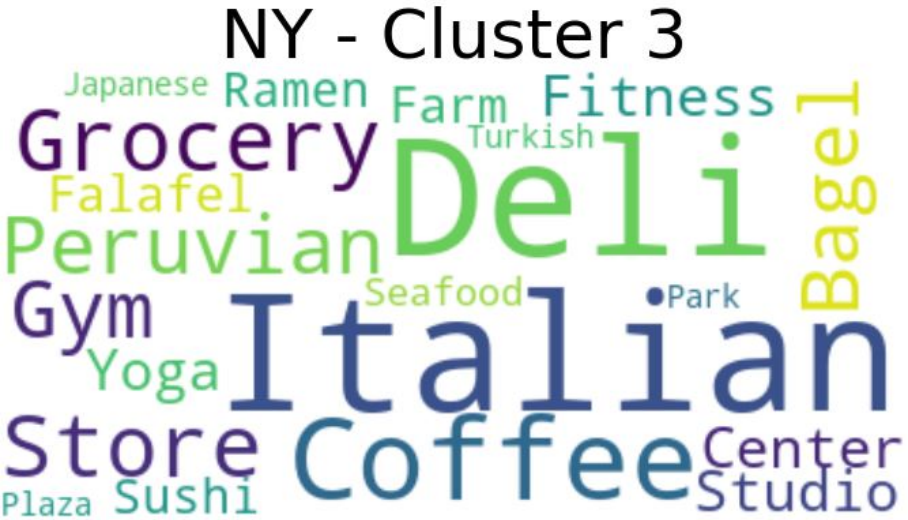

Wayne
Hawthorne
Fair Lawn
Paramus
Bergen County
Dumont
Bergenfield
Yonkers
Mamaroneck
New Rochelle
Paterson
Totowa
Elmwood Park
Woodland Park
Hackensack
Englewood
Garfield
Lodi
Little Falls
Essex County Airport
Clifton
Passaic
Hasbrouck Heights
Ridgefield Park
Fort Lee
Cedar Grove
West Caldwell
Teterboro Airport
Ridgefield
Verona
Nutley
Rutherford
Cliffside Park
Montclair
Lyndhurst
Fairview
Bloomfield
North Bergen
West Orange
Belleville
Secaucus
Guttenberg
Union City
Weehawken
East Orange
Kearny
Harrison
Hoboken
Newark
Jersey City
New York
Irvington
Millburn
Hillside
Union
Elizabeth
Bayonne
Cranford
Upper New York Bay
New Jersey
Linden
Rahway
Carteret
Woodbridge
Perth Amboy
Lower New York Bay
Gateway National Recreation Area
East Rockaway
Valley Stream
Rockville Centre
Long Beach
Floral Park
NY 101
NY 878

Los Angeles:



This is all looking good but in order to have a better understanding of what makes each cluster what it is, let us use wordclouds. By doing so we will see what the most common venues are with a quick glance. Some common words have been excluded from the wordclouds such as shop, bodega(which is the same as deli) and place. Below are the three wordclouds for each cluster:





and for Los Angeles:

What we can take from all those wordclouds is that while there are similarities there are still some differences. Both cities' neighborhoods are clearly very multicultural. Common venues are deli's

and supermarkets, asian cuisine and international cuisine like turkish or peruvian. Both have parks to offer, as well as gyms and fitness centers, conveying a fit lifestyle in both cities.

However, there are a few things that stand out in both cities. Los Angeles seems to offer specific food venues that dominate the rest : Italian for instance, clearly under-represented in New York, which is surprising given New York's immigration history. On the contrary Pizza is one of the most recurring themes in New, which comes to no surprise because NYC is renowned for their quality pizzas. Other cuisines such as, Chinese, Mexican are also more dominant in New York than Los Angeles, most likely representative of the demographics.

Furthermore, we can clearly see that NYC is more of a business city than LA, we see the words *bank*, *bus* and *station* recurring often in NYC and not a single time in LA. Again no surprise, NYC is the financial capital of the US and has one of the best transportation system in the world.

What we can say about Los Angeles, mainly from the words that do not come up in LA but are in NYC, is that it displays a more relaxed image. The word *coffee* is also more prominent in LA, commonly associated with gatherings and the outdoors. In fact, the words *Yoga* and fitness related words occur more often in LA, and the absence of the word *fast-food* conveys a healthier lifestyle in LA. This also seen by the lack of *Pizza* and *Bar* in the wordclouds, mainly associated by unhealthy habits.

6 Conclusions

The purpose of this project was to explore and compare the cities of New York and Los Angeles in order to see how attractive they could be for tourists or even for people looking to immigrate or migrate to either of these two cities.

We have successfully gathered and explored the data, using the available datasets online as well as the data available from Foursquare. We were then able to cluster the neighborhoods with a kMeans clustering model in order to better compare the two cities.

What we can draw from this project is that those two cities are very much alike. Both are very multicultural, as shown by the various cuisine venues in both cities. Both are also considered to be fit cities with parks, gyms and fitness centers.

There are however some differences. New York seems to have a stronger Chinese influence. This could be a selling point for immigrants from these two countries for instance. New York also seems to be more of a business city, that could be a positive point for stakeholders looking to start or expand their business. More over, New York has better transportation system, more convenient for people with no cars, student or tourists alike. In fact this could mean selling point for tourists, as renting a car in Los Angeles could turn out to be expensive and hinder the vacation budget. On the contrary, Los Angeles seems to be a healthier city, reflected by the lack of fast-food, and a more relaxed city.

One thing to take into consideration is that we limited ourselves to 500 m radius around neighborhoods, which in most cases is not sufficient to capture most venues, added to this, we have limited the query results from Foursquare to only 100. This to say that those results are not completely representative of the real world, they only give a general idea.

All in all, the results of this project have shown that while there are similarities in both cities, there are some arguments that could favor one city over the other whether one is looking to spend the

holidays, or start a business or just relax.

7 References

1. Silhouette Coefficient
2. Foursquare API