Final Report for IBM Applied Data Science Capstone Project

1. Introduction

As time flies and cities develop, it is meaningful to discuss the differences between cities as decades passed. I put my focus on two cities Paris and London, A lot has changed over the years and we now take a look at how the cities have grown.

London and Paris are popular for tourists all around the world. They are diverse and multicultural and offer a wide variety of experiences that is widely sought after. I try to group and analyze the neighborhoods of London and Paris respectively and draw insights to what they look like now.

The aim is to help tourists to choose their destinations depending on the experience of the local neighborhoods. My findings will help stakeholders to make decisions and address any concerns they will have including the different kinds of cuisines, stores and what the city looks like right now.

2. Data

I acquire geographical location data for both London and Paris. Using postal codes in each city as a starting point.

London

Please see the following link for the data

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

I set up different names for different kind of data information. *borough* is the Name of Neighborhood; *town* is Name of borough; *post_code* is the Postal codes for London.

There is no information about the geographical locations for London. To solve this, we derive the data of ArcGIS API, which can help us to connect people, locations, and data using interactive maps. We can obtain more detailed information of geographical locations of London. To simplify the data, i also set up different names for different information. *latitude* is the Latitude for Neighborhood; *longitude* is the Longitude for Neighborhood.

Paris

Please see the following link for the data
https://www.data.gouv.fr/fr/datasets/r/e88c6fda-1d09-42a0-a069-606d3259114e

We limit this data to only see and analyze the city of Paris. Here as well, i set up different names for different information. *postal_code* is the Postal codes for France; *nom_comm* is the Name of Neighborhood in France; *nom_dept* is the Name of the boroughs, equivalent to towns in France; *geo_point_2d* is the Tuple containing the latitude and longitude of the Neighbourhoods.

Foursquare API Data

We will need data about different venues in different neighborhoods. In order to obtain it I will use "Foursquare" Location information. Information like venue names, locations, menus and even photos will be included. After obtaining the list of neighborhood, I will connect it to the Foursquare API to gather information about venues inside each and every neighborhood. For each neighborhood, I have chosen the radius to be 500 meters. In this data set, *Neighborhood* is the Name of the Neighborhood; *Neighborhood Latitude* is the Latitude of the Neighborhood; *Neighborhood Longitude* is the Longitude of the Neighborhood; *Venue* is the Name of the Venue; *Venue Latitude* is the Latitude of Venue; *Venue Longitude* is the Longitude of Venue; *Venue Category* is the Category of Venue

Based on the information i collected for both London and Paris, there will be sufficient data to build the model. After analyzing, the observations and findings can be used for people to make decisions.

3. Methodology

First of all, we need to import all packages that we will need.

```
import pandas as pd
```

```
import requests
import numpy as np
import matplotlib.cm as cm
import matplotlib.colors as colors
import folium
from sklearn.cluster import KMeans
```

In order to analyze each of the cities individually, we plot the map to show the neighborhoods being considered, then build our model by clustering all of the similar neighborhoods together. Finally we can plot a new map with the clustered neighborhoods. At last, we will go into the insights and discuss the findings.

4. **Data Processing**

At this stage, I begin with collecting the required data for the cities of London and Paris. We need a data set that includes the information of the postal codes, neighborhood and boroughs of each city.

For the data of London, we scrape the List of areas of London from wikipedia page to take the 2nd table, by using the following code:

```
url_london = "https://en.wikipedia.org/wiki/List_of_areas_of_London"
wiki_london_url = requests.get(url_london)
wiki_london_data = pd.read_html(wiki_london_url.text)
wiki_london_data = wiki_london_data[1]
wiki_london_data
```

After running these codes, we get the following table

	Location	London borough	Post town	Postcode district	Dial code	OS grid ref
0	Abbey Wood	Bexley, Greenwich [7]	LONDON	SE2	020	TQ465785
1	Acton	Ealing, Hammersmith and Fulham[8]	LONDON	W3, W4	020	TQ205805
2	Addington	Croydon[8]	CROYDON	CR0	020	TQ375645
3	Addiscombe	Croydon[8]	CROYDON	CR0	020	TQ345665
4	Albany Park	Bexley	BEXLEY, SIDCUP	DA5, DA14	020	TQ478728
...	...	...	...	...	...	...
528	Woolwich	Greenwich	LONDON	SE18	020	TQ435795
529	Worcester Park	Sutton, Kingston upon Thames	WORCESTER PARK	KT4	020	TQ225655
530	Wormwood Scrubs	Hammersmith and Fulham	LONDON	W12	020	TQ225815
531	Yeading	Hillingdon	HAYES	UB4	020	TQ115825
532	Yiewsley	Hillingdon	WEST DRAYTON	UB7	020	TQ063804

On the other hand, for the data of Paris, we use Pandas to load the table after reading the JSON file:

```
!wget -q -O 'france-data.json' https://www.data.gouv.fr/fr/datasets/r/e88c6fda-1d09-42a0-a069-606d3259114e
print("Data Downloaded!")
paris_raw = pd.read_json('france-data.json')
paris_raw.head()
```

	datasetid	recordid	fields	geometry	record_timestamp
0	correspondances-code-insee-code-postal	21e809b1d4480333c8b6fe7addd8f3b06f343e2c	{'code_comm': '003', 'nom_dept': 'VAL-DE-MARNE...	{'type': 'Point', 'coordinates': [2.3335102498...	2016-09-21T00:29:06.175+02:00
1	correspondances-code-insee-code-postal	c38925e974a8875071da3eb1391a6935d9c97e07	{'code_comm': '430', 'nom_dept': 'SEINE-ET-MAR...	{'type': 'Point', 'coordinates': [2.7879422114...	2016-09-21T00:29:06.175+02:00
2	correspondances-code-insee-code-postal	7c0aa8ba7a7b4320a9cf5abf12288eb76e3eead8	{'code_comm': '412', 'nom_dept': 'SEINE-ET-MAR...	{'type': 'Point', 'coordinates': [2.5107818983...	2016-09-21T00:29:06.175+02:00
3	correspondances-code-insee-code-postal	b123405b4d069c33725418aab20ca0b741f8a5d8	{'code_comm': '598', 'nom_dept': 'VAL-D'OISE',...	{'type': 'Point', 'coordinates': [2.3004997834...	2016-09-21T00:29:06.175+02:00
4	correspondances-code-insee-code-postal	33dea89ab43606076200134a51f2b9d2d7d62256	{'code_comm': '040', 'nom_dept': 'SEINE-ET-MAR...	{'type': 'Point', 'coordinates': [2.5699190953...	2016-09-21T00:29:06.175+02:00

To process the data of London, we replace the spaces with underscores in the title.
The *borough* column has numbers within square brackets that we need to remove by using:

```
wiki_london_data.rename(columns=lambda x: x.strip().replace(" ", "_"), inplace=True)
wiki_london_data['borough'] = wiki_london_data['borough'].map(lambda x: x.rstrip(']').rstrip('0123456789').rstrip('['))
```

For Paris on the other hand:

```
paris_field_data = pd.DataFrame()
for f in paris_raw.fields:
    dict_new = f
    paris_field_data = paris_field_data.append(dict_new, ignore_index=True)

paris_field_data.head()
```

For both of our data sets, we only need the data of borough, neighborhood, postal codes and geographical locations (latitude and longitude). So we need to select the useful columns:

```
df1 = wiki_london_data.drop( [ wiki_london_data.columns[0], wiki_london_data.columns[4], wiki_london_data.columns[5] ],
axis=1)

df_2 = paris_field_data[['postal_code','nom_comm','nom_dept','geo_point_2d']]
```

Both of the Data sets contain information that related to all other cities in the whole country. We can narrow down the data by selecting only the neighborhoods pertaining to 'London' and 'Paris'

```
df1 = df1[df1['town'].str.contains('LONDON')]

df_paris = df_2[df_2['nom_dept'].str.contains('PARIS')].reset_index(drop=True)
```

When looking over the London data set, we can see that we don't have the geographical location data. So we need to extrapolate the missing data for our neighborhoods. We perform this by leveraging the ArcGIS API. With this we can get the latitude and longitude of our London neighborhood data.

```
from arcgis.geocoding import geocode
from arcgis.gis import GIS
gis = GIS()
```

```
def get_x_y_uk(address1):
    lat_coords = 0
    lng_coords = 0
    g = geocode(address='{}, London, England, GBR'.format(address1))[0]
    lng_coords = g['location']['x']
    lat_coords = g['location']['y']
    return str(lat_coords) +","+ str(lng_coords)
```

Then passing postal codes of London to get the geographical co-ordinates.

```
coordinates_latlng_uk = geo_coordinates_uk.apply(lambda x: get_x_y_uk(x))
```

Then we merge our source data with the geographical co-ordinates to make our data set ready.

```
london_merged = pd.concat([df1,lat_uk.astype(float), lng_uk.astype(float)], axis=1)
london_merged.columns= ['borough','town','post_code','latitude','longitude']
london_merged
```

	borough	town	post_code	latitude	longitude
0	Bexley, Greenwich	LONDON	SE2	51.49245	0.12127
1	Ealing, Hammersmith and Fulham	LONDON	W3, W4	51.51324	-0.26746
6	City	LONDON	EC3	51.51200	-0.08058
7	Westminster	LONDON	WC2	51.51651	-0.11968
9	Bromley	LONDON	SE20	51.41009	-0.05683
...	...	...	...	...	...
523	Redbridge	LONDON	IG8, E18	51.58977	0.03052
524	Redbridge, Waltham Forest	LONDON, WOODFORD GREEN	IG8	51.50642	-0.12721
527	Barnet	LONDON	N12	51.61592	-0.17674
528	Greenwich	LONDON	SE18	51.48207	0.07143
530	Hammersmith and Fulham	LONDON	W12	51.50645	-0.23691

On the other hand for the Paris data set, we don't need to get the geographical coordinates of that, we just need to extract the latitude and longitude for the column:

```
paris_lat = paris_latlng.apply(lambda x: x.split(',')[0])
paris_lat = paris_lat.apply(lambda x: x.lstrip('['))

paris_lng = paris_latlng.apply(lambda x: x.split(',')[1])
paris_lng = paris_lng.apply(lambda x: x.rstrip(']'))

paris_geo_lat  = pd.DataFrame(paris_lat.astype(float))
paris_geo_lat.columns=['Latitude']

paris_geo_lng = pd.DataFrame(paris_lng.astype(float))
paris_geo_lng.columns=['Longitude']
```

Then create our Paris data set with the required information:

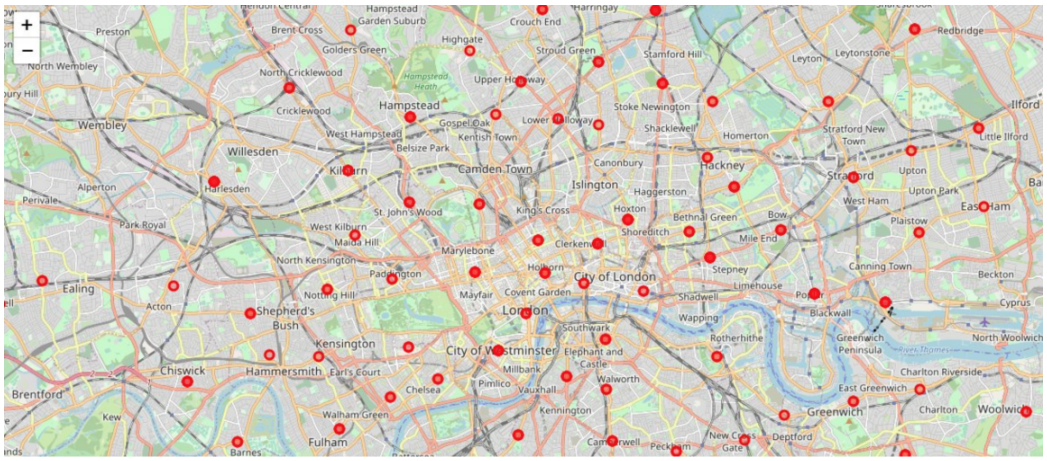
```
paris_combined_data = pd.concat([df_paris.drop('geo_point_2d', axis=1), paris_geo_lat, paris_geo_lng], axis=1)
paris_combined_data
```

	postal_code	nom_comm	nom_dept	Latitude	Longitude
0	75010	PARIS-10E-ARRONDISSEMENT	PARIS	48.876029	2.361113
1	75016	PARIS-16E-ARRONDISSEMENT	PARIS	48.860399	2.262100
2	75009	PARIS-9E-ARRONDISSEMENT	PARIS	48.876896	2.337460
3	75015	PARIS-15E-ARRONDISSEMENT	PARIS	48.840155	2.293559
4	75002	PARIS-2E-ARRONDISSEMENT	PARIS	48.867903	2.344107
5	75011	PARIS-11E-ARRONDISSEMENT	PARIS	48.859415	2.378741
6	75005	PARIS-5E-ARRONDISSEMENT	PARIS	48.844509	2.349859
7	75019	PARIS-19E-ARRONDISSEMENT	PARIS	48.886869	2.384694
8	75020	PARIS-20E-ARRONDISSEMENT	PARIS	48.863187	2.400820
9	75003	PARIS-3E-ARRONDISSEMENT	PARIS	48.863054	2.359361
10	75006	PARIS-6E-ARRONDISSEMENT	PARIS	48.848968	2.332671

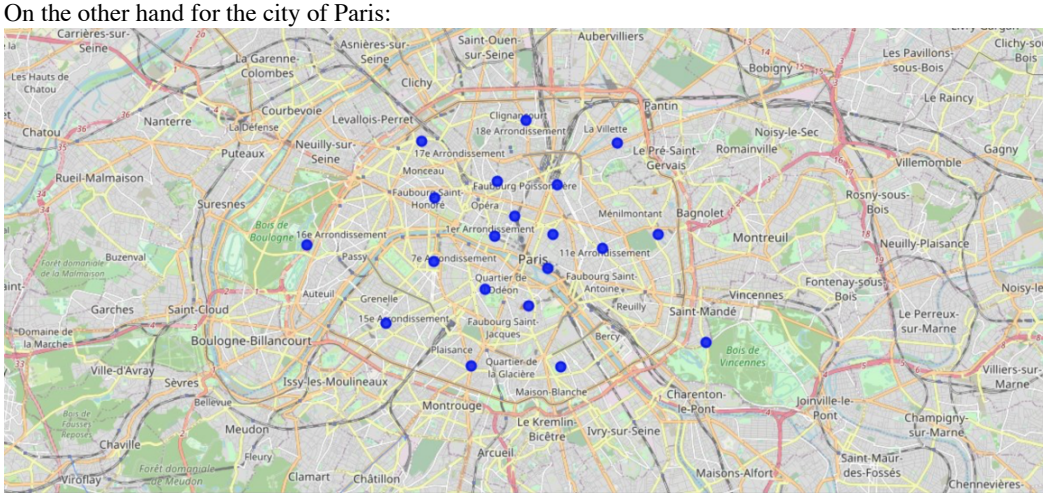
11	75018	PARIS-18E-ARRONDISSEMENT	PARIS	48.892735	2.348712
12	75008	PARIS-8E-ARRONDISSEMENT	PARIS	48.872527	2.312583
13	75013	PARIS-13E-ARRONDISSEMENT	PARIS	48.828718	2.362468
14	75012	PARIS-12E-ARRONDISSEMENT	PARIS	48.835156	2.419807
15	75007	PARIS-7E-ARRONDISSEMENT	PARIS	48.856083	2.312439

Now we can jump into the next stage of visualizing the plots.
By using the Folium package, we can visualize the maps of London and Paris with the neighborhoods that we collected.

Neighborhood map of London:



On the other hand for the city of Paris:



Now that we have visualized the neighborhoods of two cities, then we need to find out what each neighborhood looks like, and what are the common venues and venue categories within a 500m radius.
With the help of Foursquare we can define a function which collects information pertaining to each neighborhood including that of the name of the neighborhood, geographical coordinates, venue and venue categories.

```
LIMIT=100

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={}&ll={},{}&radius={}&limit
={}'.format(
            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,
```

```
            lat,
            lng,
            v['venue']['name'],
            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 = ['Neighbourhood',
                  'Neighbourhood Latitude',
                  'Neighbourhood Longitude',
                  'Venue',
                  'Venue Category']

    return(nearby_venues)
```

And the result will look like:

	Neighbourhood	Neighbourhood Latitude	Neighbourhood Longitude	Venue	Venue Category
0	Bexley, Greenwich	51.49245	0.12127	Sainsbury's	Supermarket
1	Bexley, Greenwich	51.49245	0.12127	Lesnes Abbey	Historic Site
2	Bexley, Greenwich	51.49245	0.12127	Lidl	Supermarket
3	Bexley, Greenwich	51.49245	0.12127	Abbey Wood Railway Station (ABW)	Train Station
4	Bexley, Greenwich	51.49245	0.12127	Bean @ Work	Coffee Shop

Since we are trying to find out what are the different kinds of venue categories in each neighborhood, we can use the One Hot Encoding to work with our categorical data type of the venue categories. We perform one hot encoding and then calculate the mean of the grouped venue categories for each of the neighborhoods.

```
# One hot encoding
London_venue_cat = pd.get_dummies(venues_in_London[['Venue Category']], prefix="", prefix_sep="")

# Adding neighbourhood to the mix
London_venue_cat['Neighbourhood'] = venues_in_London['Neighbourhood']

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

# Grouping and calculating the mean
London_grouped = London_venue_cat.groupby('Neighbourhood').mean().reset_index()
```

	Neighbourhood	Accessories Store	Adult Boutique	African Restaurant	American Restaurant	Antique Shop	Arcade	Arepa Restaurant	Argentinian Restaurant	Art Gallery	Art Museum	Arts & Crafts Store	Asian Restaurant	Athletics & Sports
0	Barnet	0.0	0.0	0.0	0.001887	0.0	0.0	0.0	0.007547	0.0	0.0	0.0	0.020755	0.0
1	Barnet, Brent, Camden	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.000000	0.0
2	Bexley	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.000000	0.0
3	Bexley, Greenwich	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.000000	0.0
4	Bexley, Greenwich	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.000000	0.0

In the next step, we rank and label the top venue categories in our neighborhood.

```
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]
```

```
num_top_venues = 10

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

# create columns according to number of top venues
columns = ['Neighbourhood']
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))
```

There are many categories, so we only consider top 10 categories of them.
Then we get the top venue categories in the neighborhood of London.

```
# create a new dataframe for London
neighborhoods_venues_sorted_london = pd.DataFrame(columns=columns)
```

```
neighborhoods_venues_sorted_london = pd.DataFrame(columns=columns)
neighborhoods_venues_sorted_london['Neighbourhood'] = London_grouped['Neighbourhood']

for ind in np.arange(London_grouped.shape[0]):
    neighborhoods_venues_sorted_london.iloc[ind, 1:] = return_most_common_venues(London_grouped.iloc[ind, :], num_top_ve
nues)

neighborhoods_venues_sorted_london.head()
```

	Neighbourhood	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	Barnet	Coffee Shop	Café	Grocery Store	Pub	Italian Restaurant	Supermarket	Pharmacy	Chinese Restaurant	Turkish Restaurant	Pizza Place
1	Barnet, Brent, Camden	Gym / Fitness Center	Music Venue	Clothing Store	Supermarket	Zoo Exhibit	Film Studio	Event Space	Exhibit	Falafel Restaurant	Farmers Market
2	Bexley	Supermarket	Historic Site	Train Station	Platform	Convenience Store	Coffee Shop	Bus Stop	Golf Course	Construction & Landscaping	Park
3	Bexley, Greenwich	Park	Construction & Landscaping	Sports Club	Bus Stop	Golf Course	Historic Site	Food Service	Convenience Store	Department Store	Cycle Studio
4	Bexley, Greenwich	Supermarket	Platform	Convenience Store	Historic Site	Train Station	Coffee Shop	Zoo Exhibit	Film Studio	Event Space	Exhibit

Then we move on to Model Building part. We will be using KMeans Clustering Machine learning algorithm to cluster similar neighborhood together. And we will be going with the number of clusters as 5.

```
# set number of clusters
k_num_clusters = 5

London_grouped_clustering = London_grouped.drop('Neighbourhood', 1)

# run k-means clustering
kmeans_london = KMeans(n_clusters=k_num_clusters, random_state=0).fit(London_grouped_clustering)
neighborhoods_venues_sorted_london.insert(0, 'Cluster Labels', kmeans_london.labels_ +1)
```

We then join London_merged with our neighborhood venues sorted to add latitude & longitude for each of the neighborhood to prepare it for visualization.

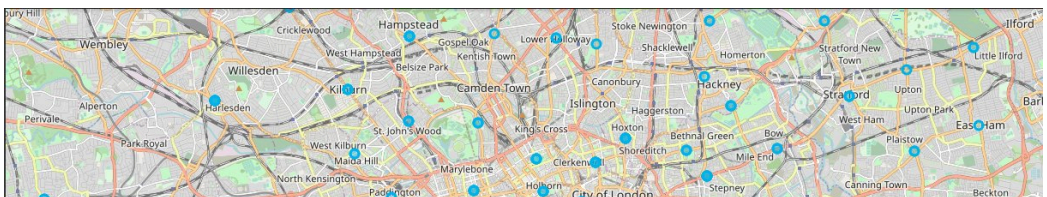
```
london_data = london_merged

london_data = london_data.join(neighborhoods_venues_sorted_london.set_index('Neighbourhood'), on='borough')

london_data.head()
```

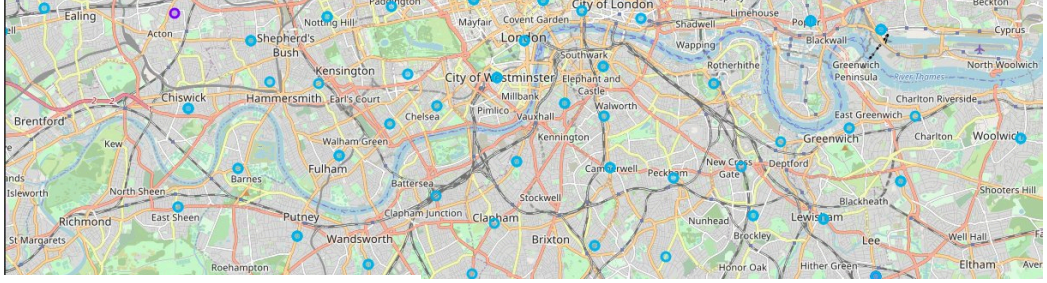
	borough	town	post_code	latitude	longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th M Comm Venu
0	Bexley, Greenwich	LONDON	SE2	51.49245	0.12127	4	Supermarket	Platform	Convenience Store	Historic Site	Train Station	Coffee Shop	Zoo Exhibit	Film Studi
1	Ealing, Hammersmith and Fulham	LONDON	W3, W4	51.51324	-0.26746	1	Grocery Store	Train Station	Breakfast Spot	Park	Indian Restaurant	Deli / Bodega	Fish Market	Exhib
6	City	LONDON	EC3	51.51200	-0.08058	2	Coffee Shop	Italian Restaurant	Hotel	Pub	Gym / Fitness Center	Food Truck	Sandwich Place	Beer
7	Westminster	LONDON	WC2	51.51651	-0.11968	2	Hotel	Coffee Shop	Pub	Sandwich Place	Café	Italian Restaurant	Restaurant	Theat
9	Bromley	LONDON	SE20	51.41009	-0.05683	2	Supermarket	Grocery Store	Convenience Store	Hotel	Fast Food Restaurant	Park	Italian Restaurant	Gym / Fitnes Cente

At the same time, missing data is collected and compiled. Although the Model is built. All that's remaining is to see the clustered neighborhoods on the map. We can use Folium package to do so. We need to drop all the NaN values to prevent data skew first.

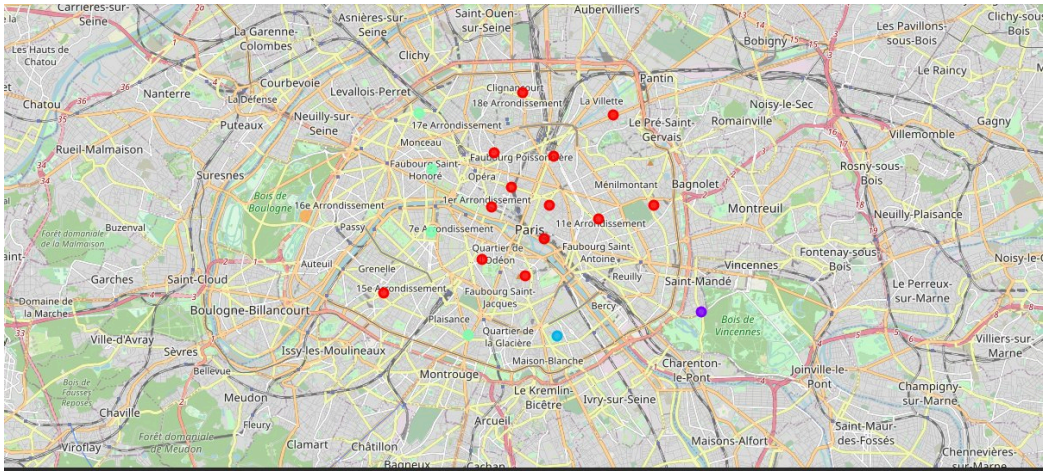
```
london_data_nonan = london_data.dropna(subset=['Cluster Labels'])
```

Map of clustered neighborhoods of London:





Map of clustered neighborhoods of Paris:



The last part of this section is to examine our clusters by expanding on our code using the Cluster Labels column:

Cluster1:

```
london_data_nonan.loc[london_data_nonan['Cluster Labels'] == 1, london_data_nonan.columns[[1] + list(range(5, london_data_nonan.shape[1]))]]
```

Cluster 2:

```
london_data_nonan.loc[london_data_nonan['Cluster Labels'] == 2, london_data_nonan.columns[[1] + list(range(5, london_data_nonan.shape[1]))]]
```

Cluster 3:

```
london_data_nonan.loc[london_data_nonan['Cluster Labels'] == 3, london_data_nonan.columns[[1] + list(range(5, london_data_nonan.shape[1]))]]
```

Cluster 4:

```
london_data_nonan.loc[london_data_nonan['Cluster Labels'] == 4, london_data_nonan.columns[[1] + list(range(5, london_data_nonan.shape[1]))]]
```

Cluster 5:

```
london_data_nonan.loc[london_data_nonan['Cluster Labels'] == 5, london_data_nonan.columns[[1] + list(range(5, london_data_nonan.shape[1]))]]
```

5. **Result and Discussion**

By comparing the data we obtained for two cities, we can see that the neighborhoods of London are very muliti-cultural. There are a lot of different cusines including Indian, Italian, Turkish and Chinese. Also, London has a lot of shopping options too, like markets, flower shops, fish markets, Fishing stores and clothing stores. The main modes of transport seem to be Buses and trains. For leisure, the neighborhoods are set up to have lots of parks, golf courses, zoo, gyms and Historic sites. Overall, the city of London offers a multicultural, diverse and certainly an entertaining experience.

Paris on the other hand, is relatively small. It has a wide variety of cuisine and eateries. There are a lot of hangout spots including many Restaurants and Bars. Paris has a lot of Bistro's. Different means of public transport in Paris which includes buses, bikes, boats or ferries. For leisure and sight seeing, there are a lot of Plazas, Trails, Parks, Historic sites, clothing shops, Art galleries and Museums. Overall, Paris seems like the relaxing vacation spot with a mix of lakes, historic spots and a wide variety of cuisine to try.

6. **Conclusion**

The purpose of this project is to analyze the cities of London and Paris, and to see how attractive it is to potential tourists and migrants. We explored both the cities based on their postal codes and then extrapolated the common venues present in each of the neighborhoods, and finally concluding with clustering similar neighborhoods together.

We could see that each of the neighborhoods in both the cities have a wide variety of experiences to offer which is unique in it's own way. The cultural diversity is quite evident which also gives the feeling of a sense of inclusion.

Both Paris and London seem to offer a vacation stay or a romantic getaway with a lot of places to explore, there are beautiful landscapes, amazing foods and a wide variety of culture. These two cities both offer beautiful and impressive landscapes and neighborhoods for tourists to explore.