“A Delightful Tale of Two Cities”

A data science report for work submitted

By

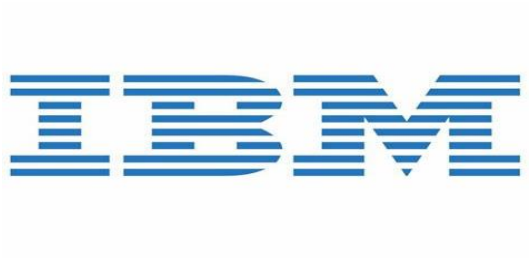
VENKATA SHANMUKESWAR TATA

as a part of

IBM Professional Certification Program with Coursera

Under the supervision of

Coursera	IBM
Coursera is an American massive open online course provider founded in 2012 by Stanford University's computer science professors Andrew Ng and Daphne Koller that offers massive open online courses, specializations, degrees, professional and master track courses.	International Business Machines Corporation is an American cloud platform and cognitive solutions multinational technology and consulting company headquartered in Armonk, New York, with more than 350,000 employees serving clients in 170 countries.



Project Details:

Author	Venkata Shanmukeswar Tata
Published	Oct 15, 2020
Version	1.0
Contributions	NA

1. Cover Page
2. Table Of Contents
3. Abstract
4. Introduction
5. Business Problem
6. Data Description
   6.1 London
   6.2 ArcGIS API
   6.3 Paris
   6.4 Foursquare API Data
7. Methodology
   7.1 Data Collection
   7.2 Data Preprocessing
   7.3 Feature Selection
   7.4 Feature Engineering
   7.5 Visualizing the Neighborhoods of London and Paris
      7.5.1 Neighborhoods of London
      7.5.2 Neighborhoods of Paris
8. Model Building
   8.1 Model Building for London neighborhood
   8.2 Model Building for Paris Neighborhood
9. Results and Discussion
10. Conclusion

References

Table of Contents

1. Cover Page 1
2. Table Of Contents 3
3. Abstract 4
4. Introduction 5
5. Business Problem 6
6. Data Description 7
   - 6.1 London 7
   - 6.2 ArcGIS API 7
   - 6.3 Paris 8
   - 6.4 Foursquare API Data 8
7. Methodology 9
   - 7.1 Data Collection 10
   - 7.2 Data Preprocessing 11
   - 7.3 Feature Selection 11
   - 7.4 Feature Engineering 12
   - 7.5 Visualizing the Neighborhoods of London and Paris 13
     - 7.5.1 Neighborhoods of London 13
     - 7.5.2 Neighborhoods of Paris 14
8. Model Building 16
   - 8.1 Model Building for London neighborhood 16
   - 8.2 Model Building for Paris Neighborhood 16
9. Results and Discussion 18
10. Conclusion 23

References 24

Abstract

As a part of Coursera IBM Professional data science project, we will be analyzing the similarities and dissimilarities of two cities. **London (capital of the United Kingdom) and Paris (capital of France)** has been taken as an example for this study

Analyzing cities using venue based data from Foursquare lead to an overall understanding of the type of venues in each neighborhood and presented some of the key features of the cities but the level of data is not adequate to provide a comprehensive analysis for a city-to-city comparison. For a potential interested person (job-seeker or person deciding to move to either of the cities) or a bigger clientele like a business corporation or city planners, one would need to do a more detailed analysis adding features such as rents, salaries, transportation, cost of living, growth rate, economy, etc.

The capstone project provided a medium to understand in depth about how real life data science projects work and what all steps go in building a data science methodology. All steps from understanding the business problem, data understanding to data preparation, and model building were discussed in detail here.

The reports highlights the methodology and analysis used for the final capstone project in the IBM Data Science Professional course. Full source code is available in GitHub

4. Introduction

Picking a city, when it comes to London and Paris is always a hard decision as both these cities are truly global, multicultural, and cosmopolitan cities found at the heart of two European nations. Along with being two of Europe's most important diplomatic centres, they are major centers for finance, commerce, sciences, fashion, arts, culture and gastronomy. Both London (capital of the United Kingdom) and Paris (capital of France) have a rich history and are two of the most visited and sought-after cities in Europe. London is the largest city within the UK and stands on River Thames in South East England. Paris, on the other hand, is located in the north-central part of the nation. Similar to London, the city also stands along a river, commonly known as the Seine River.

A Tale of Two cities, a novel written by Charles Dickens was set in London and Paris which takes place during the French Revolution. These cities were both happening then and now. A lot has changed over the years and we now take a look at how the cities have grown. London and Paris are quite the popular tourist and vacation destinations for people all around the world. They are diverse and multicultural and offer a wide variety of experiences that is widely sought after. We try to group the neighborhoods of London and Paris respectively and draw insights to what they look like now.

A Tale of Two Cities was published in 1859, right between Little Dorrit and Great Expectations, two of other Dicken's novels. These are novels set in Dicken's own century, if several decades before the years of their publication. And they're focused on individual stories of crime, poverty and class in 19th Century England and Europe. Tale of Two Cities, by contrast, is one of only two works Dickens wrote, that could properly be considered historical novels. The other is Barnaby Rudge, set in the 1770s and the early 1780s. A Tale of Two Cities has always been considered one of the least Dickensian of Dickens' novels. For example, a scholar writing for the Cambridge History of English Literature, published in 1941, wrote, many people who do not care for the rest of Dickens like it greatly; many who are enthusiastic about Dickens refuse to give it a second reading. It is the least Dickensian of all the tales. And it does have an odd relationship to his copious.

5. Business Problem

Our goal is to perform a comparison of the two cities to see how similar or dissimilar they are. Such techniques allow users to identify similar neighbourhoods among cities based on amenities or services being offered locally, and thus can help in understanding the local area activities, what are the hubs of different activities, how citizens are experiencing the city, and how they are utilising its resources.

What kind of clientele would benefit from such an analysis?

1. A potential job seeker with transferable skills may wish to search for jobs in selective cities which provide the most suitable match for their qualifications and experience in terms of salaries, social benefits, or even in terms of a culture fit for expats.

2. Further, a person buying or renting a home in a new city may want to look for recommendations for locations in the city similar to other cities known to them.

3. Similarly, a large corporation looking to expand its locations to other cities might benefit from such an analysis.

4. Many within-city urban planning computations might also benefit from modelling a city's relationship to other cities.

6. Data Description

We require geographical location data for both London and Paris. Postal codes in each city serve as a starting point. Using Postal codes, we use can find out the neighborhoods, boroughs, venues and their most popular venue categories.

a. London

To derive our solution, We scrape our data from here

This Wikipedia page has information about all the neighbourhoods, we limit it London.

1. borough: Name of Neighborhood
2. town: Name of borough
3. postcode: Postal codes for London.

This Wikipedia page lacks information about the geographical locations. To solve this problem we use ArcGIS API.

b. ArcGIS API

ArcGIS Online enables you to connect people, locations, and data using interactive maps. Work with smart, data-driven styles and intuitive analysis tools that deliver location intelligence. Share your insights with the world or specific groups.

More specifically, we use ArcGIS to get the geo locations of the neighbourhoods of London. The following columns are added to our initial dataset which prepares our data.

1. latitude: Latitude for Neighborhood
2. longitude: Longitude for Neighborhood

c. Paris

To derive our solution, We leverage JSON data available at here

The JSON file has data about all the neighbourhoods in France, we limit it to Paris.

1. postalcode : Postal codes for France
2. nomcomm : Name of Neighborhoods in France
3. nomdept : Name of the boroughs, equivalent to towns in France
4. geo_point_2d : Tuple containing the latitude and longitude of the Neighborhoods.

d. Foursquare API

We will need data about different venues in different neighbourhoods of that specific borough. In order to gain that information we will use "Foursquare" locational information. Foursquare is a location data provider with information about all manner of venues and events within an area of interest. Such information includes venue names, locations, menus and even photos. As such, the foursquare location platform will be used as the sole data source since all the stated required information can be obtained through the API.

After finding the list of neighbourhoods, we then connect to the Foursquare API to gather information about venues inside each and every neighbourhood. For each neighbourhood, we have chosen the radius to be 500 meters.

The data retrieved from Foursquare contained information of venues within a specified distance of the longitude and latitude of the postcodes. The information obtained per venue as follows:

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

7. Methodology

Based on all the information collected for both London and Paris, we have sufficient data to build our model. We cluster the neighbourhoods together based on similar venue categories. We then present our observations and findings. Using this data, our stakeholders can take the necessary decision.

```
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
```

Package breakdown:

1. Pandas : To collect and manipulate data in JSON and HTMl and then data analysis
2. requests : Handle http requests
3. matplotlib : Detailing the generated maps
4. folium : Generating maps of London and Paris
5. sklearn : To import Kmeans which is the machine learning model that we are using.

The approach taken here is to explore each of the cities individually, plot the map to show the neighbourhoods being considered and then build our model by clustering all of the similar neighbourhoods together and finally plot the new map with the clustered neighbourhoods. We draw insights and then compare and discuss our findings.

a. Data Collection

In the data collection stage, we begin with collecting the required data for the cities of London and Paris. We need data that has the postal codes, neighbourhoods and boroughs specific to each of the cities.

To collect data for London, we scrape the List of areas of London Wikipedia page to take the 2nd table 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
```

To collect data for Paris, we download the JSON file containing all the postal codes of France from [datasets]

```
!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()
```

The resultant dataframes are:

	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

533 rows × 6 columns

Table 1: London city boroughs data frame

	datasetid	recordid	fields	geometry	record_timestamp
0	correspondances-code-insee-code-postal	2bf36b38314b6c39dfbcd09225f97fa532b1fc45	{'code_comm': '645', 'nom_dept': 'ESSONNE', 's...	{'type': 'Point', 'coordinates': [2.2517129721...	2016-09-21T00:29:06.175+02:00
1	correspondances-code-insee-code-postal	7ee82e74e059b443df18bb79fc5a19b1f05e5a88	{'code_comm': '133', 'nom_dept': 'SEINE-ET-MAR...	{'type': 'Point', 'coordinates': [3.0529405055...	2016-09-21T00:29:06.175+02:00
2	correspondances-code-insee-code-postal	e2cd3186f07286705ed482a10b6aebd9de633c81	{'code_comm': '378', 'nom_dept': 'ESSONNE', 's...	{'type': 'Point', 'coordinates': [2.1971816504...	2016-09-21T00:29:06.175+02:00
3	correspondances-code-insee-code-postal	868bf03527a1d0a9defe5cf4e6fa0a730d725699	{'code_comm': '243', 'nom_dept': 'SEINE-ET-MAR...	{'type': 'Point', 'coordinates': [2.7097808131...	2016-09-21T00:29:06.175+02:00
4	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

Table 2: Paris city boroughs data frame

b. Data pre-processing

For London, We replace the spaces with underscores in the title.The borough column has numbers within square brackets that we remove using:

```
wiki_london_data.rename(columns=lambda x: x.strip().replace(" ", "_"), inplace=True)
wiki_london_data['London\xa0borough'] = wiki_london_data['London\xa0borough'].map(lambda x: x.rstrip(']').rstrip('0123456789').rstrip('['))
```

For Paris, we break down each of the nested fields and create the dataframe that we need:

```
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()
```

c. Feature Selection

For both of our datasets, we need only the borough, neighborhood, postal codes and geolocations (latitude and longitude). So we end up selecting the columns that we need by:

```
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']]
```

d. Feature Engineering

Both of our Datasets actually contain information related to all the cities in the country. We can narrow down and further process the data by selecting only the neighbourhoods pertaining to 'London' and 'Paris'. Looking over our London dataset, we can see that we don't have the geolocation data. We need to extrapolate the missing data for our neighbourhoods. We perform this by leveraging the ArcGIS API. With the Help of ArcGIS API, we can get the latitude and longitude of our London neighbourhood data. Defining London arcgis geocode function to return latitude and longitude. Passing postal codes of London to get the geographical co-ordinates. Extracting the latitude from our previously collected coordinates.

London:

```
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
```

Paris:

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

Final data frames will be as below:

	postal_code	nom_comm	nom_dept	Latitude	Longitude
0	75009	PARIS-9E-ARRONDISSEMENT	PARIS	48.876896	2.337460
1	75002	PARIS-2E-ARRONDISSEMENT	PARIS	48.867903	2.344107
2	75011	PARIS-11E-ARRONDISSEMENT	PARIS	48.859415	2.378741
3	75003	PARIS-3E-ARRONDISSEMENT	PARIS	48.863054	2.359361
4	75006	PARIS-6E-ARRONDISSEMENT	PARIS	48.848968	2.332671
5	75019	PARIS-19E-ARRONDISSEMENT	PARIS	48.886869	2.384694
6	75020	PARIS-20E-ARRONDISSEMENT	PARIS	48.863187	2.400820
7	75015	PARIS-15E-ARRONDISSEMENT	PARIS	48.840155	2.293559
8	75005	PARIS-5E-ARRONDISSEMENT	PARIS	48.844509	2.349859
9	75007	PARIS-7E-ARRONDISSEMENT	PARIS	48.856083	2.312439
10	75008	PARIS-8E-ARRONDISSEMENT	PARIS	48.872527	2.312583
11	75013	PARIS-13E-ARRONDISSEMENT	PARIS	48.828718	2.362468
12	75012	PARIS-12E-ARRONDISSEMENT	PARIS	48.835156	2.419807
13	75018	PARIS-18E-ARRONDISSEMENT	PARIS	48.892735	2.348712

	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

310 rows × 5 columns

Table 3: Paris featured engineered data frame(left) and London featured engineered data frame(Right)

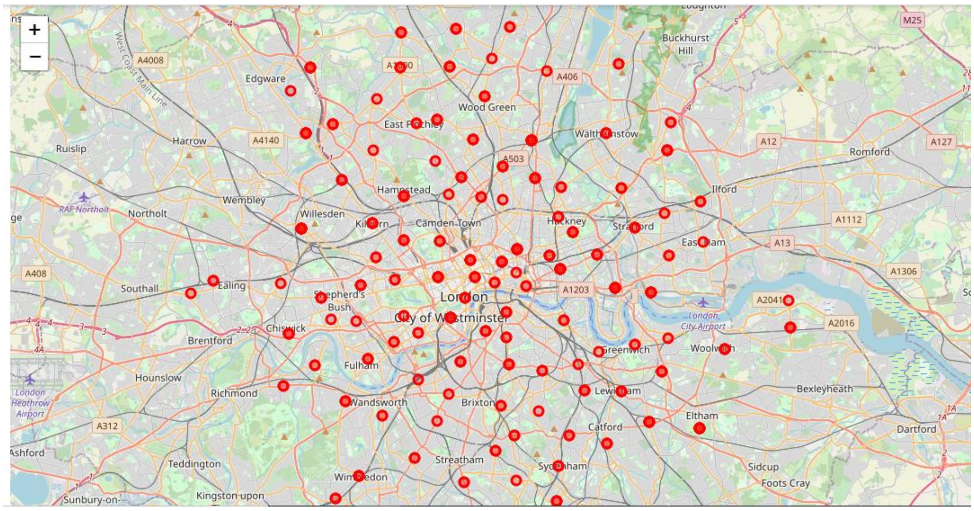
e. Visualizing the neighborhoods

i. London

To help visualize the Map of London and the neighbourhoods in London, we make use of the folium package.

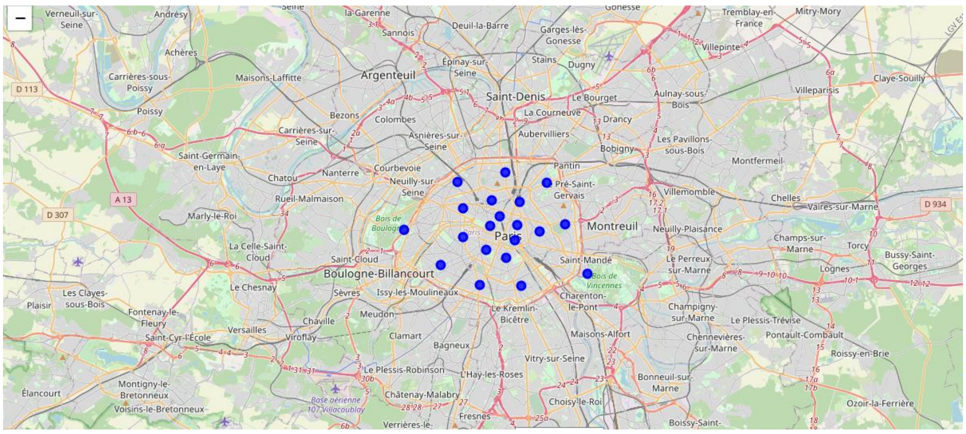
```
[50] # Creating the map of London
     map_London = folium.Map(location=[london_lat_coords, london_lng_coords], zoom_start=12)
     map_London

     # adding markers to map
     for latitude, longitude, borough, town in zip(london_merged['latitude'], london_merged['longitude'], london_merged['borough'], london_merged['town']):
         label = '{}, {}'.format(town, borough)
         label = folium.Popup(label, parse_html=True)
         folium.CircleMarker(
             [latitude, longitude],
             radius=5,
             popup=label,
             color='red',
             fill=True
             ).add_to(map_London)

     map_London
```

Map:



ii. Paris

Applying the above function to Paris neighborhood, we will see below map



Top venue categories of neighborhoods in London:

Defining a function to get the neraby venues in the neighbourhood. This will help us get venue categories which is important for our analysis

```
[75] 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={}'.for
             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']['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)
```

Getting the venues in London:

```
venues_in_London = getNearbyVenues(london_merged['borough'], london_merged['latitude'], london_merged['longitude'])
```

Venue categories:

```
venues_in_London.groupby('Venue Category').max()
```

Top most common venue categories can be retrieved by defining a function

```
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]
```

Above concepts are applied for retrieving the top venue categories of neighborhoods in Paris:

Reusing our previously defined function to get the top venue categories in the neighbourhoods of Paris.

```
[95] # create a new dataframe for Paris
     neighborhoods_venues_sorted_paris = pd.DataFrame(columns=columns)
     neighborhoods_venues_sorted_paris['Neighbourhood'] = Paris_grouped['Neighbourhood']

     for ind in np.arange(Paris_grouped.shape[0]):
         neighborhoods_venues_sorted_paris.iloc[ind, 1:] = return_most_common_venues(Paris_grouped.iloc[ind, :], num_top_venues)

     neighborhoods_venues_sorted_paris.head()
```

Top venue locations in London:

	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	Italian Restaurant	Pub	Bus Stop	Supermarket	Pizza Place	Sushi Restaurant	Pharmacy
1	Barnet, Brent, Camden	Bus Station	Gym / Fitness Center	Clothing Store	Supermarket	Zoo Exhibit	Fish & Chips Shop	Falafel Restaurant	Farmers Market	Fast Food Restaurant	Filipino Restaurant
2	Bexley	Supermarket	Historic Site	Pub	Coffee Shop	Train Station	Convenience Store	Golf Course	Construction & Landscaping	Park	Bus Stop
3	Bexley, Greenwich	Golf Course	Sports Club	Bus Stop	Convenience Store	Construction & Landscaping	Historic Site	Park	Film Studio	Exhibit	Falafel Restaurant
4	Bexley, Greenwich	Supermarket	Pub	Train Station	Coffee Shop	Convenience Store	Historic Site	Film Studio	Exhibit	Falafel Restaurant	Farmers Market

Top Venue locations in Paris:

	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
0	PARIS-10E-ARRONDISSEMENT	French Restaurant	Coffee Shop	Hotel	Bistro	Café	Japanese Restaurant	Italian Restaurant	Pizza Place	Asian Restaurant
1	PARIS-11E-ARRONDISSEMENT	Restaurant	Café	French Restaurant	Italian Restaurant	Vegetarian / Vegan Restaurant	Cocktail Bar	Bakery	Bistro	Pastry Shop
2	PARIS-12E-ARRONDISSEMENT	Zoo Exhibit	Supermarket	Bistro	Monument / Landmark	Zoo	Indie Movie Theater	Escape Room	Fountain	Food & Drink Shop
3	PARIS-13E-ARRONDISSEMENT	Vietnamese Restaurant	Asian Restaurant	Thai Restaurant	Chinese Restaurant	French Restaurant	Juice Bar	Hotel	Dessert Shop	Coffee Shop
4	PARIS-14E-ARRONDISSEMENT	French Restaurant	Hotel	Japanese Restaurant	Food & Drink Shop	Café	Bistro	Bakery	Fast Food Restaurant	Pizza Place

8. Model Building

a. Model Building for London neighborhoods

We use K-Means clustering algorithm for building our model

```
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)
kmeans_london
```

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=5, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=0, tol=0.0001, verbose=0)
```

Labelling the data:

```
neighborhoods_venues_sorted_london.insert(0, 'Cluster Labels', kmeans_london.labels_ +1)
```

Merging the data:

```
london_data = london_merged

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

london_data.head()
```

b. Model building for Paris neighborhood

We use K-Means clustering algorithm for the paris neighborhoods as well.

```
# set number of clusters
k_num_clusters = 5

Paris_grouped_clustering = Paris_grouped.drop('Neighbourhood', 1)

# run k-means clustering
kmeans_Paris = KMeans(n_clusters=k_num_clusters, random_state=0).fit(Paris_grouped_clustering)
kmeans_Paris
```

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=5, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=0, tol=0.0001, verbose=0)
```

Labelling the data and merging to form data frame

```
kmeans_Paris.labels_
```

```
array([2, 2, 3, 4, 1, 2, 0, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 2], dtype=int32)
```

Join paris_combined_data with our neighbourhood venues sorted to add latitude & longitude for each of the neighborhood to prepare it for plotting

```
paris_data = paris_combined_data

paris_data = paris_data.join(neighborhoods_venues_sorted_paris.set_index('Neighbourhood'), on='nom_comm')

paris_data.head()
```

London Neighborhood dataframe

	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 Most Common Venue
0	Bexley, Greenwich	LONDON	SE2	51.49245	0.12127	3	Supermarket	Pub	Train Station	Coffee Shop	Convenience Store	Historic Site	Film Studio	Exhibit
1	Ealing, Hammersmith and Fulham	LONDON	W3, W4	51.51324	-0.26746	4	Grocery Store	Train Station	Breakfast Spot	Park	Indian Restaurant	Dessert Shop	Exhibit	Dance Studio
6	City	LONDON	EC3	51.51200	-0.08058	1	Hotel	Coffee Shop	Italian Restaurant	Gym / Fitness Center	Pub	Restaurant	Sandwich Place	Scenic Lookout
7	Westminster	LONDON	WC2	51.51651	-0.11968	1	Hotel	Coffee Shop	Café	Sandwich Place	Pub	Italian Restaurant	Theater	Restaurant
9	Bromley	LONDON	SE20	51.41009	-0.05683	3	Supermarket	Convenience Store	Hotel	Grocery Store	Fast Food Restaurant	Park	Gastropub	Bistro

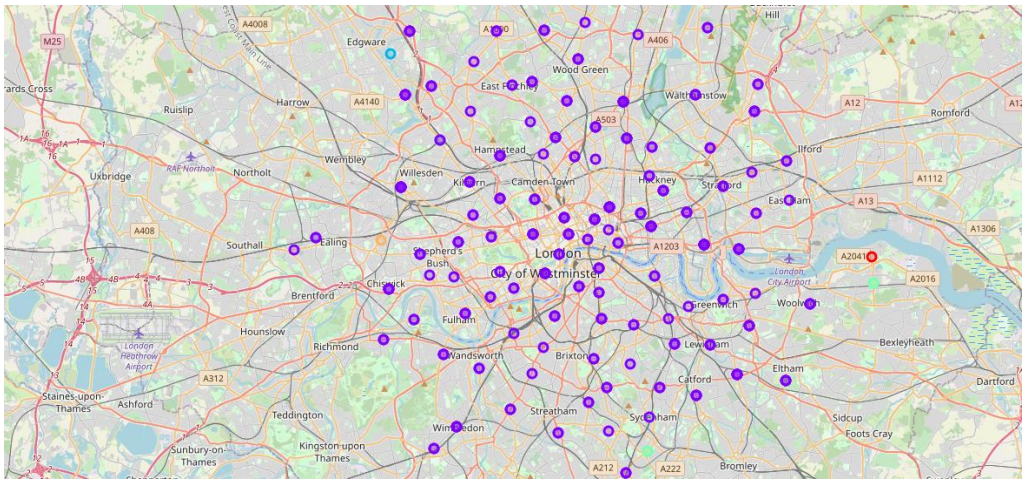
Paris Neighborhood dataframe

	postal_code	nom_comm	nom_dept	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
0	75009	PARIS-9E-ARRONDISSEMENT	PARIS	48.876896	2.337460	3	French Restaurant	Hotel	Japanese Restaurant	Bistro	Bakery	Lounge	Wine Bar
1	75002	PARIS-2E-ARRONDISSEMENT	PARIS	48.867903	2.344107	3	French Restaurant	Cocktail Bar	Italian Restaurant	Wine Bar	Bakery	Bistro	Bar
2	75011	PARIS-11E-ARRONDISSEMENT	PARIS	48.859415	2.378741	3	Restaurant	Café	French Restaurant	Italian Restaurant	Vegetarian / Vegan Restaurant	Cocktail Bar	Bakery
3	75003	PARIS-3E-ARRONDISSEMENT	PARIS	48.863054	2.359361	3	Coffee Shop	Bakery	French Restaurant	Burger Joint	Bistro	Japanese Restaurant	Gourmet Shop
4	75006	PARIS-6E-ARRONDISSEMENT	PARIS	48.848968	2.332671	3	French Restaurant	Chocolate Shop	Bistro	Bakery	Pub	Fountain	Italian Restaurant

9. Results and Discussions

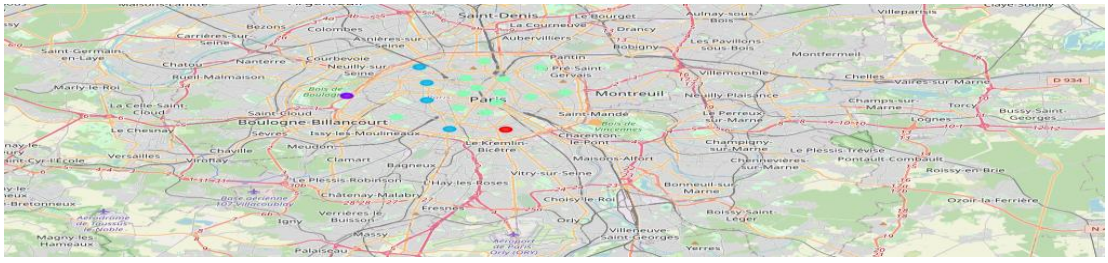
Plotting the clusters on London map:

```
map_clusters_london = folium.Map(location=[london_lat_coords, london_lng_coords], zoom_start=12)

# set color scheme for the clusters
x = np.arange(k_num_clusters)
ys = [i + x + (i*x)**2 for i in range(k_num_clusters)]
colors_array = cm.rainbow(np.linspace(0, 1, len(ys)))
rainbow = [colors.rgb2hex(i) for i in colors_array]

# add markers to the map
markers_colors = []
for lat, lon, poi, cluster in zip(london_data_nonan['latitude'], london_data_nonan['longitude'], london_data_nonan['borough'], london_data_nonan['Cluster Labels']
    label = folium.Popup('Cluster ' + str(int(cluster) +1) + '\n' + str(poi) , parse_html=True)
    folium.CircleMarker(
        [lat, lon],
        radius=5,
        popup=label,
        color=rainbow[int(cluster-1)],
        fill=True,
        fill_color=rainbow[int(cluster-1)]
        ).add_to(map_clusters_london)

map_clusters_london
```



Plotting the clusters on Paris map:

```
map_clusters_paris = folium.Map(location=[paris_lat_coords, paris_lng_coords], zoom_start=12)

# set color scheme for the clusters
x = np.arange(k_num_clusters)
ys = [i + x + (i*x)**2 for i in range(k_num_clusters)]
colors_array = cm.rainbow(np.linspace(0, 1, len(ys)))
rainbow = [colors.rgb2hex(i) for i in colors_array]

# add markers to the map
markers_colors = []
for lat, lon, poi, cluster in zip(paris_data_nonan['Latitude'], paris_data_nonan['Longitude'], paris_data_nonan['nom_comm'], paris_data_nonan['Cluster Labels']):
    label = folium.Popup('Cluster ' + str(int(cluster) +1) + ' ' + str(poi) , parse_html=True)
    folium.CircleMarker(
        [lat, lon],
        radius=5,
        popup=label,
        color=rainbow[int(cluster-1)],
        fill=True,
        fill_color=rainbow[int(cluster-1)],
        fill_opacity=0.8
        ).add_to(map_clusters_paris)

map_clusters_paris
```



Clusters Results of London:

Cluster 1:

Cluster 1:

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

	town	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 Most Common Venue	9th Most Common Venue	10th Most Common Venue
6	LONDON	1	Hotel	Coffee Shop	Italian Restaurant	Gym / Fitness Center	Pub	Restaurant	Sandwich Place	Scenic Lookout	French Restaurant	Wine Bar
7	LONDON	1	Hotel	Coffee Shop	Café	Sandwich Place	Pub	Italian Restaurant	Theater	Restaurant	Burger Joint	Bakery
10	LONDON	1	Coffee Shop	Pub	Café	Food Truck	Vietnamese Restaurant	Italian Restaurant	Gym / Fitness Center	Park	Cocktail Bar	Breakfast Spot
12	LONDON	1	Coffee Shop	Pub	Café	Food Truck	Vietnamese Restaurant	Italian Restaurant	Gym / Fitness Center	Park	Cocktail Bar	Breakfast Spot
14	BARNET, LONDON	1	Coffee Shop	Café	Grocery Store	Italian Restaurant	Pub	Bus Stop	Supermarket	Pizza Place	Sushi Restaurant	Pharmacy
...	...	...	...	...	...	...	...	...	...	...	...	...

Cluster 2:

Cluster 2

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

	town	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 Most Common Venue	9th Most Common Venue	10th Most Common Venue
379	HARROW, STANMOREEDGWARE, LONDON	2	Gym	Metro Station	Bakery	Food Stand	Food Court	Food & Drink Shop	Flower Shop	Flea Market	Fishing Store	Event Space

Cluster 3:

Cluster 3

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

	town	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 Most Common Venue	9th Most Common Venue	10th Most Common Venue
0	LONDON	3	Supermarket	Pub	Train Station	Coffee Shop	Convenience Store	Historic Site	Film Studio	Exhibit	Falafel Restaurant	Farmers Market
9	LONDON	3	Supermarket	Convenience Store	Hotel	Grocery Store	Fast Food Restaurant	Park	Gastropub	Bistro	Sandwich Place	Golf Course
29	BECKENHAM, LONDON	3	Supermarket	Convenience Store	Hotel	Grocery Store	Fast Food Restaurant	Park	Gastropub	Bistro	Sandwich Place	Golf Course
45	BEXLEYHEATH, LONDON	3	Supermarket	Historic Site	Pub	Coffee Shop	Train Station	Convenience Store	Golf Course	Construction & Landscaping	Park	Bus Stop
121	LONDON	3	Bus Station	Gym / Fitness Center	Clothing Store	Supermarket	Zoo Exhibit	Fish & Chips Shop	Falafel Restaurant	Farmers Market	Fast Food Restaurant	Filipino Restaurant
124	LONDON	3	Supermarket	Historic Site	Pub	Coffee Shop	Train Station	Convenience Store	Golf Course	Construction & Landscaping	Park	Bus Stop

Cluster 4:

Cluster 4

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

	town	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 Most Common Venue	9th Most Common Venue	10th Most Common Venue
1	LONDON	4	Grocery Store	Train Station	Breakfast Spot	Park	Indian Restaurant	Dessert Shop	Exhibit	Dance Studio	Food Stand	Food Court

Cluster 5:

Cluster 5

+ Code + Text

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

	town	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 Most Common Venue	9th Most Common Venue	10th Most Common Venue
168	LONDON, WELLING	5	Golf Course	Sports Club	Bus Stop	Convenience Store	Construction & Landscaping	Historic Site	Park	Film Studio	Exhibit	Falafel Restaurant
459	LONDON, ERITH	5	Golf Course	Sports Club	Bus Stop	Convenience Store	Construction & Landscaping	Historic Site	Park	Film Studio	Exhibit	Falafel Restaurant

Clusters results on Paris:

Cluster 1

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

	nom_comm	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 Most Common Venue	9th Most Common Venue	10th Most Common Venue
16	PARIS-16E-ARRONDISSEMENT	1	Plaza	Lake	French Restaurant	Art Museum	Bus Stop	Bus Station	Park	Boat or Ferry	Pool	Ethiopian Restaurant

Cluster 2

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

	nom_comm	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 Most Common Venue	9th Most Common Venue	10th Most Common Venue
9	PARIS-7E-ARRONDISSEMENT	2	French Restaurant	Hotel	Café	Italian Restaurant	Plaza	Cocktail Bar	Coffee Shop	Art Museum	History Museum	Bistro
10	PARIS-8E-ARRONDISSEMENT	2	French Restaurant	Hotel	Spa	Art Gallery	Theater	Cocktail Bar	Coffee Shop	Resort	Bar	Park
18	PARIS-17E-ARRONDISSEMENT	2	French Restaurant	Hotel	Italian Restaurant	Café	Bakery	Japanese Restaurant	Plaza	Bistro	Restaurant	Burger Joint
19	PARIS-14E-ARRONDISSEMENT	2	French Restaurant	Hotel	Japanese Restaurant	Food & Drink Shop	Café	Bistro	Bakery	Fast Food Restaurant	Pizza Place	Plaza

Cluster 3

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

	nom_comm	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 Most Common Venue	9th Most Common Venue	10th Most Common Venue
0	PARIS-9E-ARRONDISSEMENT	3	French Restaurant	Hotel	Japanese Restaurant	Bistro	Bakery	Lounge	Wine Bar	Bar	Vegetarian / Vegan Restaurant	Gym / Fitness Center
1	PARIS-2E-ARRONDISSEMENT	3	French Restaurant	Cocktail Bar	Italian Restaurant	Wine Bar	Bakery	Bistro	Bar	Thai Restaurant	Salad Place	Hotel
2	PARIS-11E-ARRONDISSEMENT	3	Restaurant	Café	French Restaurant	Italian Restaurant	Vegetarian / Vegan Restaurant	Cocktail Bar	Bakery	Bistro	Pastry Shop	Afghan Restaurant
3	PARIS-3E-ARRONDISSEMENT	3	Coffee Shop	Bakery	French Restaurant	Burger Joint	Bistro	Japanese Restaurant	Gourmet Shop	Italian Restaurant	Café	Sandwich Place
4	PARIS-6E-ARRONDISSEMENT	3	French Restaurant	Chocolate Shop	Bistro	Bakery	Pub	Fountain	Italian Restaurant	Athletics & Sports	Plaza	Restaurant
5	PARIS-19E-ARRONDISSEMENT	3	French Restaurant	Bar	Hotel	Bistro	Beer Bar	Brewery	Supermarket	Seafood Restaurant	Restaurant	Steakhouse

Cluster 4

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

	nom_comm	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 Most Common Venue	9th Most Common Venue	10th Most Common Venue
12	PARIS-12E-ARRONDISSEMENT	4	Zoo Exhibit	Supermarket	Bistro	Monument / Landmark	Zoo	Indie Movie Theater	Escape Room	Fountain	Food & Drink Shop	Flower Shop

Cluster 5

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

	nom_comm	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 Most Common Venue	9th Most Common Venue	10th Most Common Venue
11	PARIS-13E-ARRONDISSEMENT	5	Vietnamese Restaurant	Asian Restaurant	Thai Restaurant	Chinese Restaurant	French Restaurant	Juice Bar	Hotel	Dessert Shop	Coffee Shop	Gourmet Shop

The neighbourhoods of London are very mulitcultural. There are a lot of different cusines including Indian, Italian, Turkish and Chinese. London seems to take a step further in this direction by having a lot of Restaurants, bars, juice bars, coffee shops, Fish and Chips shop and Breakfast spots. It has a lot of shopping options too with that of the Flea markets, flower shops, fish markets, Fishing stores, clothing stores. The main modes of transport seem to be Buses and trains. For leisure, the neighbourhoods 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 is relatively small in size geographically. It has a wide variety of cusines and eateries including French, Thai, Cambodian, Asian, Chinese etc. 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 cusines to try out.

10. Conclusions

The purpose of this project was to explore the cities of London and Paris and 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 neighbourhoods finally concluding with clustering similar neighbourhoods together.

We could see that each of the neighbourhoods in both the cities have a wide variety of experiences to offer which is unique in its 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 gateway with a lot of places to explore, beautiful landscapes and a wide variety of culture. Overall, it's up to the stakeholders to decide which experience they would prefer more and which would more to their liking.

The capstone project provided a medium to understand in depth about how real life data science projects work and what all steps go in building a data science methodology. All steps from understanding the business problem, data understanding to data preparation, and model building were discussed in detail here. Many drawbacks of the current analysis and further ways to improve the analysis were also mentioned. This was an initial attempt to understand and solve the business problem at hand. However, there still exists a huge potential to extend this project in real life scenarios.

References

1. https://en.wikipedia.org/wiki/List_of_London_boroughs
2. https://en.wikipedia.org/wiki/Arrondissements_of_Paris
3. https://foursquare.com/
4. https://developers.arcgis.com/python/
5. https://python-visualization.github.io/folium/modules.html
6. https://pandas.pydata.org/pandas-docs/stable/
7. https://scikit-learn.org/stable/index.html
8. https://medium.com/@yrnigam/how-to-write-a-data-science-report-181bd49d8f4d
9. https://towardsdatascience.com/a-tale-of-two-cities-e693c15b3ddb
10. https://www.coursera.org/lecture/historical-fiction/dickens-and-the-french-revolution-a-tale-of-two-cities-oVjhu