A Tale of Two cities - Clustering the Neighbourhoods of London and Paris

1. Introduction

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 neighbourhoods of London and Paris respectively and draw insights to what they look like now.

2. Business Problem

The aim is to help tourists choose their destinations depending on the experiences that the neighbourhoods have to offer and what they would want to have. This also helps people make decisions if they are thinking about migrating to London or Paris or even if they want to relocate neighbourhoods within the city. Our findings will help stakeholders make informed decisions and address any concerns they have including the different kinds of cuisines, provision stores and what the city has to offer.

3. 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.

3.1 London

To derive our solution, We scrape our data from https://en.wikipedia.org/wiki/List_of_areas_of_London

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

1. *borough* : Name of Neighbourhood
2. *town* : Name of borough
3. *post_code* : Postal codes for London.

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

3.2 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 Neighbourhood
2. *longitude* : Longitude for Neighbourhood

3.3 Paris

To derive our solution, We leverage JSON data available at https://www.data.gouv.fr/fr/datasets/r/e88c6fda-1d09-42a0-a069-606d3259114e

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

1. *postal_code* : Postal codes for France
2. *nom_comm* : Name of Neighbourhoods in France
3. *nom_dept* : Name of the boroughs, equivalent to towns in France
4. *geo_point_2d* : Tuple containing the latitude and longitude of the Neighbourhoods.

4. Methodology

We will be creating our model with the help of Python so we start off by importing all the required packages.

```
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:

- *Pandas* : To collect and manipulate data in JSON and HTMl and then data analysis
- *requests* : Handle http requests
- *matplotlib* : Detailing the generated maps
- *folium* : Generating maps of London and Paris
- *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.

4.1 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_Lon
don"
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
```

The data looks like this:

	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

To collect data for Paris, we download the JSON file containg all the postal codes of France from https://www.data.gouv.fr/fr/datasets/r/e88c6fda-1d09-42a0-a069-606d3259114e

Using Pandas we load the table after reading the JSON file:

```
!wget -q -O 'france-data.json' https://www.data.gouv.fr/fr/data
sets/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

4.2 Data Preprocessing

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['borough'] = wiki_london_data['borough'].map(la
mbda 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()
```

4.3 Feature Selection

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

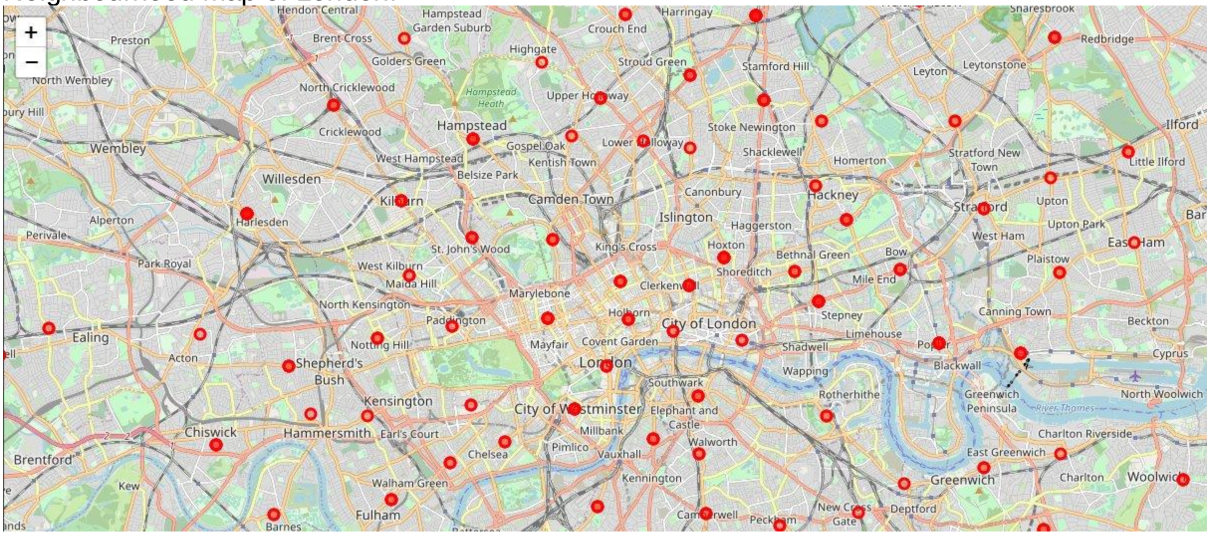
```
df1 = wiki_london_data.drop( [ wiki_london_data.columns[0], wiki
_london_data.columns[4], wiki_london_data.columns[5] ], axis=1)

df_2 = paris_field_data[['postal_code','nom_comm','nom_dept','ge
o_point_2d']]
```

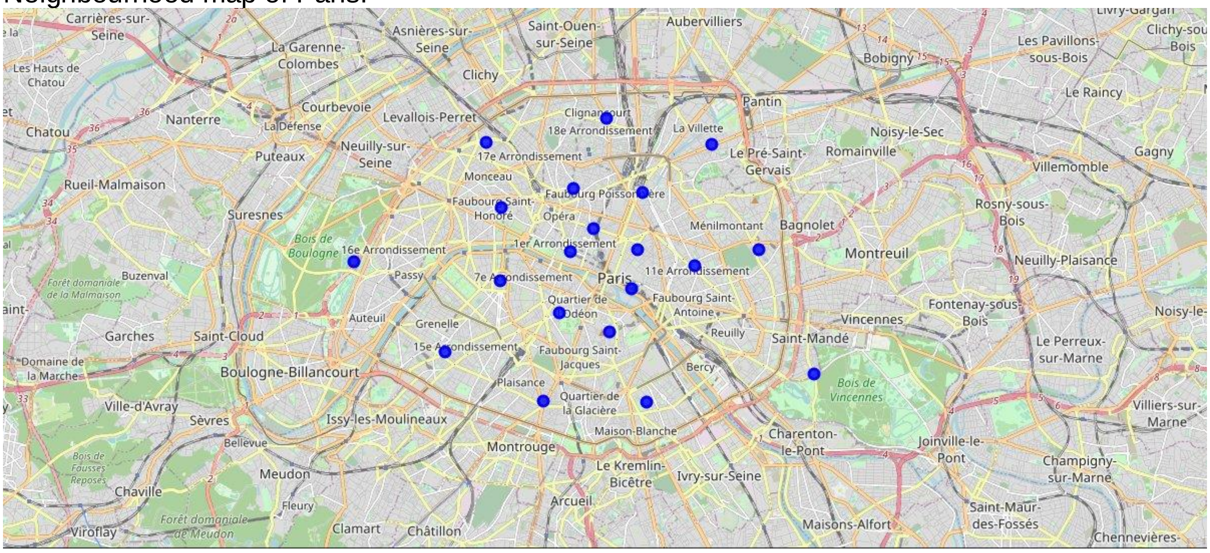
4.4 Visualizing the Neighbourhoods of London and Paris

Now that our datasets are ready, using the `Folium` package, we can visualize the maps of London and Paris with the neighbourhoods that we collected.

Neighbourhood map of London:



Neighbourhood map of Paris:



Now that we have visualized the neighbourhoods, we need to find out what each neighbourhood is like and what are the common venue and venue categories within a 500m radius.

This is where `Foursquare` comes into play. With the help of `Foursquare` we define a function which collects information pertaining to each neighbourhood including that of the name of the neighbourhood, geo-coordinates, venue and venue categories.

Resulting data looks 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

4.5 Top Venues in the Neighbourhoods

In our next step, We need to rank and label the top venue categories in our neighborhood.

Let's define a function to get the top venue categories in the neighbourhood

```
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]
```

There are many categories, we will consider top 10 categories to avoid data skew.

Defining a function to label them accurately

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

Getting the top venue categories in the neighbourhoods of London

```
# create a new dataframe for London
neighborhoods_venues_sorted_london = pd.DataFrame(columns=column
s)
neighborhoods_venues_sorted_london['Neighbourhood'] = London_gro
uped['Neighbourhood']

for ind in np.arange(London_grouped.shape[0]):
    neighborhoods_venues_sorted_london.iloc[ind, 1:] = return_mo
st_common_venues(London_grouped.iloc[ind, :], num_top_venues)

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

4.6 Model Building - KMeans

Moving on to the most exicitng part - **Model Building!** We will be using KMeans Clustering Machine learning algorithm to cluster similar neighbourhoods together. 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)
```

Our model has labelled each of the neighbourhoods, we add the label into our dataset.

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

We then join London_merged with our neighbourhood 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_londo
n.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 Comr Venu
0	Bexley, Greenwich	LONDON	SE2	51.49245	0.12127	4	Supermarket	Platform	Convenience Store	Historic Site	Train Station	Coffee Shop	Zoo Exhibit	Film Studic
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 I
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

4.7 Visualizing the clustered Neighbourhoods

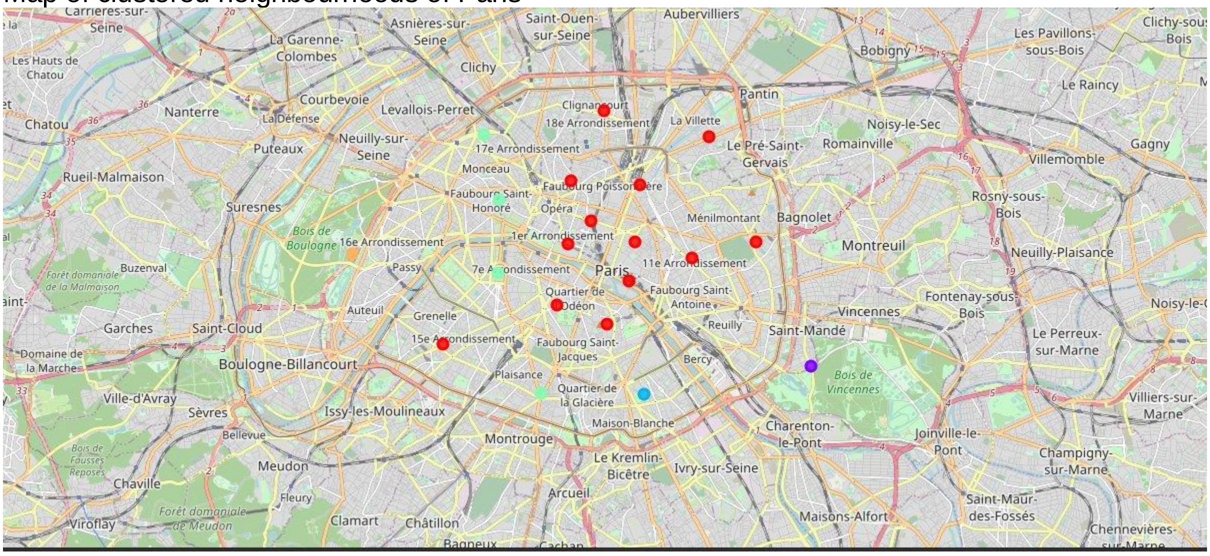
Our data is processed, missing data is collected and compiled. The Model is built. All that's remaining is to see the clustered neighbourhoods on the map. Again, we use `Folium` package to do so.

We drop all the NaN values to prevent data skew

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

Map of clustered neighbourhoods of London:

Map of clustered neighbourhoods of Paris



5. Results and Discussion

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.

6. Conclusion

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 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, beautiful landscapes, amazing food and a wide variety of culture. Overall, it's upto the stakeholders to decide which experience they would prefer more and which would more to their liking.