A Tale of Two cities

Clustering the Neighbourhoods of London and Paris

Himani Bali

10th November 2020

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.

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.

Data Description

We require geolocation 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 neighbourhoods, boroughs, venues and their most popular venue categories.

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.

borough : Name of Neighbourhood

town : Name of borough

post\_code : Postal codes for London.

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

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.

latitude : Latitude for Neighbourhood

longitude : Longitude for Neighbourhood

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.

postal\_code : Postal codes for France

nom\_comm : Name of Neighbourhoods in France

nom\_dept : Name of the boroughs, equivalent to towns in France

geo\_point\_2d : Tuple containing the latitude and longitude of the Neighbourhoods.

Foursquare API Data

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:

Neighbourhood : Name of the Neighbourhood

Neighbourhood Latitude : Latitude of the Neighbourhood

Neighbourhood Longitude : Longitude of the Neighbourhood

Venue : Name of the Venue

Venue Latitude : Latitude of Venue

Venue Longitude : Longitude of Venue

Venue Category : Category of Venue

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.

Methodology

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

In [1]:

import pandas as pd

import requests

import numpy as np

import matplotlib.cm as cm

import matplotlib.colors as colors

import folium

# import k-means for the clustering stage

from sklearn.cluster import KMeans

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.

Exploring London

Neighbourhoods of London

We begin to start collecting and refining the data needed for the our business solution to work.

Data Collection

To get the neighbourhoods in london, we start by scraping the list of areas of london wiki page.

In [2]:

url\_london = "https://en.wikipedia.org/wiki/List\_of\_areas\_of\_London"

wiki\_london\_url = requests.get(url\_london)

wiki\_london\_url

Out[2]:

<Response [200]>

Response 200 means that we are able to make the connection

In [3]:

wiki\_london\_data = pd.read\_html(wiki\_london\_url.text)

wiki\_london\_data

Out[3]:

[ 0

0 Map all coordinates in "Category:Areas of Lond...

1 Download coordinates as: KML · GPX,

Location London borough ... Dial code OS grid ref

0 Abbey Wood Bexley, Greenwich [7] ... 020 TQ465785

1 Acton Ealing, Hammersmith and Fulham[8] ... 020 TQ205805

2 Addington Croydon[8] ... 020 TQ375645

3 Addiscombe Croydon[8] ... 020 TQ345665

4 Albany Park Bexley ... 020 TQ478728

.. ... ... ... ... ...

528 Woolwich Greenwich ... 020 TQ435795

529 Worcester Park Sutton, Kingston upon Thames ... 020 TQ225655

530 Wormwood Scrubs Hammersmith and Fulham ... 020 TQ225815

531 Yeading Hillingdon ... 020 TQ115825

532 Yiewsley Hillingdon ... 020 TQ063804

[533 rows x 6 columns],

0 1

0 NaN Wikimedia Commons has media related to Distric...,

vteAreas of London vteAreas of London.1

0 Central activities zone Bloomsbury City of London wards Holborn Maryle...

1 Town centrenetwork International Knightsbridge West End Metropoli...

2 International Knightsbridge West End

3 Metropolitan Bromley Croydon Ealing Harrow Hounslow Ilford ...

4 Major Angel Barking Bexleyheath Brixton Camden Town ...

5 Districts(principal) Acton Beckenham Belgravia Bethnal Green Brentf...

6 Neighbourhoods(principal) Abbey Wood Alperton Anerley Archway Barnes Bar...

7 Lists of areasby borough Barking and Dagenham Barnet Bexley Brent Broml...

8 Fictional Canley (borough) (The Bill: TV soap) Charnham ...,

0 1

0 International Knightsbridge West End

1 Metropolitan Bromley Croydon Ealing Harrow Hounslow Ilford ...

2 Major Angel Barking Bexleyheath Brixton Camden Town ...

3 Districts(principal) Acton Beckenham Belgravia Bethnal Green Brentf...

4 Neighbourhoods(principal) Abbey Wood Alperton Anerley Archway Barnes Bar...]

Scraping the webpage gives us all the tables present on the page. We need the 2nd table, so selecting the 2nd table.

In [4]:

wiki\_london\_data = wiki\_london\_data[1]

wiki\_london\_data

Out[4]:

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

Data Preprocessing

we remove the spaces in the column titles and then we add \_ between words.

In [5]:

wiki\_london\_data.rename(columns=lambda x: x.strip().replace(" ", "\_"), inplace=True)

wiki\_london\_data

Out[5]:

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

We see that few columns have no '\_' between the words despite applying our function meaning that there are special characters

Feature Selection

We need only the boroughs, Postal codes, Post town for further steps. We can drop the locations, dial codes and OS grid.

In [6]:

df1 = wiki\_london\_data.drop( [ wiki\_london\_data.columns[0], wiki\_london\_data.columns[4], wiki\_london\_data.columns[5] ], axis=1)

In [7]:

df1.head()

Out[7]:

London borough Post\_town Postcode district

0 Bexley, Greenwich [7] LONDON SE2

1 Ealing, Hammersmith and Fulham[8] LONDON W3, W4

2 Croydon[8] CROYDON CR0

3 Croydon[8] CROYDON CR0

4 Bexley BEXLEY, SIDCUP DA5, DA14

let's rename the Postcode district column and the london borough to something simpler

In [8]:

df1.columns = ['borough','town','post\_code']

df1

Out[8]:

borough town post\_code

0 Bexley, Greenwich [7] LONDON SE2

1 Ealing, Hammersmith and Fulham[8] LONDON W3, W4

2 Croydon[8] CROYDON CR0

3 Croydon[8] CROYDON CR0

4 Bexley BEXLEY, SIDCUP DA5, DA14

... ... ... ...

528 Greenwich LONDON SE18

529 Sutton, Kingston upon Thames WORCESTER PARK KT4

530 Hammersmith and Fulham LONDON W12

531 Hillingdon HAYES UB4

532 Hillingdon WEST DRAYTON UB7

533 rows × 3 columns

Let's remove the Square brackets [ ] and numbers from the borough column

In [9]:

df1['borough'] = df1['borough'].map(lambda x: x.rstrip(']').rstrip('0123456789').rstrip('['))

df1

Out[9]:

borough town post\_code

0 Bexley, Greenwich LONDON SE2

1 Ealing, Hammersmith and Fulham LONDON W3, W4

2 Croydon CROYDON CR0

3 Croydon CROYDON CR0

4 Bexley BEXLEY, SIDCUP DA5, DA14

... ... ... ...

528 Greenwich LONDON SE18

529 Sutton, Kingston upon Thames WORCESTER PARK KT4

530 Hammersmith and Fulham LONDON W12

531 Hillingdon HAYES UB4

532 Hillingdon WEST DRAYTON UB7

533 rows × 3 columns

Take the dimension of the dataframe

In [10]:

df1.shape

Out[10]:

(533, 3)

We currently have 533 records and 3 columns of our data. It's time to perform Feature Engineering

Feature Engineering

We can only focusing on the neighbourhoods of London, so performing the changes

In [11]:

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

df1

Out[11]:

borough town post\_code

0 Bexley, Greenwich LONDON SE2

1 Ealing, Hammersmith and Fulham LONDON W3, W4

6 City LONDON EC3

7 Westminster LONDON WC2

9 Bromley LONDON SE20

... ... ... ...

523 Redbridge LONDON IG8, E18

524 Redbridge, Waltham Forest LONDON, WOODFORD GREEN IG8

527 Barnet LONDON N12

528 Greenwich LONDON SE18

530 Hammersmith and Fulham LONDON W12

310 rows × 3 columns

In [12]:

df1.shape

Out[12]:

(310, 3)

We now have only 310 rows. We can proceed with our further steps. Getting some descriptive statistics

In [13]:

df1.info()

<class 'pandas.core.frame.DataFrame'>

Int64Index: 310 entries, 0 to 530

Data columns (total 3 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 borough 310 non-null object

1 town 310 non-null object

2 post\_code 310 non-null object

dtypes: object(3)

memory usage: 9.7+ KB

Geolocations of the London Neighbourhoods

ArcGis API

We need to get the geographical co-ordinates for the neighbourhoods to plot out map. We will use the arcgis package to do so.

Arcgis doesn't have a limitation on the number of API calls made so it fits our use case perfectly.

In [14]:

pip install arcgis

Collecting arcgis

Downloading https://files.pythonhosted.org/packages/ff/d4/a50b566132f3f4065ce92f261ecf6a17af22b8a9134edbf6b42fb57fe52f/arcgis-1.8.2.tar.gz (2.0MB)

|████████████████████████████████| 2.0MB 27kB/s

Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from arcgis) (1.15.0)

Requirement already satisfied: ipywidgets>=7 in /usr/local/lib/python3.6/dist-packages (from arcgis) (7.5.1)

Requirement already satisfied: widgetsnbextension>=3 in /usr/local/lib/python3.6/dist-packages (from arcgis) (3.5.1)

Requirement already satisfied: pandas>=1 in /usr/local/lib/python3.6/dist-packages (from arcgis) (1.0.5)

Requirement already satisfied: numpy>=1.16.2 in /usr/local/lib/python3.6/dist-packages (from arcgis) (1.18.5)

Requirement already satisfied: matplotlib in /usr/local/lib/python3.6/dist-packages (from arcgis) (3.2.2)

Collecting keyring>=19

Using cached https://files.pythonhosted.org/packages/59/4d/469272124b46456c425a505a8b5f2d42d2fe95d1181de2d888d684506cf7/keyring-21.3.0-py3-none-any.whl

Collecting lerc

Using cached https://files.pythonhosted.org/packages/f2/20/73c8fa29a4ba8f8cbdb037369e6b61ac9253664a8aa0912b52169192447f/lerc-0.1.0.tar.gz

Collecting jupyterlab

Using cached https://files.pythonhosted.org/packages/82/82/a26289d4088bbb214f43db30e8e3485aee3ef3f232acb17180bbf05393fa/jupyterlab-2.2.5-py3-none-any.whl

Collecting pyshp>=2

Using cached https://files.pythonhosted.org/packages/27/16/3bf15aa864fb77845fab8007eda22c2bd67bd6c1fd13496df452c8c43621/pyshp-2.1.0.tar.gz

Requirement already satisfied: requests in /usr/local/lib/python3.6/dist-packages (from arcgis) (2.23.0)

Requirement already satisfied: requests-oauthlib in /usr/local/lib/python3.6/dist-packages (from arcgis) (1.3.0)

Collecting requests\_toolbelt

Using cached https://files.pythonhosted.org/packages/60/ef/7681134338fc097acef8d9b2f8abe0458e4d87559c689a8c306d0957ece5/requests\_toolbelt-0.9.1-py2.py3-none-any.whl

Collecting requests\_ntlm

Using cached https://files.pythonhosted.org/packages/03/4b/8b9a1afde8072c4d5710d9fa91433d504325821b038e00237dc8d6d833dc/requests\_ntlm-1.1.0-py2.py3-none-any.whl

Requirement already satisfied: ipykernel>=4.5.1 in /usr/local/lib/python3.6/dist-packages (from ipywidgets>=7->arcgis) (4.10.1)

Requirement already satisfied: traitlets>=4.3.1 in /usr/local/lib/python3.6/dist-packages (from ipywidgets>=7->arcgis) (4.3.3)

Requirement already satisfied: ipython>=4.0.0; python\_version >= "3.3" in /usr/local/lib/python3.6/dist-packages (from ipywidgets>=7->arcgis) (5.5.0)

Requirement already satisfied: nbformat>=4.2.0 in /usr/local/lib/python3.6/dist-packages (from ipywidgets>=7->arcgis) (5.0.7)

Requirement already satisfied: notebook>=4.4.1 in /usr/local/lib/python3.6/dist-packages (from widgetsnbextension>=3->arcgis) (5.3.1)

Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.6/dist-packages (from pandas>=1->arcgis) (2018.9)

Requirement already satisfied: python-dateutil>=2.6.1 in /usr/local/lib/python3.6/dist-packages (from pandas>=1->arcgis) (2.8.1)

Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.6/dist-packages (from matplotlib->arcgis) (2.4.7)

Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.6/dist-packages (from matplotlib->arcgis) (0.10.0)

Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.6/dist-packages (from matplotlib->arcgis) (1.2.0)

Requirement already satisfied: importlib-metadata; python\_version < "3.8" in /usr/local/lib/python3.6/dist-packages (from keyring>=19->arcgis) (1.7.0)

Collecting jeepney>=0.4.2; sys\_platform == "linux"

Using cached https://files.pythonhosted.org/packages/79/31/2e8d42727595faf224c6dbb748c32b192e212f25495fe841fb7ce8e168b8/jeepney-0.4.3-py3-none-any.whl

Collecting SecretStorage>=3; sys\_platform == "linux"

Using cached https://files.pythonhosted.org/packages/c3/50/8a02cad020e949e6d7105f5f4530d41e3febcaa5b73f8f2148aacb3aeba5/SecretStorage-3.1.2-py3-none-any.whl

Requirement already satisfied: jinja2>=2.10 in /usr/local/lib/python3.6/dist-packages (from jupyterlab->arcgis) (2.11.2)

Collecting jupyterlab-server<2.0,>=1.1.5

Using cached https://files.pythonhosted.org/packages/b4/eb/560043dcd8376328f8b98869efed66ef68307278406ab99c7f63a34d4ae2/jupyterlab\_server-1.2.0-py3-none-any.whl

Requirement already satisfied: tornado!=6.0.0,!=6.0.1,!=6.0.2 in /usr/local/lib/python3.6/dist-packages (from jupyterlab->arcgis) (5.1.1)

Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.6/dist-packages (from requests->arcgis) (3.0.4)

Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.6/dist-packages (from requests->arcgis) (1.24.3)

Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-packages (from requests->arcgis) (2020.6.20)

Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.6/dist-packages (from requests->arcgis) (2.10)

Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.6/dist-packages (from requests-oauthlib->arcgis) (3.1.0)

Collecting cryptography>=1.3

Using cached https://files.pythonhosted.org/packages/ba/91/84a29d6a27fd6dfc21f475704c4d2053d58ed7a4033c2b0ce1b4ca4d03d9/cryptography-3.0-cp35-abi3-manylinux2010\_x86\_64.whl

Collecting ntlm-auth>=1.0.2

Using cached https://files.pythonhosted.org/packages/ff/84/97c550164b54942b0e908c31ef09d9469f3ba4cd7332a671e2125732f63b/ntlm\_auth-1.5.0-py2.py3-none-any.whl

Requirement already satisfied: jupyter-client in /usr/local/lib/python3.6/dist-packages (from ipykernel>=4.5.1->ipywidgets>=7->arcgis) (5.3.5)

Requirement already satisfied: ipython-genutils in /usr/local/lib/python3.6/dist-packages (from traitlets>=4.3.1->ipywidgets>=7->arcgis) (0.2.0)

Requirement already satisfied: decorator in /usr/local/lib/python3.6/dist-packages (from traitlets>=4.3.1->ipywidgets>=7->arcgis) (4.4.2)

Requirement already satisfied: setuptools>=18.5 in /usr/local/lib/python3.6/dist-packages (from ipython>=4.0.0; python\_version >= "3.3"->ipywidgets>=7->arcgis) (49.2.0)

Requirement already satisfied: pexpect; sys\_platform != "win32" in /usr/local/lib/python3.6/dist-packages (from ipython>=4.0.0; python\_version >= "3.3"->ipywidgets>=7->arcgis) (4.8.0)

Requirement already satisfied: prompt-toolkit<2.0.0,>=1.0.4 in /usr/local/lib/python3.6/dist-packages (from ipython>=4.0.0; python\_version >= "3.3"->ipywidgets>=7->arcgis) (1.0.18)

Requirement already satisfied: simplegeneric>0.8 in /usr/local/lib/python3.6/dist-packages (from ipython>=4.0.0; python\_version >= "3.3"->ipywidgets>=7->arcgis) (0.8.1)

Requirement already satisfied: pygments in /usr/local/lib/python3.6/dist-packages (from ipython>=4.0.0; python\_version >= "3.3"->ipywidgets>=7->arcgis) (2.1.3)

Requirement already satisfied: pickleshare in /usr/local/lib/python3.6/dist-packages (from ipython>=4.0.0; python\_version >= "3.3"->ipywidgets>=7->arcgis) (0.7.5)

Requirement already satisfied: jsonschema!=2.5.0,>=2.4 in /usr/local/lib/python3.6/dist-packages (from nbformat>=4.2.0->ipywidgets>=7->arcgis) (2.6.0)

Requirement already satisfied: jupyter-core in /usr/local/lib/python3.6/dist-packages (from nbformat>=4.2.0->ipywidgets>=7->arcgis) (4.6.3)

Requirement already satisfied: terminado>=0.8.1 in /usr/local/lib/python3.6/dist-packages (from notebook>=4.4.1->widgetsnbextension>=3->arcgis) (0.8.3)

Requirement already satisfied: Send2Trash in /usr/local/lib/python3.6/dist-packages (from notebook>=4.4.1->widgetsnbextension>=3->arcgis) (1.5.0)

Requirement already satisfied: nbconvert in /usr/local/lib/python3.6/dist-packages (from notebook>=4.4.1->widgetsnbextension>=3->arcgis) (5.6.1)

Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.6/dist-packages (from importlib-metadata; python\_version < "3.8"->keyring>=19->arcgis) (3.1.0)

Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.6/dist-packages (from jinja2>=2.10->jupyterlab->arcgis) (1.1.1)

Collecting json5

Using cached https://files.pythonhosted.org/packages/2b/81/22bf51a5bc60dde18bb6164fd597f18ee683de8670e141364d9c432dd3cf/json5-0.9.5-py2.py3-none-any.whl

Requirement already satisfied: cffi!=1.11.3,>=1.8 in /usr/local/lib/python3.6/dist-packages (from cryptography>=1.3->requests\_ntlm->arcgis) (1.14.1)

Requirement already satisfied: pyzmq>=13 in /usr/local/lib/python3.6/dist-packages (from jupyter-client->ipykernel>=4.5.1->ipywidgets>=7->arcgis) (19.0.2)

Requirement already satisfied: ptyprocess>=0.5 in /usr/local/lib/python3.6/dist-packages (from pexpect; sys\_platform != "win32"->ipython>=4.0.0; python\_version >= "3.3"->ipywidgets>=7->arcgis) (0.6.0)

Requirement already satisfied: wcwidth in /usr/local/lib/python3.6/dist-packages (from prompt-toolkit<2.0.0,>=1.0.4->ipython>=4.0.0; python\_version >= "3.3"->ipywidgets>=7->arcgis) (0.2.5)

Requirement already satisfied: entrypoints>=0.2.2 in /usr/local/lib/python3.6/dist-packages (from nbconvert->notebook>=4.4.1->widgetsnbextension>=3->arcgis) (0.3)

Requirement already satisfied: bleach in /usr/local/lib/python3.6/dist-packages (from nbconvert->notebook>=4.4.1->widgetsnbextension>=3->arcgis) (3.1.5)

Requirement already satisfied: mistune<2,>=0.8.1 in /usr/local/lib/python3.6/dist-packages (from nbconvert->notebook>=4.4.1->widgetsnbextension>=3->arcgis) (0.8.4)

Requirement already satisfied: pandocfilters>=1.4.1 in /usr/local/lib/python3.6/dist-packages (from nbconvert->notebook>=4.4.1->widgetsnbextension>=3->arcgis) (1.4.2)

Requirement already satisfied: testpath in /usr/local/lib/python3.6/dist-packages (from nbconvert->notebook>=4.4.1->widgetsnbextension>=3->arcgis) (0.4.4)

Requirement already satisfied: defusedxml in /usr/local/lib/python3.6/dist-packages (from nbconvert->notebook>=4.4.1->widgetsnbextension>=3->arcgis) (0.6.0)

Requirement already satisfied: pycparser in /usr/local/lib/python3.6/dist-packages (from cffi!=1.11.3,>=1.8->cryptography>=1.3->requests\_ntlm->arcgis) (2.20)

Requirement already satisfied: webencodings in /usr/local/lib/python3.6/dist-packages (from bleach->nbconvert->notebook>=4.4.1->widgetsnbextension>=3->arcgis) (0.5.1)

Requirement already satisfied: packaging in /usr/local/lib/python3.6/dist-packages (from bleach->nbconvert->notebook>=4.4.1->widgetsnbextension>=3->arcgis) (20.4)

Building wheels for collected packages: arcgis, lerc, pyshp

Building wheel for arcgis (setup.py) ... done

Created wheel for arcgis: filename=arcgis-1.8.2-py2.py3-none-any.whl size=2622661 sha256=32d7a98d89878851616895124a49fc49b22387150423db0507cded14b981e6f4

Stored in directory: /root/.cache/pip/wheels/6b/7e/c6/525051b3bfef47058a9d5ecf450b35fa565aff22fb145008ae

Building wheel for lerc (setup.py) ... done

Created wheel for lerc: filename=lerc-0.1.0-cp36-none-any.whl size=571699 sha256=a5ae1a9faf9b61c1ace2adcbb5f5f01d5ec5c9489bd9fc6684fecaed12abe467

Stored in directory: /root/.cache/pip/wheels/91/25/fc/0975ce9d050d66a246aa3cd51b04af4ef94a8ad96ccda87f86

Building wheel for pyshp (setup.py) ... done

Created wheel for pyshp: filename=pyshp-2.1.0-cp36-none-any.whl size=32609 sha256=3873926930f04a718ce39aa83893277627912732586ff030a1d12541c308ca5c

Stored in directory: /root/.cache/pip/wheels/a6/0c/de/321b5192ad416b328975a2f0385f72c64db4656501eba7cc1a

Successfully built arcgis lerc pyshp

ERROR: jupyterlab-server 1.2.0 has requirement jsonschema>=3.0.1, but you'll have jsonschema 2.6.0 which is incompatible.

Installing collected packages: jeepney, cryptography, SecretStorage, keyring, lerc, json5, jupyterlab-server, jupyterlab, pyshp, requests-toolbelt, ntlm-auth, requests-ntlm, arcgis

Successfully installed SecretStorage-3.1.2 arcgis-1.8.2 cryptography-3.0 jeepney-0.4.3 json5-0.9.5 jupyterlab-2.2.5 jupyterlab-server-1.2.0 keyring-21.3.0 lerc-0.1.0 ntlm-auth-1.5.0 pyshp-2.1.0 requests-ntlm-1.1.0 requests-toolbelt-0.9.1

In [15]:

from arcgis.geocoding import geocode

from arcgis.gis import GIS

gis = GIS()

Defining London arcgis geocode function to return latitude and longitude

In [16]:

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)

Checking sample data

In [17]:

c = get\_x\_y\_uk('SE2')

In [18]:

c

Out[18]:

'51.492450000000076,0.12127000000003818'

Looks good, We Copy over the postal codes of london to pass it into the geolocator function that we just defined above

In [19]:

geo\_coordinates\_uk = df1['post\_code']

geo\_coordinates\_uk

Out[19]:

0 SE2

1 W3, W4

6 EC3

7 WC2

9 SE20

...

523 IG8, E18

524 IG8

527 N12

528 SE18

530 W12

Name: post\_code, Length: 310, dtype: object

Passing postal codes of london to get the geographical co-ordinates

In [20]:

coordinates\_latlng\_uk = geo\_coordinates\_uk.apply(lambda x: get\_x\_y\_uk(x))

coordinates\_latlng\_uk

Out[20]:

0 51.492450000000076,0.12127000000003818

1 51.51324000000005,-0.2674599999999714

6 51.51200000000006,-0.08057999999994081

7 51.51651000000004,-0.11967999999995982

9 51.41009000000008,-0.05682999999993399

...

523 51.589770000000044,0.030520000000024083

524 51.50642000000005,-0.1272099999999341

527 51.615920000000074,-0.1767399999999384

528 51.48207000000008,0.07143000000002075

530 51.50645000000003,-0.2369099999999662

Name: post\_code, Length: 310, dtype: object

Latitude

Extracting the latitude from our previously collected coordinates

In [21]:

lat\_uk = coordinates\_latlng\_uk.apply(lambda x: x.split(',')[0])

lat\_uk

Out[21]:

0 51.492450000000076

1 51.51324000000005

6 51.51200000000006

7 51.51651000000004

9 51.41009000000008

...

523 51.589770000000044

524 51.50642000000005

527 51.615920000000074

528 51.48207000000008

530 51.50645000000003

Name: post\_code, Length: 310, dtype: object

Longitude

Extracting the Longitude from our previously collected coordinates

In [22]:

lng\_uk = coordinates\_latlng\_uk.apply(lambda x: x.split(',')[1])

lng\_uk

Out[22]:

0 0.12127000000003818

1 -0.2674599999999714

6 -0.08057999999994081

7 -0.11967999999995982

9 -0.05682999999993399

...

523 0.030520000000024083

524 -0.1272099999999341

527 -0.1767399999999384

528 0.07143000000002075

530 -0.2369099999999662

Name: post\_code, Length: 310, dtype: object

We now have the geographical co-ordinates of the London Neighbourhoods.

We proceed with Merging our source data with the geographical co-ordinates to make our dataset ready for the next stage

In [23]:

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

Out[23]:

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

In [24]:

london\_merged.dtypes

Out[24]:

borough object

town object

post\_code object

latitude float64

longitude float64

dtype: object

Co-ordinates for London

Getting the geocode for London to help visualize it on the map

In [25]:

london = geocode(address='London, England, GBR')[0]

london\_lng\_coords = london['location']['x']

london\_lat\_coords = london['location']['y']

london\_lng\_coords

Out[25]:

-0.1272099999999341

In [26]:

london\_lat\_coords

Out[26]:

51.50642000000005

Visualize the Map of London

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

In [27]:

# 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

Out[27]:

Make this Notebook Trusted to load map: File -> Trust Notebook

Venues in London

To proceed with the next part, we need to define Foursquare API credentials.

Using Foursquare API, we are able to get the venue and venue categories around each neighbourhood in London.

In [32]:

CLIENT\_ID = 'LDIJF4KI5VGMMA3NNDLFZWHR12TCMNTUL0TUC3QPZ3SJD040'

CLIENT\_SECRET = '0DXHVDFCZXNXFSLOFGOONJSS35KH4NAZXZN2AAAX5GCZVVTH'

VERSION = '20180605' # Foursquare API version

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

In [57]:

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,

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

In [58]:

venues\_in\_London = getNearbyVenues(london\_merged['borough'], london\_merged['latitude'], london\_merged['longitude'])

Bexley, Greenwich

Ealing, Hammersmith and Fulham

City

Westminster

Bromley

Islington

Islington

Barnet

Enfield

Wandsworth

Southwark

City

Richmond upon Thames

Barnet

Islington

Wandsworth

Westminster

Bromley

Newham

Ealing

Westminster

Lewisham

Camden

Southwark

Tower Hamlets

Bexley

City

Lewisham

Greenwich

Tower Hamlets

Camden

Haringey

Tower Hamlets

Haringey

Barnet

Brent

Lambeth

Lewisham

Tower Hamlets

Kensington and ChelseaHammersmith and Fulham

Brent

Barnet

Barnet

Southwark

Tower Hamlets

Camden

Tower Hamlets

Waltham Forest

Newham

Islington

Richmond upon Thames

Lewisham

Camden

Westminster

Greenwich

Kensington and Chelsea

Barnet

Westminster

Lewisham

Waltham Forest

Hounslow, Ealing, Hammersmith and Fulham

Brent

Barnet

Lambeth, Wandsworth

Islington

Barnet

Merton

Barnet

Westminster

Barnet, Brent, Camden

Lewisham

Bexley

Haringey

Bromley

Tower Hamlets

Newham

Hackney

Dartford

Islington

Southwark

Lewisham

Brent

Southwark

Ealing

Kensington and Chelsea

Wandsworth

Southwark

Barnet

Newham

Richmond upon Thames

Enfield

Southwark

Greenwich

Bexley, Greenwich

Islington & City

Barnet

Islington

Haringey, Islington

Camden

Newham

Lewisham

Haringey

Barnet

Camden

Hammersmith and Fulham

Lambeth

Barnet

Camden

Barnet

Enfield

Greenwich

Hounslow

Lewisham

Hounslow

Hackney

Hackney

Hackney

Hackney

Hackney

Barnet

Hammersmith and Fulham

Camden

Barnet

Ealing

Brent

Haringey

Barnet

Lambeth

Waltham Forest

Islington

Camden

Lewisham

Camden

Kensington and Chelsea

Islington

Hackney

Lewisham

Greenwich, Lewisham

Haringey

Hackney

Barnet

Tower Hamlets

Islington

Lambeth, Southwark

Brent

Kensington and Chelsea

Camden

Brent, Harrow

Greenwich

Brent, Camden

Camden and Islington

Brent

Kingston upon Thames

Westminster

Lewisham

Lambeth

Hackney

Tower Hamlets

Lewisham

Lewisham

Waltham Forest

Waltham Forest

Tower Hamlets

Westminster

Newham

Westminster

Bexley

Hackney

Westminster

Hackney

Newham

Newham

Westminster

Westminster

Greenwich

Merton

Greenwich

Tower Hamlets

Barnet

Westminster

Tower Hamlets

Richmond upon Thames

Bromley

Haringey and Barnet

Islington

Brent

Lewisham

Greenwich

Barnet

Southwark

Wandsworth

Croydon

Barnet

Kensington and Chelsea

Newham

Kensington and Chelsea

Southwark

Barnet

Tower Hamlets

Hammersmith and Fulham

Barnet

Lambeth

Westminster

Enfield

Brent, Ealing

Hammersmith and Fulham

Southwark

Bromley

Islington

Westminster

Newham

Greenwich

Tower Hamlets

Camden

Wandsworth

Brent

Harrow, Brent

Tower Hamlets

Merton

Wandsworth

Southwark

Hammersmith and Fulham

Croydon

Haringey

Hackney

Tower Hamlets

Hammersmith and Fulham

Greenwich

Hackney

Newham

Redbridge, Waltham Forest

Westminster

Camden

Hackney

Kensington and Chelsea

Croydon

Merton

Redbridge

Haringey

Lewisham

Wandsworth

Enfield

Tower Hamlets

Westminster

Camden

Lewisham

Westminster

Islington

Camden

Hackney

Tower Hamlets

Lambeth

Hackney

Brent

Newham

Lambeth

Haringey

Southwark

Camden

Lewisham, Bromley

Lewisham, Southwark

City, Westminster

Bexley, Greenwich

Wandsworth

Wandsworth

Haringey

Haringey

Haringey

Barnet

Tower Hamlets

Islington, Camden

Lambeth

Haringey

Hackney

Islington

Croydon

Waltham Forest

Newham

Lambeth

Waltham Forest

Waltham Forest

Southwark

Wandsworth

Redbridge

Tower Hamlets

Greenwich

Kensington and Chelsea

Ealing

Haringey

Hackney

Newham

Camden

Bexley

Barnet

Hammersmith and Fulham

Lambeth

Greenwich

Westminster

Barnet

Hammersmith and Fulham

Tower Hamlets

Brent

Merton

Enfield

Haringey

Redbridge

Redbridge, Waltham Forest

Barnet

Greenwich

Hammersmith and Fulham

Sampling our data

In [59]:

venues\_in\_London.head()

Out[59]:

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

In [60]:

venues\_in\_London.shape

Out[60]:

(10567, 5)

Wow, we have scraped together 10567 records for venues. This will definitely make the clustering interesting.

Grouping by Venue Categories

We need to now see how many Venue Categories are there for further processing

In [62]:

venues\_in\_London.groupby('Venue Category').max()

Out[62]:

Neighbourhood Neighbourhood Latitude Neighbourhood Longitude Venue

Venue Category

Accessories Store Westminster 51.51651 -0.11968 James Smith & Sons

Adult Boutique Islington 51.52969 -0.08697 Sh! Women's Erotic Emporium

African Restaurant Westminster 51.52587 -0.08808 Red Sea Restaurant

American Restaurant Waltham Forest 51.61780 0.02912 The Fat Bear

Antique Shop Westminster 51.55506 -0.11968 The London Silver Vaults

... ... ... ... ...

Wings Joint Hammersmith and Fulham 51.54187 -0.12273 Wingmans

Women's Store Kensington and ChelseaHammersmith and Fulham 51.55457 -0.11478 Vivien of Holloway

Xinjiang Restaurant Southwark 51.47480 -0.09313 Silk Road

Yoga Studio Westminster 51.55457 -0.03558 yogahaven

Zoo Exhibit Camden 51.53354 -0.14606 Penguin Beach

295 rows × 4 columns

We can see 295 records, just goes to show how diverse and interesting the place is.

One Hot Encoding

We need to Encode our venue categories to get a better result for our clustering

In [63]:

London\_venue\_cat = pd.get\_dummies(venues\_in\_London[['Venue Category']], prefix="", prefix\_sep="")

London\_venue\_cat

Out[63]:

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 Australian Restaurant Auto Garage Auto Workshop BBQ Joint Bagel Shop Bakery Bar Beach Bed & Breakfast Beer Bar Beer Garden Beer Store Bike Shop Bistro Boarding House Bookstore Boutique Boxing Gym Brasserie Brazilian Restaurant Breakfast Spot Brewery Bridal Shop Building Burger Joint Burrito Place Bus Station ... Sporting Goods Shop Sports Bar Sports Club Sri Lankan Restaurant Stables Stationery Store Steakhouse Street Food Gathering Student Center Supermarket Sushi Restaurant Szechuan Restaurant Taco Place Tailor Shop Tapas Restaurant Tea Room Tennis Court Thai Restaurant Theater Thrift / Vintage Store Tour Provider Tourist Information Center Toy / Game Store Trail Train Station Tunnel Turkish Restaurant University Vape Store Vegetarian / Vegan Restaurant Video Game Store Vietnamese Restaurant Warehouse Store Wine Bar Wine Shop Wings Joint Women's Store Xinjiang Restaurant Yoga Studio Zoo Exhibit

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ... 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ... 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ... 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ... 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ... 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ...

10562 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ... 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

10563 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ... 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

10564 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ... 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

10565 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ... 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

10566 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ... 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

10567 rows × 295 columns

Adding Neighbourhood into the mix.

In [64]:

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]

London\_venue\_cat.head()

Out[64]:

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 Australian Restaurant Auto Garage Auto Workshop BBQ Joint Bagel Shop Bakery Bar Beach Bed & Breakfast Beer Bar Beer Garden Beer Store Bike Shop Bistro Boarding House Bookstore Boutique Boxing Gym Brasserie Brazilian Restaurant Breakfast Spot Brewery Bridal Shop Building Burger Joint Burrito Place ... Sporting Goods Shop Sports Bar Sports Club Sri Lankan Restaurant Stables Stationery Store Steakhouse Street Food Gathering Student Center Supermarket Sushi Restaurant Szechuan Restaurant Taco Place Tailor Shop Tapas Restaurant Tea Room Tennis Court Thai Restaurant Theater Thrift / Vintage Store Tour Provider Tourist Information Center Toy / Game Store Trail Train Station Tunnel Turkish Restaurant University Vape Store Vegetarian / Vegan Restaurant Video Game Store Vietnamese Restaurant Warehouse Store Wine Bar Wine Shop Wings Joint Women's Store Xinjiang Restaurant Yoga Studio Zoo Exhibit

0 Bexley, Greenwich 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ... 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 Bexley, Greenwich 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ... 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

2 Bexley, Greenwich 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ... 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

3 Bexley, Greenwich 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ... 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

4 Bexley, Greenwich 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ... 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

5 rows × 296 columns

Venue categories mean value

We will group the Neighbourhoods and calculate the mean venue categories value in each Neighbourhood

In [65]:

London\_grouped = London\_venue\_cat.groupby('Neighbourhood').mean().reset\_index()

London\_grouped.head()

Out[65]:

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 Australian Restaurant Auto Garage Auto Workshop BBQ Joint Bagel Shop Bakery Bar Beach Bed & Breakfast Beer Bar Beer Garden Beer Store Bike Shop Bistro Boarding House Bookstore Boutique Boxing Gym Brasserie Brazilian Restaurant Breakfast Spot Brewery Bridal Shop Building Burger Joint Burrito Place ... Sporting Goods Shop Sports Bar Sports Club Sri Lankan Restaurant Stables Stationery Store Steakhouse Street Food Gathering Student Center Supermarket Sushi Restaurant Szechuan Restaurant Taco Place Tailor Shop Tapas Restaurant Tea Room Tennis Court Thai Restaurant Theater Thrift / Vintage Store Tour Provider Tourist Information Center Toy / Game Store Trail Train Station Tunnel Turkish Restaurant University Vape Store Vegetarian / Vegan Restaurant Video Game Store Vietnamese Restaurant Warehouse Store Wine Bar Wine Shop Wings Joint Women's Store Xinjiang Restaurant Yoga Studio Zoo Exhibit

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 0.0 0.0 0.007547 0.001887 0.013208 0.020755 0.00566 0.0 0.0 0.00566 0.0 0.0 0.0 0.0 0.0 0.009434 0.0 0.0 0.0 0.0 0.00566 0.0 0.0 0.0 0.0 0.0 ... 0.0 0.0 0.000 0.0 0.0 0.009434 0.001887 0.0 0.00566 0.035849 0.018868 0.0 0.0 0.0 0.001887 0.00566 0.001887 0.011321 0.00566 0.0 0.0 0.0 0.001887 0.0 0.009434 0.0 0.028302 0.0 0.0 0.0 0.0 0.001887 0.0 0.0 0.0 0.0 0.0 0.0 0.0 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 0.0 0.0 0.000000 0.000000 0.000000 0.000000 0.00000 0.0 0.0 0.00000 0.0 0.0 0.0 0.0 0.0 0.000000 0.0 0.0 0.0 0.0 0.00000 0.0 0.0 0.0 0.0 0.0 ... 0.0 0.0 0.000 0.0 0.0 0.000000 0.000000 0.0 0.00000 0.250000 0.000000 0.0 0.0 0.0 0.000000 0.00000 0.000000 0.000000 0.00000 0.0 0.0 0.0 0.000000 0.0 0.000000 0.0 0.000000 0.0 0.0 0.0 0.0 0.000000 0.0 0.0 0.0 0.0 0.0 0.0 0.0 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 0.0 0.0 0.000000 0.000000 0.000000 0.000000 0.00000 0.0 0.0 0.00000 0.0 0.0 0.0 0.0 0.0 0.000000 0.0 0.0 0.0 0.0 0.00000 0.0 0.0 0.0 0.0 0.0 ... 0.0 0.0 0.000 0.0 0.0 0.000000 0.000000 0.0 0.00000 0.230769 0.000000 0.0 0.0 0.0 0.000000 0.00000 0.000000 0.000000 0.00000 0.0 0.0 0.0 0.000000 0.0 0.115385 0.0 0.000000 0.0 0.0 0.0 0.0 0.000000 0.0 0.0 0.0 0.0 0.0 0.0 0.0 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 0.0 0.0 0.000000 0.000000 0.000000 0.000000 0.00000 0.0 0.0 0.00000 0.0 0.0 0.0 0.0 0.0 0.000000 0.0 0.0 0.0 0.0 0.00000 0.0 0.0 0.0 0.0 0.0 ... 0.0 0.0 0.125 0.0 0.0 0.000000 0.000000 0.0 0.00000 0.000000 0.000000 0.0 0.0 0.0 0.000000 0.00000 0.000000 0.000000 0.00000 0.0 0.0 0.0 0.000000 0.0 0.000000 0.0 0.000000 0.0 0.0 0.0 0.0 0.000000 0.0 0.0 0.0 0.0 0.0 0.0 0.0 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 0.0 0.0 0.000000 0.000000 0.000000 0.000000 0.00000 0.0 0.0 0.00000 0.0 0.0 0.0 0.0 0.0 0.000000 0.0 0.0 0.0 0.0 0.00000 0.0 0.0 0.0 0.0 0.0 ... 0.0 0.0 0.000 0.0 0.0 0.000000 0.000000 0.0 0.00000 0.285714 0.000000 0.0 0.0 0.0 0.000000 0.00000 0.000000 0.000000 0.00000 0.0 0.0 0.0 0.000000 0.0 0.142857 0.0 0.000000 0.0 0.0 0.0 0.0 0.000000 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0

5 rows × 296 columns

Let's make a function to get the top most common venue categories

In [66]:

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 way too many venue categories, we can take the top 10 to cluster the neighbourhoods.

Creating a function to label the columns of the venue correctly

In [67]:

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))

Top venue categories

Getting the top venue categories in London

In [74]:

# create a new dataframe for London

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\_venues)

neighborhoods\_venues\_sorted\_london.head()

Out[74]:

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

Model Building

K Means

Let's cluster the city of london to roughly 5 to make it easier to analyze.

We use the K Means clustering technique to do so.

In [75]:

# 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)

kmeans\_london

Out[75]:

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 Clustered Data

In [76]:

kmeans\_london.labels\_

Out[76]:

array([1, 4, 3, 1, ..., 1, 1, 1, 1], dtype=int32)

So our model has labeled the city

In [77]:

neighborhoods\_venues\_sorted\_london.insert(0, 'Cluster Labels', kmeans\_london.labels\_ +1)

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

In [78]:

london\_data = london\_merged

london\_data = london\_data.join(neighborhoods\_venues\_sorted\_london.set\_index('Neighbourhood'), on='borough')

london\_data.head()

Out[78]:

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 9th Most Common Venue 10th Most Common Venue

0 Bexley, Greenwich LONDON SE2 51.49245 0.12127 4 Supermarket Platform Convenience Store Historic Site Train Station Coffee Shop Zoo Exhibit Film Studio Event Space Exhibit

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 Exhibit Falafel Restaurant Farmers Market

6 City LONDON EC3 51.51200 -0.08058 2 Coffee Shop Italian Restaurant Hotel Pub Gym / Fitness Center Food Truck Sandwich Place Beer Bar Wine Bar Cocktail Bar

7 Westminster LONDON WC2 51.51651 -0.11968 2 Hotel Coffee Shop Pub Sandwich Place Café Italian Restaurant Restaurant Theater French Restaurant Bakery

9 Bromley LONDON SE20 51.41009 -0.05683 2 Supermarket Grocery Store Convenience Store Hotel Fast Food Restaurant Park Italian Restaurant Gym / Fitness Center Historic Site Golf Course

Drop all the NaN values to prevent data skew

In [79]:

london\_data\_nonan = london\_data.dropna(subset=['Cluster Labels'])

Visualizing the clustered neighbourhood

Let's plot the clusters

In [80]:

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

Out[80]:

Make this Notebook Trusted to load map: File -> Trust Notebook

Examining our Clusters

Cluster 1

In [81]:

london\_data\_nonan.loc[london\_data\_nonan['Cluster Labels'] == 1, london\_data\_nonan.columns[[1] + list(range(5, london\_data\_nonan.shape[1]))]]

Out[81]:

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 1 Grocery Store Train Station Breakfast Spot Park Indian Restaurant Deli / Bodega Fish Market Exhibit Falafel Restaurant Farmers Market

249 LONDON 1 Fried Chicken Joint Pub Grocery Store Italian Restaurant Gym / Fitness Center Train Station Fish & Chips Shop Farmers Market Ethiopian Restaurant Event Space

470 LONDON 1 Grocery Store Pizza Place Coffee Shop Pub Italian Restaurant Kebab Restaurant Park Sandwich Place Farmers Market Fast Food Restaurant

Cluster 2

In [82]:

london\_data\_nonan.loc[london\_data\_nonan['Cluster Labels'] == 2, london\_data\_nonan.columns[[1] + list(range(5, london\_data\_nonan.shape[1]))]]

Out[82]:

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 2 Coffee Shop Italian Restaurant Hotel Pub Gym / Fitness Center Food Truck Sandwich Place Beer Bar Wine Bar Cocktail Bar

7 LONDON 2 Hotel Coffee Shop Pub Sandwich Place Café Italian Restaurant Restaurant Theater French Restaurant Bakery

9 LONDON 2 Supermarket Grocery Store Convenience Store Hotel Fast Food Restaurant Park Italian Restaurant Gym / Fitness Center Historic Site Golf Course

10 LONDON 2 Pub Coffee Shop Café Food Truck Vietnamese Restaurant Hotel Gym / Fitness Center Park Cocktail Bar Breakfast Spot

12 LONDON 2 Pub Coffee Shop Café Food Truck Vietnamese Restaurant Hotel Gym / Fitness Center Park Cocktail Bar Breakfast Spot

... ... ... ... ... ... ... ... ... ... ... ... ...

523 LONDON 2 Café Pub Grocery Store Coffee Shop BBQ Joint Bridal Shop Seafood Restaurant Park Bar Bakery

524 LONDON, WOODFORD GREEN 2 Hotel Café Garden Outdoor Sculpture Monument / Landmark Theater Plaza Pub Bakery Coffee Shop

527 LONDON 2 Coffee Shop Café Grocery Store Pub Italian Restaurant Supermarket Pharmacy Chinese Restaurant Turkish Restaurant Pizza Place

528 LONDON 2 Pub Grocery Store Bus Stop Indian Restaurant Coffee Shop Golf Course Turkish Restaurant Supermarket Fish & Chips Shop Construction & Landscaping

530 LONDON 2 Pub Coffee Shop Café Grocery Store Gastropub Pizza Place Bakery Thai Restaurant Italian Restaurant Park

300 rows × 12 columns

Cluster 3

In [83]:

london\_data\_nonan.loc[london\_data\_nonan['Cluster Labels'] == 3, london\_data\_nonan.columns[[1] + list(range(5, london\_data\_nonan.shape[1]))]]

Out[83]:

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 3 Bakery Construction & Landscaping Gym Metro Station Food Service Food Court Food Stand Food & Drink Shop Flower Shop Flea Market

Cluster 4

In [84]:

london\_data\_nonan.loc[london\_data\_nonan['Cluster Labels'] == 4, london\_data\_nonan.columns[[1] + list(range(5, london\_data\_nonan.shape[1]))]]

Out[84]:

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 4 Supermarket Platform Convenience Store Historic Site Train Station Coffee Shop Zoo Exhibit Film Studio Event Space Exhibit

45 BEXLEYHEATH, LONDON 4 Supermarket Historic Site Train Station Platform Convenience Store Coffee Shop Bus Stop Golf Course Construction & Landscaping Park

124 LONDON 4 Supermarket Historic Site Train Station Platform Convenience Store Coffee Shop Bus Stop Golf Course Construction & Landscaping Park

292 LONDON, SIDCUP 4 Supermarket Historic Site Train Station Platform Convenience Store Coffee Shop Bus Stop Golf Course Construction & Landscaping Park

507 LONDON 4 Supermarket Historic Site Train Station Platform Convenience Store Coffee Shop Bus Stop Golf Course Construction & Landscaping Park

Cluster 5

In [85]:

london\_data\_nonan.loc[london\_data\_nonan['Cluster Labels'] == 5, london\_data\_nonan.columns[[1] + list(range(5, london\_data\_nonan.shape[1]))]]

Out[85]:

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

121 LONDON 5 Gym / Fitness Center Music Venue Clothing Store Supermarket Zoo Exhibit Film Studio Event Space Exhibit Falafel Restaurant Farmers Market

Exploring Paris

Neighbourhoods of Paris

Data Collection

We read the json data with pandas.

In [86]:

!wget -q -O 'france-data.json' https://www.data.gouv.fr/fr/datasets/r/e88c6fda-1d09-42a0-a069-606d3259114e

print("Data Downloaded!")

Data Downloaded!

In [36]:

paris\_raw = pd.read\_json('france-data.json')

paris\_raw.head()

Out[36]:

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

Data Preprocessing

We break down each of the nested fields and create the dataframe that we need

In [37]:

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()

Out[37]:

code\_arr code\_cant code\_comm code\_dept code\_reg geo\_point\_2d geo\_shape id\_geofla insee\_com nom\_comm nom\_dept nom\_region population postal\_code statut superficie z\_moyen

0 3 34 003 94 11 [48.80588035965699, 2.333510249842654] {'type': 'Polygon', 'coordinates': [[[2.343851... 32123 94003 ARCUEIL VAL-DE-MARNE ILE-DE-FRANCE 19.5 94110 Chef-lieu canton 232.0 70.0

1 1 09 430 77 11 [49.07375705850074, 2.787942211427862] {'type': 'Polygon', 'coordinates': [[[2.779756... 33986 77430 SAINT-PATHUS SEINE-ET-MARNE ILE-DE-FRANCE 5.5 77178 Commune simple 535.0 101.0

2 2 32 412 77 11 [48.47430724856196, 2.510781898359126] {'type': 'Polygon', 'coordinates': [[[2.505938... 18225 77412 SAINT-GERMAIN-SUR-ECOLE SEINE-ET-MARNE ILE-DE-FRANCE 0.4 77930 Commune simple 252.0 62.0

3 2 24 598 95 11 [48.98840778751709, 2.300499783419528] {'type': 'Polygon', 'coordinates': [[[2.297857... 32145 95598 SOISY-SOUS-MONTMORENCY VAL-D'OISE ILE-DE-FRANCE 17.2 95230 Chef-lieu canton 394.0 59.0

4 2 32 040 77 11 [48.51197791258124, 2.569919095331142] {'type': 'Polygon', 'coordinates': [[[2.558809... 28430 77040 BOISSISE-LE-ROI SEINE-ET-MARNE ILE-DE-FRANCE 3.6 77310 Commune simple 711.0 69.0

Feature Selection

We take the columns that we require, in case of paris it would be the geo\_point\_2d, nom\_dept, nom\_comm and postal\_code

In [38]:

df\_2 = paris\_field\_data[['postal\_code','nom\_comm','nom\_dept','geo\_point\_2d']]

df\_2

Out[38]:

postal\_code nom\_comm nom\_dept geo\_point\_2d

0 94110 ARCUEIL VAL-DE-MARNE [48.80588035965699, 2.333510249842654]

1 77178 SAINT-PATHUS SEINE-ET-MARNE [49.07375705850074, 2.787942211427862]

2 77930 SAINT-GERMAIN-SUR-ECOLE SEINE-ET-MARNE [48.47430724856196, 2.510781898359126]

3 95230 SOISY-SOUS-MONTMORENCY VAL-D'OISE [48.98840778751709, 2.300499783419528]

4 77310 BOISSISE-LE-ROI SEINE-ET-MARNE [48.51197791258124, 2.569919095331142]

... ... ... ... ...

1295 77230 VINANTES SEINE-ET-MARNE [49.00336689019137, 2.739932368905603]

1296 91350 GRIGNY ESSONNE [48.6568896115566, 2.387896478580817]

1297 77440 JAIGNES SEINE-ET-MARNE [48.98705524224735, 3.074558248708759]

1298 77220 GRETZ-ARMAINVILLIERS SEINE-ET-MARNE [48.74953288926905, 2.7262357852056303]

1299 77580 VOULANGIS SEINE-ET-MARNE [48.84031218010945, 2.889716285735505]

1300 rows × 4 columns

We have managed to collect 1300 records.

Feature Engineering

We localize to only include Paris

In [39]:

df\_paris = df\_2[df\_2['nom\_dept'].str.contains('PARIS')].reset\_index(drop=True)

df\_paris

Out[39]:

postal\_code nom\_comm nom\_dept geo\_point\_2d

0 75010 PARIS-10E-ARRONDISSEMENT PARIS [48.87602855694339, 2.361112904561707]

1 75016 PARIS-16E-ARRONDISSEMENT PARIS [48.86039876035177, 2.262099559395783]

2 75009 PARIS-9E-ARRONDISSEMENT PARIS [48.87689616237872, 2.337460241388529]

3 75015 PARIS-15E-ARRONDISSEMENT PARIS [48.84015541860987, 2.293559372435076]

4 75002 PARIS-2E-ARRONDISSEMENT PARIS [48.86790337886785, 2.344107166658533]

5 75011 PARIS-11E-ARRONDISSEMENT PARIS [48.85941549762748, 2.378741060237548]

6 75005 PARIS-5E-ARRONDISSEMENT PARIS [48.844508659617546, 2.349859385560182]

7 75019 PARIS-19E-ARRONDISSEMENT PARIS [48.88686862295828, 2.384694327870042]

8 75020 PARIS-20E-ARRONDISSEMENT PARIS [48.86318677744551, 2.400819826729021]

9 75003 PARIS-3E-ARRONDISSEMENT PARIS [48.86305413181178, 2.359361058970589]

10 75006 PARIS-6E-ARRONDISSEMENT PARIS [48.84896809191946, 2.332670898588416]

11 75018 PARIS-18E-ARRONDISSEMENT PARIS [48.892735074561706, 2.348711933867703]

12 75008 PARIS-8E-ARRONDISSEMENT PARIS [48.87252726662346, 2.312582560420059]

13 75013 PARIS-13E-ARRONDISSEMENT PARIS [48.82871768452136, 2.362468228516128]

14 75012 PARIS-12E-ARRONDISSEMENT PARIS [48.83515623066034, 2.419807034965275]

15 75007 PARIS-7E-ARRONDISSEMENT PARIS [48.85608259819694, 2.312438687733857]

16 75001 PARIS-1ER-ARRONDISSEMENT PARIS [48.8626304851685, 2.336293446550539]

17 75004 PARIS-4E-ARRONDISSEMENT PARIS [48.854228281954754, 2.357361938142205]

18 75017 PARIS-17E-ARRONDISSEMENT PARIS [48.88733716648682, 2.307485559493426]

19 75014 PARIS-14E-ARRONDISSEMENT PARIS [48.82899321160942, 2.327100883257538]

In [40]:

df\_paris.shape

Out[40]:

(20, 4)

We have managed to bring down the records to just 20 from 1300. Paris data is quite small than initally thought

In [41]:

df\_paris.dtypes

Out[41]:

postal\_code object

nom\_comm object

nom\_dept object

geo\_point\_2d object

dtype: object

Gelocations of the Neighbourhoods of Paris

We don't need to get the geo coordinates using an external data source or collect it with the arcgis API call since we already have it stored in the geo\_point\_2d column as a tuple in the df\_paris dataframe.

Checking one of the geo coordinates.

In [87]:

df\_paris['geo\_point\_2d'][0]

Out[87]:

[48.87602855694339, 2.361112904561707]

In [43]:

temp1 = df\_paris['geo\_point\_2d']

temp1

Out[43]:

0 [48.87602855694339, 2.361112904561707]

1 [48.86039876035177, 2.262099559395783]

2 [48.87689616237872, 2.337460241388529]

3 [48.84015541860987, 2.293559372435076]

4 [48.86790337886785, 2.344107166658533]

5 [48.85941549762748, 2.378741060237548]

6 [48.844508659617546, 2.349859385560182]

7 [48.88686862295828, 2.384694327870042]

8 [48.86318677744551, 2.400819826729021]

9 [48.86305413181178, 2.359361058970589]

10 [48.84896809191946, 2.332670898588416]

11 [48.892735074561706, 2.348711933867703]

12 [48.87252726662346, 2.312582560420059]

13 [48.82871768452136, 2.362468228516128]

14 [48.83515623066034, 2.419807034965275]

15 [48.85608259819694, 2.312438687733857]

16 [48.8626304851685, 2.336293446550539]

17 [48.854228281954754, 2.357361938142205]

18 [48.88733716648682, 2.307485559493426]

19 [48.82899321160942, 2.327100883257538]

Name: geo\_point\_2d, dtype: object

In [45]:

paris\_latlng = df\_paris['geo\_point\_2d'].astype('str')

Spliting the geo\_point\_2d column into latitude and longitude.

Latitude

In [46]:

paris\_lat = paris\_latlng.apply(lambda x: x.split(',')[0])

paris\_lat = paris\_lat.apply(lambda x: x.lstrip('['))

paris\_lat

Out[46]:

0 48.87602855694339

1 48.86039876035177

2 48.87689616237872

3 48.84015541860987

4 48.86790337886785

5 48.85941549762748

6 48.844508659617546

7 48.88686862295828

8 48.86318677744551

9 48.86305413181178

10 48.84896809191946

11 48.892735074561706

12 48.87252726662346

13 48.82871768452136

14 48.83515623066034

15 48.85608259819694

16 48.8626304851685

17 48.854228281954754

18 48.88733716648682

19 48.82899321160942

Name: geo\_point\_2d, dtype: object

Longitude

In [47]:

paris\_lng = paris\_latlng.apply(lambda x: x.split(',')[1])

paris\_lng = paris\_lng.apply(lambda x: x.rstrip(']'))

paris\_lng

Out[47]:

0 2.361112904561707

1 2.262099559395783

2 2.337460241388529

3 2.293559372435076

4 2.344107166658533

5 2.378741060237548

6 2.349859385560182

7 2.384694327870042

8 2.400819826729021

9 2.359361058970589

10 2.332670898588416

11 2.348711933867703

12 2.312582560420059

13 2.362468228516128

14 2.419807034965275

15 2.312438687733857

16 2.336293446550539

17 2.357361938142205

18 2.307485559493426

19 2.327100883257538

Name: geo\_point\_2d, dtype: object

In [48]:

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']

Preparing our combined data by dropping the geo\_point\_2d column from our previously stored df\_paris and concatenating with the latitude and longitude extracted from it

In [49]:

paris\_combined\_data = pd.concat([df\_paris.drop('geo\_point\_2d', axis=1), paris\_geo\_lat, paris\_geo\_lng], axis=1)

paris\_combined\_data

Out[49]:

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

16 75001 PARIS-1ER-ARRONDISSEMENT PARIS 48.862630 2.336293

17 75004 PARIS-4E-ARRONDISSEMENT PARIS 48.854228 2.357362

18 75017 PARIS-17E-ARRONDISSEMENT PARIS 48.887337 2.307486

19 75014 PARIS-14E-ARRONDISSEMENT PARIS 48.828993 2.327101

Co-ordinates for Paris

In [50]:

paris = geocode(address='Paris, France, FR')[0]

paris\_lng\_coords = paris['location']['x']

paris\_lat\_coords = paris['location']['y']

print("The geolocation of Paris: ", paris\_lat\_coords, paris\_lng\_coords)

The geolocation of Paris: 48.85341000000005 2.3488000000000397

Visualize the Map of Paris

In [52]:

# Creating the map of Paris

map\_Paris= folium.Map(location=[paris\_lat\_coords, paris\_lng\_coords], zoom\_start=12)

map\_Paris

# adding markers to map

for latitude, longitude, borough, town in zip(paris\_combined\_data['Latitude'], paris\_combined\_data['Longitude'], paris\_combined\_data['nom\_comm'], paris\_combined\_data['nom\_dept']):

label = '{}, {}'.format(town, borough)

label = folium.Popup(label, parse\_html=True)

folium.CircleMarker(

[latitude, longitude],

radius=5,

popup=label,

color='Blue',

fill=True,

fill\_opacity=0.8

).add\_to(map\_Paris)

map\_Paris

Out[52]:

Make this Notebook Trusted to load map: File -> Trust Notebook

Venues in Paris

Using our previously defined function. Let's get the neaby venues present in each neighbourhood of Paris

In [88]:

venues\_in\_Paris = getNearbyVenues(paris\_combined\_data['nom\_comm'], paris\_combined\_data['Latitude'], paris\_combined\_data['Longitude'])

PARIS-10E-ARRONDISSEMENT

PARIS-16E-ARRONDISSEMENT

PARIS-9E-ARRONDISSEMENT

PARIS-15E-ARRONDISSEMENT

PARIS-2E-ARRONDISSEMENT

PARIS-11E-ARRONDISSEMENT

PARIS-5E-ARRONDISSEMENT

PARIS-19E-ARRONDISSEMENT

PARIS-20E-ARRONDISSEMENT

PARIS-3E-ARRONDISSEMENT

PARIS-6E-ARRONDISSEMENT

PARIS-18E-ARRONDISSEMENT

PARIS-8E-ARRONDISSEMENT

PARIS-13E-ARRONDISSEMENT

PARIS-12E-ARRONDISSEMENT

PARIS-7E-ARRONDISSEMENT

PARIS-1ER-ARRONDISSEMENT

PARIS-4E-ARRONDISSEMENT

PARIS-17E-ARRONDISSEMENT

PARIS-14E-ARRONDISSEMENT

Sampling our Data

In [89]:

venues\_in\_Paris.head()

Out[89]:

Neighbourhood Neighbourhood Latitude Neighbourhood Longitude Venue Venue Category

0 PARIS-10E-ARRONDISSEMENT 48.876029 2.361113 Les Orientalistes Mediterranean Restaurant

1 PARIS-10E-ARRONDISSEMENT 48.876029 2.361113 Les Enfants Perdus French Restaurant

2 PARIS-10E-ARRONDISSEMENT 48.876029 2.361113 Marrow Café

3 PARIS-10E-ARRONDISSEMENT 48.876029 2.361113 Café A Café

4 PARIS-10E-ARRONDISSEMENT 48.876029 2.361113 Marks & Spencer Food Food & Drink Shop

In [90]:

venues\_in\_Paris.shape

Out[90]:

(1324, 5)

We have managed to collect 1324 venue records for the neighbourhoods in Paris

Grouping by Venue Categories

We need to now see how many Venue Categories are there for further processing

In [91]:

venues\_in\_Paris.groupby('Venue Category').max()

Out[91]:

Neighbourhood Neighbourhood Latitude Neighbourhood Longitude Venue

Venue Category

Afghan Restaurant PARIS-11E-ARRONDISSEMENT 48.859415 2.378741 Afghanistan

African Restaurant PARIS-9E-ARRONDISSEMENT 48.876896 2.361113 Wally Le Saharien

American Restaurant PARIS-19E-ARRONDISSEMENT 48.892735 2.384694 Harper's

Antique Shop PARIS-9E-ARRONDISSEMENT 48.876896 2.337460 Hôtel des Ventes Drouot

Argentinian Restaurant PARIS-3E-ARRONDISSEMENT 48.863054 2.359361 Anahi

... ... ... ... ...

Wine Bar PARIS-9E-ARRONDISSEMENT 48.892735 2.400820 Vingt Vins d'Art

Wine Shop PARIS-3E-ARRONDISSEMENT 48.892735 2.361113 Trois Fois Vin

Women's Store PARIS-2E-ARRONDISSEMENT 48.867903 2.344107 & Other Stories

Zoo PARIS-12E-ARRONDISSEMENT 48.835156 2.419807 Parc zoologique de Paris

Zoo Exhibit PARIS-12E-ARRONDISSEMENT 48.835156 2.419807 Grande Serre du Parc Zoologique de Paris

198 rows × 4 columns

198 records in Paris. Paris is also multicultural and diverse as London

One Hot Encoding

We need to Encode our venue categories to get a better result for our clustering

In [92]:

Paris\_venue\_cat = pd.get\_dummies(venues\_in\_Paris[['Venue Category']], prefix="", prefix\_sep="")

Paris\_venue\_cat

Out[92]:

Afghan Restaurant African Restaurant American Restaurant Antique Shop Argentinian Restaurant Art Gallery Art Museum Arts & Crafts Store Asian Restaurant Auvergne Restaurant Baby Store Bakery Bank Bar Basque Restaurant Bed & Breakfast Beer Bar Beer Garden Beer Store Bike Rental / Bike Share Bistro Boat or Ferry Bookstore Boutique Boxing Gym Brasserie Brazilian Restaurant Breakfast Spot Brewery Bridge Bubble Tea Shop Burger Joint Bus Station Bus Stop Café Cambodian Restaurant Canal Candy Store Cheese Shop Chinese Restaurant ... Scandinavian Restaurant Science Museum Sculpture Garden Seafood Restaurant Shanxi Restaurant Shoe Store Shopping Mall Smoke Shop Snack Place Southwestern French Restaurant Souvlaki Shop Spa Spanish Restaurant Speakeasy Sporting Goods Shop Sports Bar Steakhouse Supermarket Sushi Restaurant Szechuan Restaurant Taco Place Tailor Shop Tapas Restaurant Tea Room Thai Restaurant Theater Thrift / Vintage Store Tibetan Restaurant Toy / Game Store Trail Turkish Restaurant Udon Restaurant Vegetarian / Vegan Restaurant Venezuelan Restaurant Vietnamese Restaurant Wine Bar Wine Shop Women's Store Zoo Zoo Exhibit

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ... 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ... 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 ... 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 ... 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ... 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ...

1319 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ... 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1320 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ... 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1321 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ... 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1322 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ... 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1323 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 ... 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1324 rows × 198 columns

Adding Neighbourhoods to our Data

In [93]:

Paris\_venue\_cat['Neighbourhood'] = venues\_in\_Paris['Neighbourhood']

# moving neighborhood column to the first column

fixed\_columns = [Paris\_venue\_cat.columns[-1]] + list(Paris\_venue\_cat.columns[:-1])

Paris\_venue\_cat = Paris\_venue\_cat[fixed\_columns]

Paris\_venue\_cat.head()

Out[93]:

Neighbourhood Afghan Restaurant African Restaurant American Restaurant Antique Shop Argentinian Restaurant Art Gallery Art Museum Arts & Crafts Store Asian Restaurant Auvergne Restaurant Baby Store Bakery Bank Bar Basque Restaurant Bed & Breakfast Beer Bar Beer Garden Beer Store Bike Rental / Bike Share Bistro Boat or Ferry Bookstore Boutique Boxing Gym Brasserie Brazilian Restaurant Breakfast Spot Brewery Bridge Bubble Tea Shop Burger Joint Bus Station Bus Stop Café Cambodian Restaurant Canal Candy Store Cheese Shop ... Scandinavian Restaurant Science Museum Sculpture Garden Seafood Restaurant Shanxi Restaurant Shoe Store Shopping Mall Smoke Shop Snack Place Southwestern French Restaurant Souvlaki Shop Spa Spanish Restaurant Speakeasy Sporting Goods Shop Sports Bar Steakhouse Supermarket Sushi Restaurant Szechuan Restaurant Taco Place Tailor Shop Tapas Restaurant Tea Room Thai Restaurant Theater Thrift / Vintage Store Tibetan Restaurant Toy / Game Store Trail Turkish Restaurant Udon Restaurant Vegetarian / Vegan Restaurant Venezuelan Restaurant Vietnamese Restaurant Wine Bar Wine Shop Women's Store Zoo Zoo Exhibit

0 PARIS-10E-ARRONDISSEMENT 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ... 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 PARIS-10E-ARRONDISSEMENT 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ... 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

2 PARIS-10E-ARRONDISSEMENT 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 ... 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

3 PARIS-10E-ARRONDISSEMENT 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 ... 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

4 PARIS-10E-ARRONDISSEMENT 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ... 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

5 rows × 199 columns

Venue categories mean value

We will group the Neighbourhoods and calculate the mean venue categories value in each Neighbourhood

In [94]:

Paris\_grouped = Paris\_venue\_cat.groupby('Neighbourhood').mean().reset\_index()

Paris\_grouped.head()

Out[94]:

Neighbourhood Afghan Restaurant African Restaurant American Restaurant Antique Shop Argentinian Restaurant Art Gallery Art Museum Arts & Crafts Store Asian Restaurant Auvergne Restaurant Baby Store Bakery Bank Bar Basque Restaurant Bed & Breakfast Beer Bar Beer Garden Beer Store Bike Rental / Bike Share Bistro Boat or Ferry Bookstore Boutique Boxing Gym Brasserie Brazilian Restaurant Breakfast Spot Brewery Bridge Bubble Tea Shop Burger Joint Bus Station Bus Stop Café Cambodian Restaurant Canal Candy Store Cheese Shop ... Scandinavian Restaurant Science Museum Sculpture Garden Seafood Restaurant Shanxi Restaurant Shoe Store Shopping Mall Smoke Shop Snack Place Southwestern French Restaurant Souvlaki Shop Spa Spanish Restaurant Speakeasy Sporting Goods Shop Sports Bar Steakhouse Supermarket Sushi Restaurant Szechuan Restaurant Taco Place Tailor Shop Tapas Restaurant Tea Room Thai Restaurant Theater Thrift / Vintage Store Tibetan Restaurant Toy / Game Store Trail Turkish Restaurant Udon Restaurant Vegetarian / Vegan Restaurant Venezuelan Restaurant Vietnamese Restaurant Wine Bar Wine Shop Women's Store Zoo Zoo Exhibit

0 PARIS-10E-ARRONDISSEMENT 0.000000 0.02 0.0 0.0 0.0 0.0 0.000000 0.0 0.030000 0.0 0.0 0.020000 0.000000 0.020000 0.0 0.0 0.0 0.01 0.0 0.0 0.070000 0.0 0.01 0.0 0.01 0.000000 0.0 0.02 0.0 0.0 0.0 0.02 0.0 0.0 0.040000 0.000000 0.0 0.0 0.0 ... 0.0 0.0 0.0 0.02 0.01 0.0 0.01 0.0 0.0 0.0 0.0 0.0 0.01 0.0 0.0 0.01 0.0 0.0 0.0 0.0 0.01 0.0 0.0 0.000000 0.020000 0.0 0.0 0.0 0.0 0.000000 0.0 0.0 0.010000 0.0 0.000000 0.020000 0.02 0.0 0.0 0.0

1 PARIS-11E-ARRONDISSEMENT 0.021739 0.00 0.0 0.0 0.0 0.0 0.021739 0.0 0.043478 0.0 0.0 0.043478 0.000000 0.043478 0.0 0.0 0.0 0.00 0.0 0.0 0.021739 0.0 0.00 0.0 0.00 0.000000 0.0 0.00 0.0 0.0 0.0 0.00 0.0 0.0 0.065217 0.000000 0.0 0.0 0.0 ... 0.0 0.0 0.0 0.00 0.00 0.0 0.00 0.0 0.0 0.0 0.0 0.0 0.00 0.0 0.0 0.00 0.0 0.0 0.0 0.0 0.00 0.0 0.0 0.000000 0.000000 0.0 0.0 0.0 0.0 0.000000 0.0 0.0 0.043478 0.0 0.021739 0.043478 0.00 0.0 0.0 0.0

2 PARIS-12E-ARRONDISSEMENT 0.000000 0.00 0.0 0.0 0.0 0.0 0.000000 0.0 0.000000 0.0 0.0 0.000000 0.000000 0.000000 0.0 0.0 0.0 0.00 0.0 0.0 0.200000 0.0 0.00 0.0 0.00 0.000000 0.0 0.00 0.0 0.0 0.0 0.00 0.0 0.0 0.000000 0.000000 0.0 0.0 0.0 ... 0.0 0.0 0.0 0.00 0.00 0.0 0.00 0.0 0.0 0.0 0.0 0.0 0.00 0.0 0.0 0.00 0.0 0.2 0.0 0.0 0.00 0.0 0.0 0.000000 0.000000 0.0 0.0 0.0 0.0 0.000000 0.0 0.0 0.000000 0.0 0.000000 0.000000 0.00 0.0 0.2 0.2

3 PARIS-13E-ARRONDISSEMENT 0.000000 0.00 0.0 0.0 0.0 0.0 0.000000 0.0 0.196721 0.0 0.0 0.016393 0.000000 0.000000 0.0 0.0 0.0 0.00 0.0 0.0 0.000000 0.0 0.00 0.0 0.00 0.000000 0.0 0.00 0.0 0.0 0.0 0.00 0.0 0.0 0.000000 0.016393 0.0 0.0 0.0 ... 0.0 0.0 0.0 0.00 0.00 0.0 0.00 0.0 0.0 0.0 0.0 0.0 0.00 0.0 0.0 0.00 0.0 0.0 0.0 0.0 0.00 0.0 0.0 0.000000 0.081967 0.0 0.0 0.0 0.0 0.016393 0.0 0.0 0.000000 0.0 0.245902 0.000000 0.00 0.0 0.0 0.0

4 PARIS-14E-ARRONDISSEMENT 0.000000 0.00 0.0 0.0 0.0 0.0 0.000000 0.0 0.000000 0.0 0.0 0.071429 0.035714 0.000000 0.0 0.0 0.0 0.00 0.0 0.0 0.071429 0.0 0.00 0.0 0.00 0.035714 0.0 0.00 0.0 0.0 0.0 0.00 0.0 0.0 0.035714 0.000000 0.0 0.0 0.0 ... 0.0 0.0 0.0 0.00 0.00 0.0 0.00 0.0 0.0 0.0 0.0 0.0 0.00 0.0 0.0 0.00 0.0 0.0 0.0 0.0 0.00 0.0 0.0 0.035714 0.000000 0.0 0.0 0.0 0.0 0.000000 0.0 0.0 0.000000 0.0 0.000000 0.000000 0.00 0.0 0.0 0.0

5 rows × 199 columns

Top venue categories

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

In [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()

Out[95]:

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 PARIS-10E-ARRONDISSEMENT French Restaurant Bistro Coffee Shop Café Japanese Restaurant Italian Restaurant Indian Restaurant Asian Restaurant Hotel Pizza Place

1 PARIS-11E-ARRONDISSEMENT Restaurant French Restaurant Café Bar Italian Restaurant Wine Bar Cocktail Bar Vegetarian / Vegan Restaurant Pastry Shop Asian Restaurant

2 PARIS-12E-ARRONDISSEMENT Zoo Exhibit Bistro Monument / Landmark Supermarket Zoo Argentinian Restaurant Exhibit French Restaurant Fountain Food & Drink Shop

3 PARIS-13E-ARRONDISSEMENT Vietnamese Restaurant Asian Restaurant Thai Restaurant Chinese Restaurant French Restaurant Juice Bar Hotel Cambodian Restaurant Sandwich Place Dance Studio

4 PARIS-14E-ARRONDISSEMENT French Restaurant Hotel Japanese Restaurant Pizza Place Bistro Bakery Brasserie Bank Pet Store Café

Model Building

K Means

Let's cluster the city of Paris to roughly 5 to make it easier to analyze.

We use the K Means clustering technique to do so.

In [96]:

# 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

Out[96]:

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 Clustered Data

In [97]:

kmeans\_Paris.labels\_

Out[97]:

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

So our model has labeled the city, we insert it in our data.

In [98]:

neighborhoods\_venues\_sorted\_paris.insert(0, 'Cluster Labels', kmeans\_Paris.labels\_ +1)

Join paris\_combined\_data with our neighbourhood venues sorted to add latitude & longitude for each of the neighborhood to prepare it for plotting

In [99]:

paris\_data = paris\_combined\_data

paris\_data = paris\_data.join(neighborhoods\_venues\_sorted\_paris.set\_index('Neighbourhood'), on='nom\_comm')

paris\_data.head()

Out[99]:

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 8th Most Common Venue 9th Most Common Venue 10th Most Common Venue

0 75010 PARIS-10E-ARRONDISSEMENT PARIS 48.876029 2.361113 5 French Restaurant Bistro Coffee Shop Café Japanese Restaurant Italian Restaurant Indian Restaurant Asian Restaurant Hotel Pizza Place

1 75016 PARIS-16E-ARRONDISSEMENT PARIS 48.860399 2.262100 4 Plaza Bike Rental / Bike Share Lake Art Museum Bus Station Bus Stop Boat or Ferry Park French Restaurant Pool

2 75009 PARIS-9E-ARRONDISSEMENT PARIS 48.876896 2.337460 5 French Restaurant Hotel Japanese Restaurant Bistro Tea Room Wine Bar Lounge Cocktail Bar Bakery Pizza Place

3 75015 PARIS-15E-ARRONDISSEMENT PARIS 48.840155 2.293559 5 Italian Restaurant Hotel French Restaurant Restaurant Thai Restaurant Park Brasserie Lebanese Restaurant Japanese Restaurant Coffee Shop

4 75002 PARIS-2E-ARRONDISSEMENT PARIS 48.867903 2.344107 5 Cocktail Bar French Restaurant Bakery Italian Restaurant Coffee Shop Hotel Wine Bar Bar Thai Restaurant Bistro

Drop all the NaN values to prevent data skew

In [100]:

paris\_data\_nonan = paris\_data.dropna(subset=['Cluster Labels'])

Visualizing the clustered neighbourhood

Let's plot the clusters

In [101]:

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

Out[101]:

Make this Notebook Trusted to load map: File -> Trust Notebook

Examining our Clusters

Cluster 1

In [102]:

paris\_data\_nonan.loc[paris\_data\_nonan['Cluster Labels'] == 1, paris\_data\_nonan.columns[[1] + list(range(5, paris\_data\_nonan.shape[1]))]]

Out[102]:

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

14 PARIS-12E-ARRONDISSEMENT 1 Zoo Exhibit Bistro Monument / Landmark Supermarket Zoo Argentinian Restaurant Exhibit French Restaurant Fountain Food & Drink Shop

Cluster 2

In [103]:

paris\_data\_nonan.loc[paris\_data\_nonan['Cluster Labels'] == 2, paris\_data\_nonan.columns[[1] + list(range(5, paris\_data\_nonan.shape[1]))]]

Out[103]:

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

13 PARIS-13E-ARRONDISSEMENT 2 Vietnamese Restaurant Asian Restaurant Thai Restaurant Chinese Restaurant French Restaurant Juice Bar Hotel Cambodian Restaurant Sandwich Place Dance Studio

Cluster 3

In [104]:

paris\_data\_nonan.loc[paris\_data\_nonan['Cluster Labels'] == 3, paris\_data\_nonan.columns[[1] + list(range(5, paris\_data\_nonan.shape[1]))]]

Out[104]:

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-8E-ARRONDISSEMENT 3 French Restaurant Hotel Art Gallery Cocktail Bar Spa Furniture / Home Store Grocery Store Italian Restaurant Mediterranean Restaurant Middle Eastern Restaurant

15 PARIS-7E-ARRONDISSEMENT 3 Hotel French Restaurant Café Italian Restaurant Plaza History Museum Coffee Shop Cocktail Bar Bistro Ice Cream Shop

18 PARIS-17E-ARRONDISSEMENT 3 French Restaurant Hotel Italian Restaurant Bakery Plaza Japanese Restaurant Café Bar Bistro Breakfast Spot

19 PARIS-14E-ARRONDISSEMENT 3 French Restaurant Hotel Japanese Restaurant Pizza Place Bistro Bakery Brasserie Bank Pet Store Café

Cluster 4

In [105]:

paris\_data\_nonan.loc[paris\_data\_nonan['Cluster Labels'] == 4, paris\_data\_nonan.columns[[1] + list(range(5, paris\_data\_nonan.shape[1]))]]

Out[105]:

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

1 PARIS-16E-ARRONDISSEMENT 4 Plaza Bike Rental / Bike Share Lake Art Museum Bus Station Bus Stop Boat or Ferry Park French Restaurant Pool

Cluster 5

In [106]:

paris\_data\_nonan.loc[paris\_data\_nonan['Cluster Labels'] == 5, paris\_data\_nonan.columns[[1] + list(range(5, paris\_data\_nonan.shape[1]))]]

Out[106]:

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-10E-ARRONDISSEMENT 5 French Restaurant Bistro Coffee Shop Café Japanese Restaurant Italian Restaurant Indian Restaurant Asian Restaurant Hotel Pizza Place

2 PARIS-9E-ARRONDISSEMENT 5 French Restaurant Hotel Japanese Restaurant Bistro Tea Room Wine Bar Lounge Cocktail Bar Bakery Pizza Place

3 PARIS-15E-ARRONDISSEMENT 5 Italian Restaurant Hotel French Restaurant Restaurant Thai Restaurant Park Brasserie Lebanese Restaurant Japanese Restaurant Coffee Shop

4 PARIS-2E-ARRONDISSEMENT 5 Cocktail Bar French Restaurant Bakery Italian Restaurant Coffee Shop Hotel Wine Bar Bar Thai Restaurant Bistro

5 PARIS-11E-ARRONDISSEMENT 5 Restaurant French Restaurant Café Bar Italian Restaurant Wine Bar Cocktail Bar Vegetarian / Vegan Restaurant Pastry Shop Asian Restaurant

6 PARIS-5E-ARRONDISSEMENT 5 French Restaurant Hotel Italian Restaurant Bakery Plaza Café Pub Coffee Shop Asian Restaurant Bistro

7 PARIS-19E-ARRONDISSEMENT 5 French Restaurant Bar Supermarket Hotel Brewery Seafood Restaurant Sushi Restaurant Beer Bar Bistro Japanese Restaurant

8 PARIS-20E-ARRONDISSEMENT 5 Bakery Plaza Bistro Bar French Restaurant Japanese Restaurant Italian Restaurant Café Park Hotel

9 PARIS-3E-ARRONDISSEMENT 5 French Restaurant Coffee Shop Italian Restaurant Burger Joint Bistro Bakery Japanese Restaurant Cocktail Bar Art Gallery Chinese Restaurant

10 PARIS-6E-ARRONDISSEMENT 5 French Restaurant Italian Restaurant Bistro Chocolate Shop Bakery Plaza Seafood Restaurant Restaurant Pub Fountain

11 PARIS-18E-ARRONDISSEMENT 5 French Restaurant Bar Italian Restaurant Restaurant Hotel Pizza Place Bistro Supermarket Vietnamese Restaurant Plaza

16 PARIS-1ER-ARRONDISSEMENT 5 Japanese Restaurant French Restaurant Hotel Ramen Restaurant Plaza Bakery Café Coffee Shop Korean Restaurant Art Museum

17 PARIS-4E-ARRONDISSEMENT 5 French Restaurant Ice Cream Shop Clothing Store Pastry Shop Italian Restaurant Hotel Gay Bar Seafood Restaurant Pedestrian Plaza Bistro

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.

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 gateaway with a lot of places to explore, beautiful landscapes 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.