Capstone Project – Final Report

Finding Safe Neighbourhoods for Living for International Students in London

Senpei Peng

1 Introduction

London is one of the most popular study destinations and has been ranked the 1st for Best Student Cities. It is also a popular destination for international students looking for higher education experiences. According to the Evening Standard, there are more than 120,000 international students in London in the year of 2018/19. Except for the expensive rent in London, many students are also concerned about the safety of the areas for living.

This project is going to review the boroughs in London based on total crimes numbers and types of each area, top venues in the borough will also be explored. In addition, the neighbourhoods will be clustered using k-means clustering. The results can be a reference for International students planning to find a safe and cosy home in London.

2 Data Source

The data used in this study requires both the number of crimes committed during the past few years in each borough and the most popular venues within these areas. The safest borough and the top 10 venues in each area is analysed. In the section of finding the popular venues, foursquare data will be needed as location data.

The dataset used for crime is from Kaggle.com, containing 13 million crimes classified by the borough in which they occurred, together with their category and time. The link to the data source is https://www.kaggle.com/jboysen/london-crime.

After the treatment of the crime data for all the boroughs in London, the dataset is complemented with additional information from the table on Wikipedia webpage: List of London boroughs (https://en.wikipedia.org/wiki/List_of_London_boroughs).

Once the safest borough is selected (Kingston upon Thames), the neighbourhoods and districts of this borough are found on the Wikipedia webpage: List of districts in the Royal Borough of Kingston upon Thames (https://en.wikipedia.org/wiki/List_of_districts_in_the_Royal_Borough_of_Kingston_upon_Thames).

The top venues are explored in each neighbourhood using the data from Foursquare using the developer API (https://foursquare.com/developers/apps).

3 Methodology

3.1 Crime Data

Import the necessary libraries.
Import the crime data from the CSV file downloaded from the data source.

```
# Import necessary libraries
import numpy as np
import pandas as pd
from bs4 import BeautifulSoup

!conda install -c conda-forge geocoder --yes
import geocoder
!conda install -c conda-forge geopy --yes
from geopy.geocoders import Nominatim
from pandas.io.json import json_normalize

import requests
import random

from IPython.display import Image
from IPython.core.display import HTML

from pandas.io.json import json_normalize
import folium

print ('Libraries import completed.')
```

```
# Load data

df = pd.read_csv ('crime.csv')
df.head (10)
```

	lsoa_code	borough	major_category	minor_category	value	year	month
0	E01001116	Croydon	Burglary	Burglary in Other Buildings	0	2016	11
1	E01001646	Greenwich	Violence Against the Person	Other violence	0	2016	11
2	E01000677	Bromley	Violence Against the Person	Other violence	0	2015	5
3	E01003774	Redbridge	Burglary	Burglary in Other Buildings	0	2016	3
4	E01004563	Wandsworth	Robbery	Personal Property	0	2008	6
5	E01001320	Ealing	Theft and Handling	Other Theft	0	2012	5

Process the table to remain only the data of year 2016 (otherwise the data frame will be difficult to process as it contains 13 million entries).

```
# Drop the results of years 2008 - 2015 to reduce the process demand
df.drop (df.index[df['year'] != 2016], inplace = True)

df = df.reset_index (drop = True)
df.shape
```

```
(1498956, 7)
```

Sum up the total number of crimes in each borough in 2016.

```
# Sum up the total number of crimes in each borough in 2016
df ['borough'].value_counts ()
```

```
Lambeth                 17605
Southwark               16560
Croydon                 16254
Newham                  15622
Ealing                  15284
Tower Hamlets           15219
```

Calculate the number of crimes in each major category, grouped by borough, and the total number of crimes for each borough.

```
# Showing the number of crimes in each major category by borough
crime_sum = pd.pivot_table (df, values = ['value'], index = ['borough'], co
                            aggfunc = np.sum, fill_value = 0)

crime_sum.reset_index (inplace = True)

# Showing the total number of each borough
crime_sum ['total']  = crime_sum.sum (axis = 1)

crime_sum.columns = crime_sum.columns.map ('. join)
crime_sum
```

	borough	valueBurglary	valueCriminal Damage	valueDrugs	valueOther Notifiable Offences	valueRobbery	valueTheft and Handling
0	Barking and Dagenham	1287	1949	919	378	534	5607
1	Barnet	3402	2183	906	499	464	9731
2	Bexley	1123	1673	646	294	209	4392
3	Brent	2631	2280	2096	536	919	9026
4	Bromley	2214	2202	728	417	369	7584
5	Camden	2652	1935	1493	490	899	14088

3.2 Location Data for Boroughs

Import data from Wikipedia page and use BeautifulSoup to parse the xml code.

```
# Importing data from Wikipedia
wiki_url = 'https://en.wikipedia.org/wiki/List_of_London_boroughs'
wiki_raw_data = requests.get (wiki_url).text

# Use beautiful soup to parse the xml code
soup = BeautifulSoup (wiki_raw_data, 'html')
soup.prettify ()
```

```
'<!DOCTYPE html>\n<html class="client-nojs" dir="ltr" lang="en">\n <hea
d>\n  <meta charset="utf-8"/>\n  <title>\n   List of London boroughs -
Wikipedia\n  </title>\n  <script>\n   document.documentElement.className
e="client-js";RLCONF={"wgBreakFrames":!1,"wgSeparatorTransformTable":
["",""],"wgDigitTransformTable":["",""],"wgDefaultDateFormat":"dmy","wg
MonthNames":["","January","February","March","April","May","June","Jul
y","August","September","October","November","December"],"wgRequestI
```

Convert the raw data into a data frame and then read into a table.

```
# Convert the raw data into data frame
table = soup.find_all ('table', {'class': 'wikitable sortable'})
print (table)
```

```
borough_table = pd.read_html(str(table[0]), index_col=None, header=0) [0]
borough_table.head ()
```

	Borough	Inner	Status	Local authority	Political control	Headquarters	Area (sq mi)	Population (2013 est) [1]	Co-ordinates
0	Barking and Dagenham [note 1]	NaN	NaN	Barking and Dagenham London Borough Council	Labour	Town Hall, 1 Town Square	13.93	194352	51°33′39″N 0°09′21″E / 51.5607°N 0.1557°E
1	Barnet	NaN	NaN	Barnet London Borough Council	Conservative	Barnet House, 2 Bristol Avenue, Colindale	33.49	369088	51°37′31″N 0°09′06″W / 51.6252°N 0.1517°W
2	Bexley	NaN	NaN	Bexley London Borough Council	Conservative	Civic Offices, 2 Watling Street	23.38	236687	51°27′18″N 0°09′02″E / 51.4549°N 0.1505°E
3	Brent	NaN	NaN	Brent London Borough Council	Labour	Brent Civic Centre, Engineers Way	16.70	317264	51°33′32″N 0°16′54″W / 51.5588°N 0.2817°W

Add the data of City of London as it appears in a separate table.

Clean the text in the table.

```
# Append the table of City of London

city_table = pd.read_html (str (table[1]), index_col = None, header = 0) [(
city_table.columns = ['Borough','Inner','Status','Local authority','Politic
                      'Area (sq mi)','Population (2013 est)[1]','Co-ordinat
borough_table = borough_table.append (city_table, ignore_index = True)
borough_table.tail()
```

32	City of London	([note 5]	Sui generis;City;Ceremonial county	Corporation of London;Inner Temple;Middle Temple		?	Guildhall	1.12

```
# Clean the text in the table

borough_table = borough_table.replace ('note 1', '', regex = True)\
.replace ('note 2', '', regex = True)\
.replace ('note 3', '', regex = True)\
.replace ('note 4', '', regex = True)\
.replace ('note 5', '', regex = True)

borough_table.head (10)
```

```
# Clean the brackets
borough_table1 = borough_table.replace ('\[\]', '', regex = True)
borough_table1.head (10)
```

	Borough	Inner	Status	Local authority	Political control	Headquarters	Area (sq mi)	Population (2013 est) [1]	Co-ordinates
0	Barking and Dagenham	NaN	NaN	Barking and Dagenham London Borough Council	Labour	Town Hall, 1 Town Square	13.93	194352	51°33′39″N 0°09′21″E / 51.5607°N 0.1557°E
1	Barnet	NaN	NaN	Barnet London Borough Council	Conservative	Barnet House, 2 Bristol Avenue, Colindale	33.49	369088	51°37′31″N 0°09′06″W / 51.6252°N 0.1517°W
2	Bexley	NaN	NaN	Bexley London Borough Council	Conservative	Civic Offices, 2 Watling Street	23.38	236687	51°27′18″N 0°09′02″E / 51.4549°N 0.1505°E
3	Brent	NaN	NaN	Brent London Borough Council	Labour	Brent Civic Centre, Engineers Way	16.70	317264	51°33′32″N 0°16′54″W / 51.5588°N 0.2817°W

Append the location data with the table of number of crimes, drop the columns containing unnecessary information.

```
# Combine the data frames of the crimes and location data
ldn_crime = pd.merge (borough_table1, crime_sum, on = 'Borough')
ldn_crime.head ()
```

```
# Drop the columns containing not relevant data
ldn_crime.drop (columns = ['Inner', 'Status', 'Political control', 'Headqua
ldn_crime.head ()
```

	Borough	Local authority	Area (sq mi)	Population (2013 est) [1]	Co-ordinates	Nr. in map	Burglary	Criminal Damage	Drug Crimes	Other Offences
0	Barnet	Barnet London Borough Council	33.49	369088	51°37′31″N 0°09′06″W / 51.6252°N 0.1517°W	31	3402	2183	906	499
1	Bexley	Bexley London Borough Council	23.38	236687	51°27′18″N 0°09′02″E / 51.4549°N 0.1505°E	23	1123	1673	646	294

3.3 Exploratory Data Analysis – Finding the Safest Borough

Import the libraries.

```
# Import libraries

%matplotlib inline

import matplotlib as mpl
import matplotlib.pyplot as plt
import matplotlib.cm as cm
import matplotlib.colors as colors

print ('Import Completed')
```

Rearrange the table of crimes with location data by the name of boroughs.

```
# Rearrage the table by boroughs
ldn_crime.sort_values (['Total'], ascending = True, axis = 0, inplace = Tru
ldn_crime.head (10)
```

	Borough	Local authority	Area (sq mi)	Population (2013 est) [1]	Co-ordinates	Nr. in map	Burglary	Criminal Damage	Drug Crimes	Off
29	City of London	Corporation of London;Inner Temple;Middle Temple	1.12	7000	51°30′56″N 0°05′32″W / 51.5155°N 0.0922°W	1	2	2	10	
16	Kingston upon Thames	Kingston upon Thames London Borough Council	14.38	166793	51°24′31″N 0°18′23″W / 51.4085°N 0.3064°W	16	879	1054	743	

Obtain the safest 10 boroughs by the total number of crimes.

```
# List of 10 safest boroughs in London
ldn_safe10 = ldn_crime.head (10)
```

The boroughs of least crimes committed in 2016 are: City of London, Kingston upon Thames, Sutton, Richmond upon Thames, Merton, Bexley, Harrow, Havering, Redbridge, Kensingtong and Chealsea

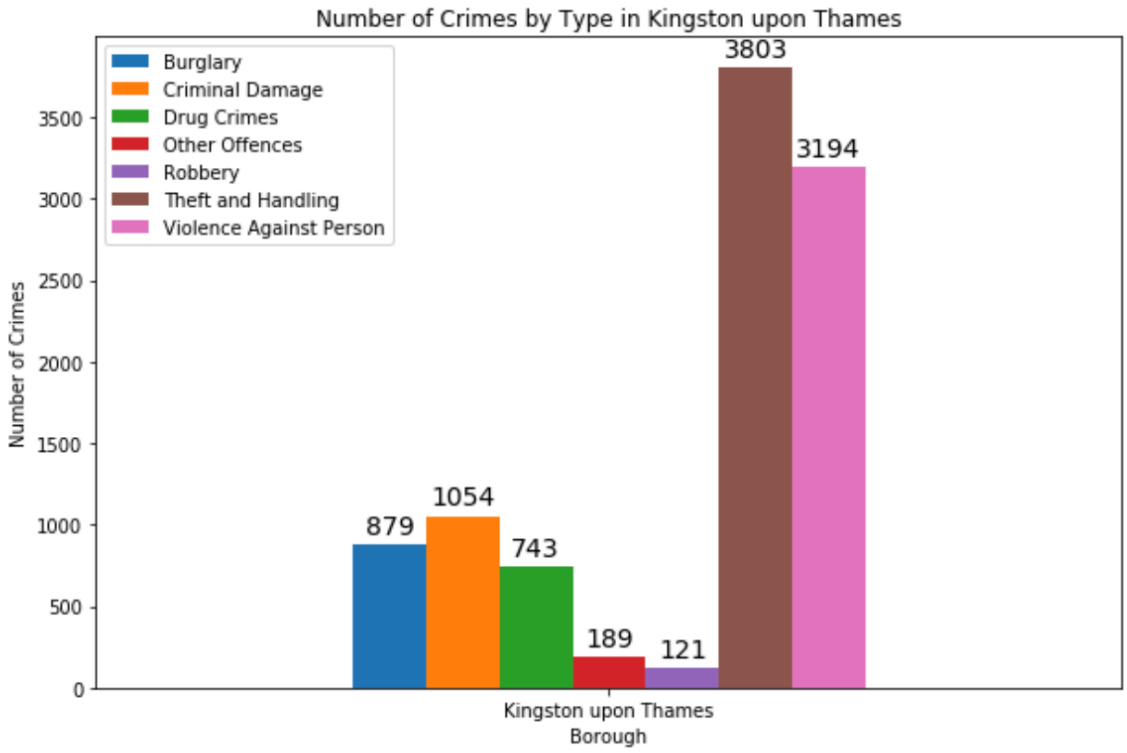
Exclude the City of London as it is not an appropriate borough for the subsequent analysis. The borough of Kingston upon Thames is selected.

It is notable that the borough City of London has an area of only 1.12 square miles, not comparable to other boroughs with much larger areas. It is also an area where a lot of business located in but not living districts. In order to find the best areas for living, the analysis will exclude the results of City of London.

Show the number of crimes in each major category in Kingston upon Thames.

```
# Plot the crime types of borough with the second least crime 'Kingston upo
kingston =  ldn_safe10[ldn_safe10 ['Borough'] == 'Kingston upon Thames']

kingston_crime = kingston[['Borough','Burglary','Criminal Damage','Drug Cri
                  'Robbery','Theft and Handling','Violence Against Person']]

kingston_crime.set_index('Borough',inplace = True)

ax = kingston_crime.plot(kind='bar', figsize=(10, 6.5), rot=0)

ax.set_ylabel('Number of Crimes') # add to x-label to the plot
ax.set_xlabel('Borough') # add y-label to the plot
ax.set_title('Number of Crimes by Type in Kingston upon Thames') # add titl

# Creating a function to display the percentage.

for p in ax.patches:
    ax.annotate(np.round(p.get_height(),decimals=2),
                (p.get_x()+p.get_width()/2., p.get_height()),
                ha='center',
                va='center',
                xytext=(0, 10),
                textcoords='offset points',
                fontsize = 14
               )

plt.show()
```

Plot the numbers in a bar chart.



3.4 Explore the Neighbourhoods in Kingston (The Safest Borough from Analysis)

From the Wikipedia page, create a table containing the neighbourhoods in Kingston.

```
# Create a Table of Kingston
Neighbourhood = ['Berrylands','Canbury','Chessington','Coombe','Hook','King
'Kingston Vale','Malden Rushett','Motspur Park','New Malden','Norbiton',
'Old Malden','Seething Wells','Surbiton','Tolworth']

Borough = ['Kingston upon Thames','Kingston upon Thames','Kingston upon Tha
           'Kingston upon Thames','Kingston upon Thames','Kingston upon Thar
           'Kingston upon Thames','Kingston upon Thames','Kingston upon Thar
           'Kingston upon Thames','Kingston upon Thames','Kingston upon Thar

Latitude = ['','','','','','','','','','','','','','']
Longitude = ['','','','','','','','','','','','','','']

df_kingston = {'Neighbourhood': Neighbourhood,'Borough':Borough,'Latitude':
kingston_neigh = pd.DataFrame(data = df_kingston, columns=['Neighbourhood',

kingston_neigh
```

	Neighbourhood	Borough	Latitude	Longitude
0	Berrylands	Kingston upon Thames		
1	Canbury	Kingston upon Thames		
2	Chessington	Kingston upon Thames		
3	Coombe	Kingston upon Thames		

Append the location data of each neighbourhood in Kingston table.

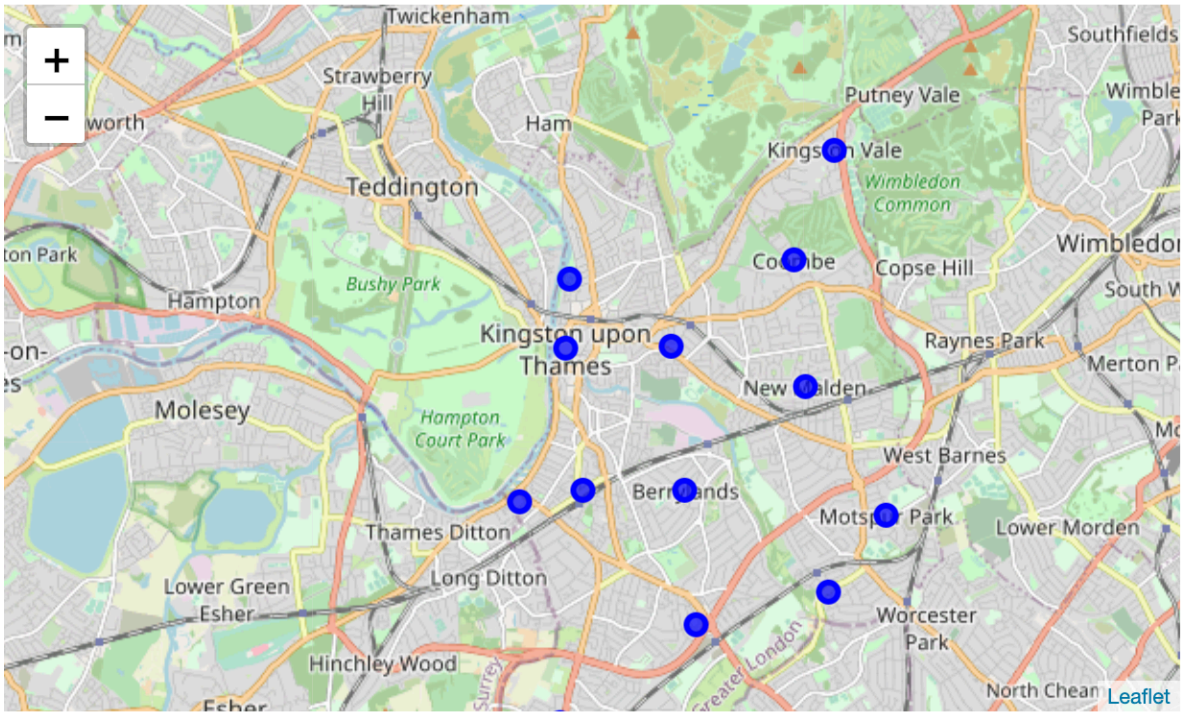
```
# Find the Co-ordinates for each Neighbourhood in the Borough

Latitude = []
Longitude = []

for i in range(len(Neighbourhood)):
    address = '{},London,United Kingdom'.format(Neighbourhood[i])
    geolocator = Nominatim(user_agent="London_agent")
    location = geolocator.geocode(address)
    Latitude.append(location.latitude)
    Longitude.append(location.longitude)
print(Latitude, Longitude)
```

```
df_kingston = {'Neighbourhood': Neighbourhood,'Borough':Borough,'Latitude':
kingston_neigh = pd.DataFrame(data = df_kingston, columns=['Neighbourhood',

kingston_neigh
```

	Neighbourhood	Borough	Latitude	Longitude
0	Berrylands	Kingston upon Thames	51.393781	-0.284802
1	Canbury	Kingston upon Thames	51.417499	-0.305553
2	Chessington	Kingston upon Thames	51.358336	-0.298622
3	Coombe	Kingston upon Thames	51.419450	-0.265398

Visualise the neighbourhoods in Kingston on a map.

```
# Address of the central neighbourhood
address = 'Kingston upon Thames, London, United Kingdom'

geolocator = Nominatim(user_agent="ld_explorer")
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude
print('The geograpical coordinate of Berrylands, London are {}, {}.'.format

# Get the location data of Centre of Kington upon Thames
kingston_map = folium.Map(location = [latitude, longitude], zoom_start = 12

# add markers to map
for lat, lng, borough, neighbourhood in zip(kingston_neigh['Latitude'], kir
    label = '{}, {}'.format(neighbourhood, borough)
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius = 5,
        popup = label,
        color = 'blue',
        fill = True,
        fill_color = 'blue',
        fill_opacity = 0.7,
        parse_html = False).add_to(kingston_map)

kingston_map
```



Getting the popular venues in each neighbourhood using the data returned by Foursquare.

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

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

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

    nearby_venues = pd.DataFrame([item for venue_list in venues_list for it
    nearby_venues.columns = ['Neighbourhood',
                  'Neighbourhood Latitude',
                  'Neighbourhood Longitude',
                  'Venue',
                  'Venue Latitude',
                  'Venue Longitude',
                  'Venue Category']

    return(nearby_venues)
```

```
# The venues in Kingston

kingston_venues = getNearbyVenues (names = kingston_neigh ['Neighbourhood']
                                   latitudes = kingston_neigh ['Latitude'],
                                   longitudes = kingston_neigh ['Longitude']
```

```
Berrylands
Canbury
Chessington
Coombe
Hook
Kingston upon Thames
Kingston Vale
Malden Rushett
```

Arrange the neighbourhoods with the venues returned by Foursquare.

```
kingston_venues.groupby ('Neighbourhood').count()
```

Neighbourhood	Neighbourhood Latitude	Neighbourhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
Berrylands	5	5	5	5	5	5
Canbury	14	14	14	14	14	14
Coombe	1	1	1	1	1	1
Hook	4	4	4	4	4	4
Kingston Vale	4	4	4	4	4	4

Create a table of Kingston neighbourhoods with One hot encoding.

```
# One hot encoding
kingston_onehot = pd.get_dummies(kingston_venues[['Venue Category']], prefi

# add neighborhood column back to dataframe
kingston_onehot['Neighbourhood'] = kingston_venues['Neighbourhood']

# move neighborhood column to the first column
fixed_columns = [kingston_onehot.columns[-1]] + list(kingston_onehot.columr
kingston_onehot = kingston_onehot[fixed_columns]

kingston_onehot.head()
```

	Neighbourhood	Asian Restaurant	Athletics & Sports	Auto Garage	Bakery	Bar	Beer Bar	Bistro	Bowling Alley	Breakfast Spot
0	Berrylands	0	0	0	0	0	0	0	0	0
1	Berrylands	0	0	0	0	0	0	0	0	0

Fetch the occurrence frequency of each type of venue.

```
# Get the occurence frequency of each type of venues

kingston_grouped = kingston_onehot.groupby ('Neighbourhood').mean ().reset_
kingston_grouped
```

	Neighbourhood	Asian Restaurant	Athletics & Sports	Auto Garage	Bakery	Bar	Beer Bar	Bistro	Bowlir Alle
0	Berrylands	0.000000	0.00	0.00	0.000000	0.000000	0.000000	0.000000	0.00000
1	Canbury	0.000000	0.00	0.00	0.000000	0.000000	0.000000	0.000000	0.00000
2	Coombe	0.000000	0.00	0.00	0.000000	0.000000	0.000000	0.000000	0.00000
3	Hook	0.000000	0.00	0.00	0.250000	0.000000	0.000000	0.000000	0.00000
4	Kingston Vale	0.000000	0.00	0.00	0.000000	0.250000	0.000000	0.000000	0.00000

Get the top venues for each neighbourhood and arrange in a data frame.

List the venues in a data frame

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

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

```
num_top_venues = 10

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

# create columns according to number of top venues
columns = ['Neighbourhood']
for ind in np.arange(num_top_venues):
    try:
        columns.append('{}{} Most Common Venue'.format(ind+1, indicators[ir
    except:
        columns.append('{}th Most Common Venue'.format(ind+1))

# create a new dataframe
neighbourhoods_venues_sorted = pd.DataFrame(columns = columns)
neighbourhoods_venues_sorted['Neighbourhood'] = kingston_grouped['Neighbour

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

neighbourhoods_venues_sorted.head()
```

	Neighbourhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue
0	Berrylands	Gym / Fitness Center	Park	Café	Bus Stop	Sauna / Steam Room	Fish & Chips Shop	Electronics Store
1	Canbury	Pub	Gym / Fitness Center	Shop & Service	Park	Plaza	Café	Fish & Chips Shop
2	Coombe	Tea Room	Turkish Restaurant	French Restaurant	Discount Store	Electronics Store	Farmers Market	Fast Food Restaurant

3.5 Use K-means clustering to cluster the similar neighbourhoods (5 clusters).

Import the libraries and set the parameters.

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

# set number of clusters
kclusters = 5

kingston_grouped_clustering = kingston_grouped.drop('Neighbourhood', 1)

# run k-means clustering
kmeans = KMeans(n_clusters=kclusters, random_state=0).fit(kingston_grouped_

# check cluster labels generated for each row in the dataframe
kmeans.labels_[0:10]
```

Add clustering labels.

```
# add clustering labels
#neighbourhoods_venues_sorted.insert(0, 'Cluster Labels', kmeans.labels_)

kingston_merged = kingston_neigh

kingston_merged = kingston_merged.join(neighbourhoods_venues_sorted.set_inc

kingston_merged.head()
```

	Neighbourhood	Borough	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Cor
0	Berrylands	Kingston upon Thames	51.393781	-0.284802	3.0	Gym / Fitness Center	Park	Café	Bus
1	Canbury	Kingston upon Thames	51.417499	-0.305553	0.0	Pub	Gym / Fitness Center	Shop & Service	
2	Chessington	Kingston upon Thames	51.358336	-0.298622	NaN	NaN	NaN	NaN	

Drop the data with null values.

```
kingston_merged.dropna (inplace = True)
kingston_merged.head ()
```

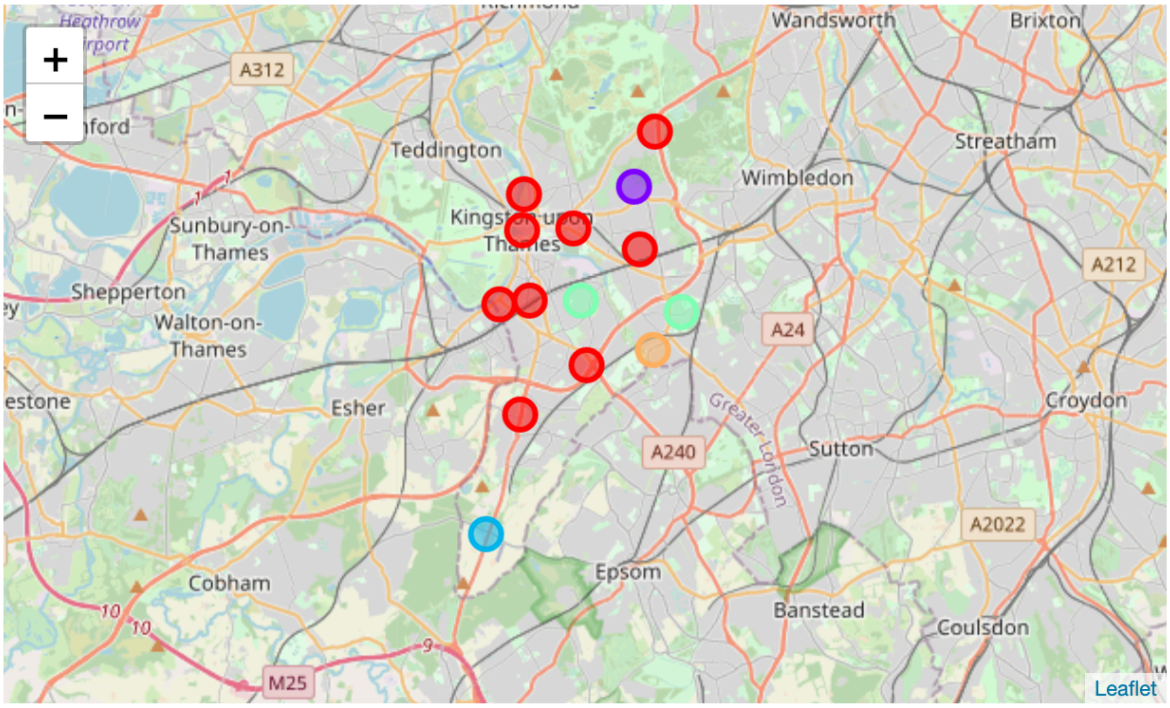
	Neighbourhood	Borough	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4t Cc
0	Berrylands	Kingston upon Thames	51.393781	-0.284802	3.0	Gym / Fitness Center	Park	Café	Bu
1	Canbury	Kingston upon Thames	51.417499	-0.305553	0.0	Pub	Gym / Fitness Center	Shop & Service	
3	Coombe	Kingston upon Thames	51.419450	-0.265398	1.0	Tea Room	Turkish Restaurant	French Restaurant	D

Change the cluster column labels.

```
# Change type of cluster labels
kingston_merged ['Cluster Labels'] = kingston_merged ['Cluster Labels'].ast
```

Visualise the clustering results on a map.

```
map_clusters = folium.Map(location=[latitude, longitude], zoom_start=11.5)

# set color scheme for the clusters
x = np.arange(kclusters)
ys = [i + x + (i*x)**2 for i in range(kclusters)]
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(kingston_merged['Latitude'], kingston_mer
    label = folium.Popup(str(poi) + ' Cluster ' + str(cluster), parse_html=
    folium.CircleMarker(
        [lat, lon],
        radius=8,
        popup=label,
        color=rainbow[cluster-1],
        fill=True,
        fill_color=rainbow[cluster-1],
        fill_opacity=0.5).add_to(map_clusters)

map_clusters
```



Examine the results.

```
kingston_merged[kingston_merged['Cluster Labels'] == 0]
```

	Neighbourhood	Borough	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	
1	Canbury	Kingston upon Thames	51.417499	-0.305553	0	Pub	Gym / Fitness Center	Shop & Service	
4	Hook	Kingston upon Thames	51.367898	-0.307145	0	Bakery	Supermarket	Indian Restaurant	
5	Kingston upon Thames	Kingston upon Thames	51.409627	-0.306262	0	Café	Pub	Burger Joint	R
6	Kingston Vale	Kingston upon Thames	51.431850	-0.258138	0	Grocery Store	Bar	Soccer Field	
9	New Malden	Kingston upon Thames	51.405335	-0.263407	0	Indian Restaurant	Korean Restaurant	Gastropub	R
10	Norbiton	Kingston upon Thames	51.409999	-0.287396	0	Platform	Italian Restaurant	Indian Restaurant	
12	Seething Wells	Kingston upon Thames	51.392642	-0.314366	0	Indian Restaurant	Pub	Coffee Shop	R
13	Surbiton	Kingston upon Thames	51.393756	-0.303310	0	Coffee Shop	Pub	Grocery Store	R
14	Tolworth	Kingston upon Thames	51.378876	-0.282860	0	Grocery Store	Pharmacy	Sandwich Place	

```
Second Cluster
```

```
kingston_merged[kingston_merged['Cluster Labels'] == 1]
```

	Neighbourhood	Borough	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th M Comm Ven
3	Coombe	Kingston upon Thames	51.41945	-0.265398	1	Tea Room	Turkish Restaurant	French Restaurant	Disco St

Third Cluster

```
kingston_merged[kingston_merged['Cluster Labels'] == 2]
```

	Neighbourhood	Borough	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th M Comm Ven
7	Malden Rushett	Kingston upon Thames	51.341052	-0.319076	2	Grocery Store	Pub	Garden Center	Restaur

Fourth Cluster

```
kingston_merged[kingston_merged['Cluster Labels'] == 3]
```

	Neighbourhood	Borough	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th M Comm Ven
0	Berrylands	Kingston upon Thames	51.393781	-0.284802	3	Gym / Fitness Center	Park	Café	Bus St
8	Motspur Park	Kingston upon Thames	51.390985	-0.248898	3	Gym	Park	Restaurant	Bus St

Fifth Cluster

```
: kingston_merged[kingston_merged['Cluster Labels'] == 4]
```

:

	Neighbourhood	Borough	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th M Comm Ver
11	Old Malden	Kingston upon Thames	51.382484	-0.25909	4	Train Station	Pub	Food	De Bode

5 Discussions

From the results of this project, it can be seen that the Kingston upon Thames is shown to be the safest borough in London for living. However, this result only takes in

consideration of total crimes occurred in that area without looking into the types of the crimes and the population of the area and therefore the results may not be strictly accurate. For example, the crimes of theft and burglary happens more frequently in the areas popular with tourists, such as Westminster, where many major tourist attractions are located, whereas the violence related crimes can be a major concern for international students when finding a home in a specific area.

Actually, the borough of Kingston upon Thames is not a very popular living areas for students as the location is somewhat remote to central London and only one university is located near this area. Most of the universities are located in central London, so further analysis of boroughs such as Camden, Kensington and Chelsea are desired.

6 Conclusions

The exploratory analysis shows that the borough of Kingston upon Thames is the safest borough in London as least number of total crimes was occurred Around this area. Also, for this borough, the most popular venues in each neighbourhood are listed for users to make decision from. Similar approach can be used to explore other boroughs and their neighbourhoods.