Applied Data Science Capstone Project
Battle of the Neighborhoods
Report
(By: Aditya Gupta)

Introduction

Problem Statement and The Target Audience:

The aim of this project is to select the safest borough in London based on the crime rate and to explore the neighborhoods of that borough to find the 10 most common venues in each neighborhood and finally cluster the neighborhoods using k-mean clustering.

This report will be of a great interest to the people who are looking to relocate to London. And in order to find a descent neighborhood or to hunt for an apartment, safety is considered as a major concern. The crime statistics will provide an insight into this issue and help us in solving the problem statement.

We will focus on the safest borough and explore its neighborhoods and the 10 most common venues in each neighborhood so that the best neighborhood suited to an individual's needs can be selected.

Data Acquisition & Cleaning

Data Acquisition:

The data required for this project is collected from three different sources. The first data source of the project uses the London Crime Data that shows the crime per borough in London. The dataset contains the following columns:

- **lsoa_code**: code for lower Super Output Area in Great London
- **borough**: Common name for London borough
- **major_category**: High level categorization of crime
- **minor_category**: Low level categorization of crime when major category
- **value**: monthly reported count of categorical crime in borough
- **year**: Year of reported counts, 2008-2016
- **month**: Month of reported counts, 1-12

The second source of data is scrapped from a wikipedia page that contains the list of London boroughs. This page contains additional information about boroughs, the following are the columns of dataset:

- **Borough**: The names of the 33 London Boroughs
- **Inner**: Categorizing the borough as an Inner London Borough or an Outer London Borough.
- **Status**: Categorizing the borough as Royal, City or other borough
- **Local Authority**: The local authority assigned to the borough
- **Political Conrtrol**: The political party that controls the borough
- **Headquaters**: Headquaters of the borough
- **Area(sq mi)**: Area of the borough in square miles

- **Population(2013 est)[1]**: The population in the borough recorder during the year 2013
- **Co-ordinates**: The latitude and longitudes of the borough
- **Nr. in map**: The number assigned to each borough to represent visually on a map

The third data source is the list of Neighborhoods in the Royal Borough of Kingston upon Thames as found on the wikipedia page. This dataset is scrapped throught he list of neighborhood available on the site with the following columns:

- **Neighborhood**: Name of the neighborhood in the Borough
- **Borough**: Name of the borough
- **Latitude**: Latitude of the borough
- **Longitude**: Longitude of the borough

Data Cleaning and Scrapping

Preprocessing a real world data set from Kaggle showing the London Crimes from 2008 to 2016:

Dataset URL: [https://www.kaggle.com/jboysen/london-crime]

- Reading the dataset using Pandas:

```
df = pd.read_csv('/content/drive/My Drive/Colab Notebooks/Dataset/London Crime/london_crime_by_lsoa.csv')
df.head()
```

	lsoa_code	borough	major_category	minor_category	value	year	month
0	E01001116	Croydon	Burglary	Burglary in Other Buildings	0.0	2016.0	11.0
1	E01001646	Greenwich	Violence Against the Person	Other violence	0.0	2016.0	11.0
2	E01000677	Bromley	Violence Against the Person	Other violence	0.0	2015.0	5.0
3	E01003774	Redbridge	Burglary	Burglary in Other Buildings	0.0	2016.0	3.0
4	E01004563	Wandsworth	Robbery	Personal Property	0.0	2008.0	6.0

- Accessing the most recent crime rates in dataset(2016)

```
# Taking only the most recent year (2016) and dropping the rest
df.drop(df.index[df['year'] != 2016], inplace = True)

# Removing all the entires where crime values are null
df = df[df.value != 0]

# Reset the index and dropping the previous index
df = df.reset_index(drop=True)
```

```
[ ] # Shape of the data frame
df.shape
```

```
(20631, 7)
```

```python
# View the top of the dataset
df.head()
```

	lsoa_code	borough	major_category	minor_category	value	year	month
0	E01004177	Sutton	Theft and Handling	Theft/Taking of Pedal Cycle	1.0	2016.0	8.0
1	E01000733	Bromley	Criminal Damage	Criminal Damage To Motor Vehicle	1.0	2016.0	4.0
2	E01003989	Southwark	Theft and Handling	Theft From Shops	4.0	2016.0	8.0
3	E01002276	Havering	Burglary	Burglary in a Dwelling	1.0	2016.0	8.0
4	E01003674	Redbridge	Drugs	Possession Of Drugs	2.0	2016.0	11.0

- Renaming the Columns:

```python
df.columns = ['LSOA_Code', 'Borough','Major_Category','Minor_Category','No_of_Crimes','Year','Month']
df.head()
```

	LSOA_Code	Borough	Major_Category	Minor_Category	No_of_Crimes	Year	Month
0	E01004177	Sutton	Theft and Handling	Theft/Taking of Pedal Cycle	1.0	2016.0	8.0
1	E01000733	Bromley	Criminal Damage	Criminal Damage To Motor Vehicle	1.0	2016.0	4.0
2	E01003989	Southwark	Theft and Handling	Theft From Shops	4.0	2016.0	8.0
3	E01002276	Havering	Burglary	Burglary in a Dwelling	1.0	2016.0	8.0
4	E01003674	Redbridge	Drugs	Possession Of Drugs	2.0	2016.0	11.0

- Total number of crimes in each Borough:

```
[ ]  df['Borough'].value_counts()
```

```
Lambeth                    954
Southwark                  883
Croydon                    857
Newham                     828
Ealing                     811
Tower Hamlets              802
Brent                      786
Hackney                    781
Barnet                     753
Lewisham                   737
Haringey                   736
Enfield                    710
Wandsworth                 700
Greenwich                  687
Camden                     682
Westminster                666
Hillingdon                 666
Waltham Forest             655
Islington                  649
Hounslow                   631
Redbridge                  608
Bromley                    604
Hammersmith and Fulham     544
Barking and Dagenham       531
```

- Pivoting the table to view the no. of crimes for each major category in each Borough:

```
[ ]  London_crime = pd.pivot_table(df,values=['No_of_Crimes'],
                                   index=['Borough'],
                                   columns=['Major_Category'],
                                   aggfunc=np.sum,fill_value=0)
     London_crime.head()
```

	No_of_Crimes						
Major_Category	Burglary	Criminal Damage	Drugs	Other Notifiable Offences	Robbery	Theft and Handling	Violence Against the Person
Borough							
Barking and Dagenham	92	107	47	25	27	264	335
Barnet	191	115	41	29	26	500	364
Bexley	51	99	42	11	7	249	231
Brent	153	112	87	30	40	457	467
Bromley	110	104	35	24	17	324	344

```
[ ] # Reset the index
    London_crime.reset_index(inplace = True)
```

```
[ ] # Total crimes per Borough
    London_crime['Total'] = London_crime.sum(axis=1)
    London_crime.head(30)
```

1	Barnet	191	115	41	29	26	500	364	1266
2	Bexley	51	99	42	11	7	249	231	690
3	Brent	153	112	87	30	40	457	467	1346
4	Bromley	110	104	35	24	17	324	344	958
5	Camden	133	119	106	21	51	683	444	1557
6	City of London	0	0	1	0	2	4	1	8
7	Croydon	112	186	73	39	54	498	529	1491
8	Ealing	120	148	64	29	49	525	496	1431
9	Enfield	123	88	69	26	38	389	422	1155
10	Greenwich	111	129	55	20	32	394	466	1207

- Removing the multi index so that it will be easier to merge:

```
[ ] London_crime.columns = London_crime.columns.map(''.join)
    London_crime.head()
```

	Borough	No_of_CrimesBurglary	No_of_CrimesCriminal Damage	No_of_CrimesDrugs	No_of_CrimesOther Notifiable Offences	No_of_CrimesRobbery	Nc
0	Barking and Dagenham	92	107	47	25	27	
1	Barnet	191	115	41	29	26	
2	Bexley	51	99	42	11	7	
3	Brent	153	112	87	30	40	
4	Bromley	110	104	35	24	17	

- Renaming the columns:

```
[ ] London_crime.columns = ['Borough','Burglary', 'Criminal Damage','Drugs','Other Notifiable Offences',
                            'Robbery','Theft and Handling','Violence Against the Person','Total']
    London_crime.head()
```

	Borough	Burglary	Criminal Damage	Drugs	Other Notifiable Offences	Robbery	Theft and Handling	Violence Against the Person	Total
0	Barking and Dagenham	92	107	47	25	27	264	335	897
1	Barnet	191	115	41	29	26	500	364	1266
2	Bexley	51	99	42	11	7	249	231	690
3	Brent	153	112	87	30	40	457	467	1346
4	Bromley	110	104	35	24	17	324	344	958

Scraping additional information of the different Boroughs in London from a Wikipedia page

Dataset URL: https://en.wikipedia.org/wiki/List_of_London_boroughs

- Using Beautiful soup to scrap the latitude and longitiude of the boroughs in London:

```
[ ] # getting data from internet
    wikipedia_link='https://en.wikipedia.org/wiki/List_of_London_boroughs'
    raw_wikipedia_page= requests.get(wikipedia_link).text

    # using beautiful soup to parse the HTML/XML codes.
    soup = BeautifulSoup(raw_wikipedia_page,'xml')
    print(soup.prettify())
```

```
     Privacy policy
    </a>
   </li>
   <li id="footer-places-about">
    <a href="/wiki/Wikipedia:About" title="Wikipedia:About">
     About Wikipedia
    </a>
   </li>
   <li id="footer-places-disclaimer">
    <a href="/wiki/Wikipedia:General_disclaimer" title="Wikipedia:General disclaimer">
     Disclaimers
    </a>
   </li>
   <li id="footer-places-contact">
    <a href="//en.wikipedia.org/wiki/Wikipedia:Contact_us">
```

```
[ ] # extracting the raw table inside that webpage
    table = soup.find_all('table', {'class':'wikitable sortable'})
    print(table)
```

```
<td><a href="/wiki/Conservative_Party_(UK)" title="Conservative Party (UK)">Conservative</a>
</td>
<td><a href="/wiki/Westminster_City_Hall" title="Westminster City Hall">Westminster City Hall</a>, 64 Victoria Stre
</td>
<td>8.29
</td>
<td>226,841
</td>
<td><span class="plainlinks nourlexpansion"><a class="external text" href="//geohack.toolforge.org/geohack.php?page
</td>
<td>2
</td></tr></tbody></table>, <table class="wikitable sortable" style="font-size:95%" width="100%">
<tbody><tr>
<th width="100px">Borough
</th>
<th>Inner
</th>
```

- Converting the table into a data frame

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

	Borough	Inner	Status	Local authority	Political control	Headquarters	Area (sq mi)	Population (2013 est) [1]	Co-ordinates	N m
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	

- The second table on the site contains the addition Borough i.e. City of London:

```
[ ] # Read in the second table
    London_table1 = pd.read_html(str(table[1]), index_col=None, header=0)[0]

    # Rename the columns to match the previous table to append the tables.

    London_table1.columns = ['Borough','Inner','Status','Local authority','Political control',
                             'Headquarters','Area (sq mi)','Population (2013 est)[1]','Co-ordinates','Nr. in map']

    # View the table
    London_table1
```

	Borough	Inner	Status	Local authority	Political control	Headquarters	Area (sq mi)	Population (2013 est) [1]	Co-ordinates	N m
0	City of London	([note 5]	Sui generis;City;Ceremonial county	Corporation of London;Inner Temple;Middle Temple	?	Guildhall	1.12	7000	51°30′56″N 0°05′32″W / 51.5155°N 0.0922°W	

- Append the data frame together:

```
London_table = London_table.append(London_table1, ignore_index = True)
London_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

- Removing Unnecessary string in the Data set:

```
[ ] London_table = London_table.replace('note 1','', regex=True)
    London_table = London_table.replace('note 2','', regex=True)
    London_table = London_table.replace('note 3','', regex=True)
    London_table = London_table.replace('note 4','', regex=True)
    London_table = London_table.replace('note 5','', regex=True)

    # View the top of the data set
    London_table.head()
```

	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

- These 3 Boroughs don't match because of the unnecessary symbols present "[]"
- Find the index of the Boroughs that didn't match

Find the index of the Boroughs that didn't match

```
: index of first borough is",London_table.index[London_table['Borough'] == 'Barking and Dagenham []'].tolist())
: index of second borough is",London_table.index[London_table['Borough'] == 'Greenwich []'].tolist())
: index of third borough is",London_table.index[London_table['Borough'] == 'Hammersmith and Fulham []'].tolist())
```

```
The index of first borough is [0]
The index of second borough is [9]
The index of third borough is [11]
```

Changing the Borough names to match the other data frame

```
[ ] London_table.iloc[0,0] = 'Barking and Dagenham'
    London_table.iloc[9,0] = 'Greenwich'
    London_table.iloc[11,0] = 'Hammersmith and Fulham'
```

- Changing the Borough names to match the other data frame
- Check if the Borough names in both data sets match

```
[ ] London_table.iloc[0,0] = 'Barking and Dagenham'
    London_table.iloc[9,0] = 'Greenwich'
    London_table.iloc[11,0] = 'Hammersmith and Fulham'
```

Check if the Borough names in both data sets match

```
[ ] set(df.Borough) - set(London_table.Borough)
```

```
set()
```

- The Borough names in both data frames match We can combine both the data frames together:

```
Ld_crime = pd.merge(London_crime, London_table, on='Borough')
Ld_crime.head(10)
```

0	Barking and Dagenham	92	107	47	25	27	264	335	897	NaN	NaN
1	Barnet	191	115	41	29	26	500	364	1266	NaN	NaN
2	Bexley	51	99	42	11	7	249	231	690	NaN	NaN
3	Brent	153	112	87	30	40	457	467	1346	NaN	NaN

- Rearranging the Columns:

```
columnsTitles = ['Borough','Local authority','Political control','Headquarters',
                 'Area (sq mi)','Population (2013 est)[1]',
                 'Inner','Status',
                 'Burglary','Criminal Damage','Drugs','Other Notifiable Offences',
                 'Robbery','Theft and Handling','Violence Against the Person','Total','Co-ordinates']

Ld_crime = Ld_crime.reindex(columns=columnsTitles)

Ld_crime = Ld_crime[['Borough','Local authority','Political control','Headquarters',
                 'Area (sq mi)','Population (2013 est)[1]','Co-ordinates',
                 'Burglary','Criminal Damage','Drugs','Other Notifiable Offences',
                 'Robbery','Theft and Handling','Violence Against the Person','Total']]

Ld_crime.head()
```

	Borough	Local authority	Political control	Headquarters	Area (sq mi)	Population (2013 est)[1]	Co-ordinates	Burglary	Criminal Damage	Drugs	Other Notifiable Offences
0	Barking and Dagenham	Barking and Dagenham London Borough	Labour	Town Hall, 1 Town Square	13.93	194352	51°33′39″N 0°09′21″E / 51.5607°N 0.1557°E	92	107	47	25

Methodology

The methodology in this project consists of two parts:

- **Exploratory Data Analysis**: Visualise the crime rates in the London boroughs to idenity the safest borough and extract the neighborhoods in that borough to find the 10 most common venues in each neighborhood.
- **Modelling**: To help people find similar neighborhoods in the safest borough we will be clustering similar neighborhoods using K - means clustering which is a form of unsupervised machine learning algorithm that clusters data based on predefined cluster size. We will use a cluster size of 5 for this project that will cluster the 15 neighborhoods into 5 clusters. The reason to conduct a K- means clustering is to cluster neighborhoods with similar venues together so that people can shortlist the area of their interests based on the venues/amenities around each neighborhood.

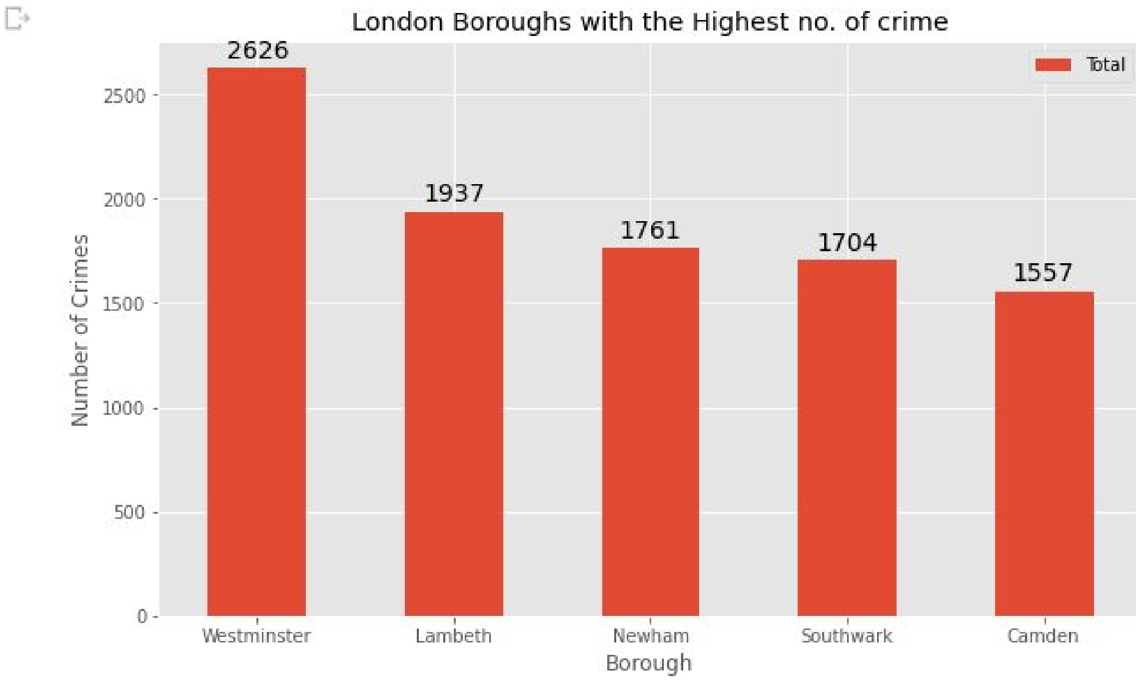
Exploratory Data Analysis

Descriptive statistics of the data:

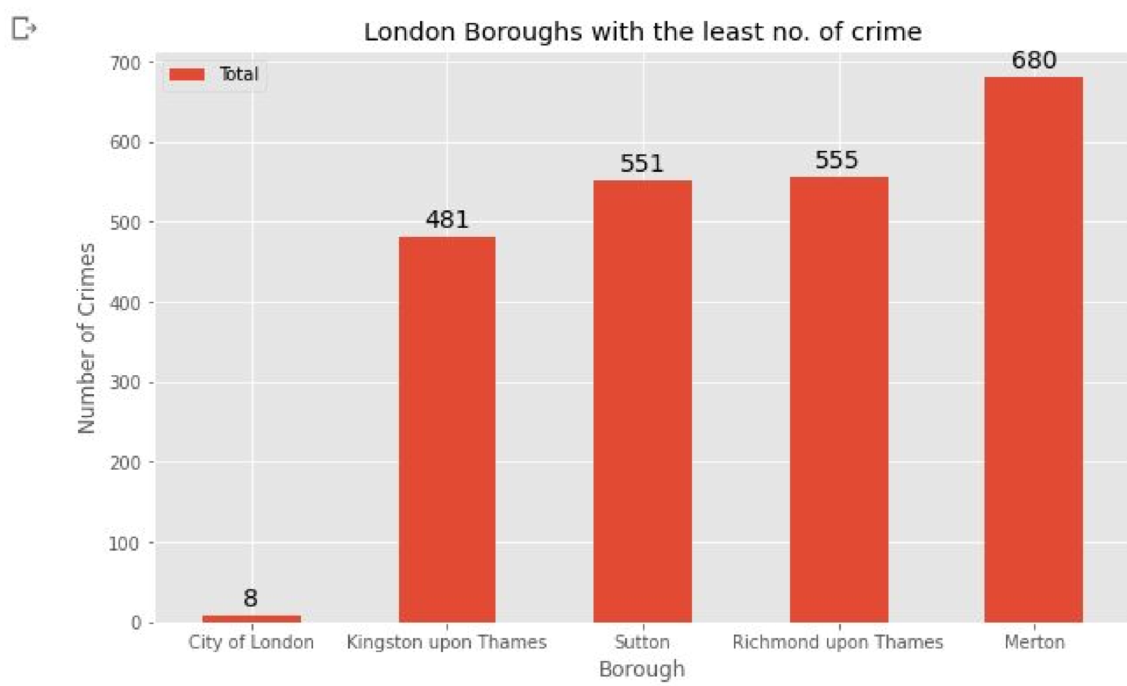
```
[ ] London_crime.describe()
```

	Burglary	Criminal Damage	Drugs	Other Notifiable Offences	Robbery	Theft and Handling	Violence Against the Person	Total
count	33.000000	33.000000	33.000000	33.000000	33.000000	33.000000	33.000000	33.000000
mean	110.121212	104.242424	65.212121	24.757576	33.333333	470.151515	373.727273	1181.545455
std	41.052373	35.411007	38.764318	11.797855	21.548879	251.751753	139.767278	485.989139
min	0.000000	0.000000	1.000000	0.000000	2.000000	4.000000	1.000000	8.000000
25%	82.000000	77.000000	41.000000	18.000000	14.000000	310.000000	309.000000	914.000000
50%	112.000000	107.000000	59.000000	25.000000	27.000000	457.000000	395.000000	1207.000000
75%	133.000000	124.000000	104.000000	30.000000	49.000000	566.000000	467.000000	1457.000000
max	191.000000	186.000000	169.000000	60.000000	73.000000	1446.000000	654.000000	2626.000000

Visualize the five boroughs with the highest number of crimes:



Visualize the five boroughs with the least number of crimes:

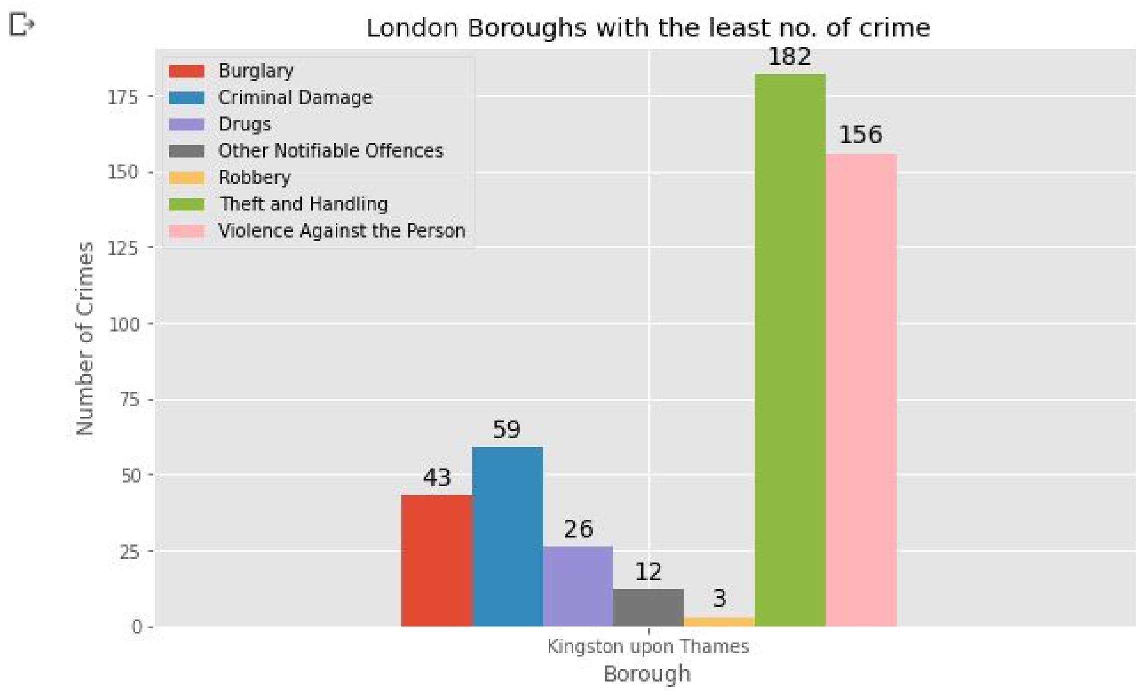


The borough City of London has the lowest no. of crimes recorded for the year 2016, Looking into the details of the borough:

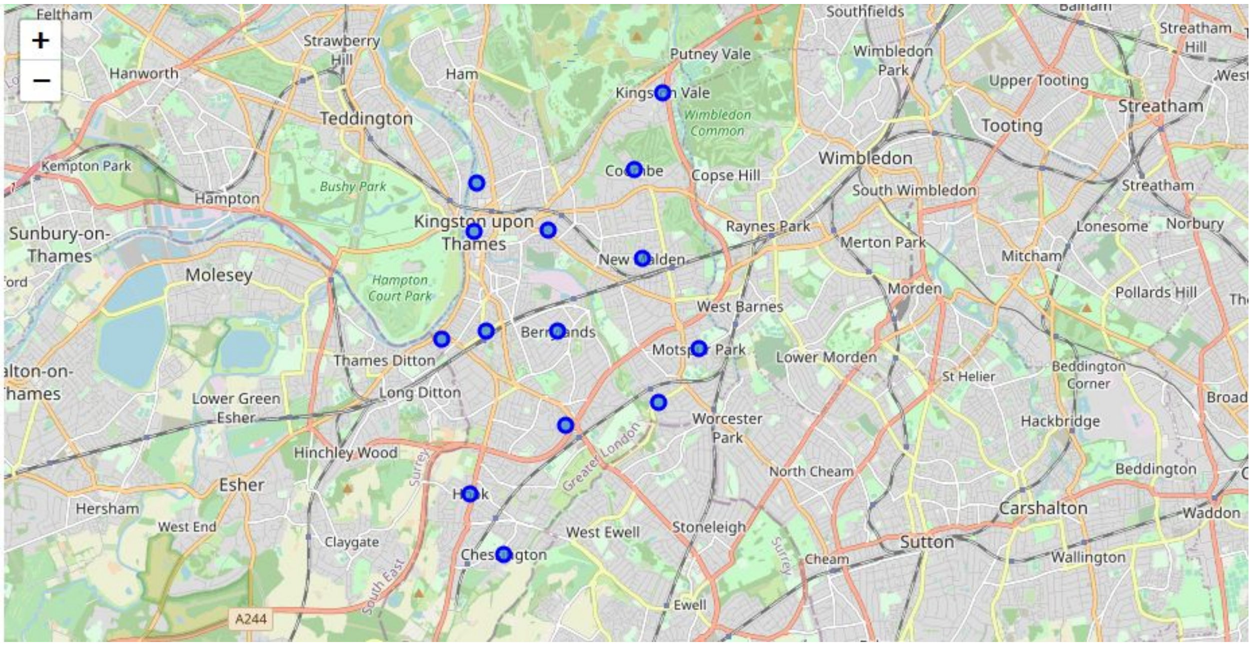
```
[ ] df_col = df_bot5[df_bot5['Borough'] == 'City of London']
    df_col = df_col[['Borough','Total','Area (sq mi)','Population (2013 est)[1]']]
    df_col
```

	Borough	Total	Area (sq mi)	Population (2013 est)[1]
6	City of London	8	1.12	7000

Visualizing different types of crimes in the borough 'Kingston upon Thames':



Visualize the Neighborhood of Kingston upon Thames Borough:



Modelling

- Finding all the venues within a 500 meter radius of each neighborhood.
- Perform one hot ecoding on the venues data.
- Grouping the venues by the neighborhood and calculating their mean.
- Performing a K-means clustering (Defining K = 5)

Extracting the venues from each Neighborhood:

```
[ ]  kut_venues = getNearbyVenues(names=kut_neig['Neighborhood'],
                                   latitudes=kut_neig['Latitude'],
                                   longitudes=kut_neig['Longitude']
                                  )
```

```
Berrylands
Canbury
Chessington
Coombe
Hook
Kingston upon Thames
Kingston Vale
Malden Rushett
Motspur Park
New Malden
Norbiton
Old Malden
Seething Wells
Surbiton
Tolworth
```

Venue Details of Each Neighborhood:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Berrylands	51.393781	-0.284802	Surbiton Racket & Fitness Club	51.392676	-0.290224	Gym / Fitness Center
1	Berrylands	51.393781	-0.284802	Alexandra Park	51.394230	-0.281206	Park
2	Berrylands	51.393781	-0.284802	K2 Bus Stop	51.392302	-0.281534	Bus Stop
3	Berrylands	51.393781	-0.284802	Cafe Rosa	51.390175	-0.282490	Café
4	Canbury	51.417499	-0.305553	The Boater's Inn	51.418546	-0.305915	Pub

Result

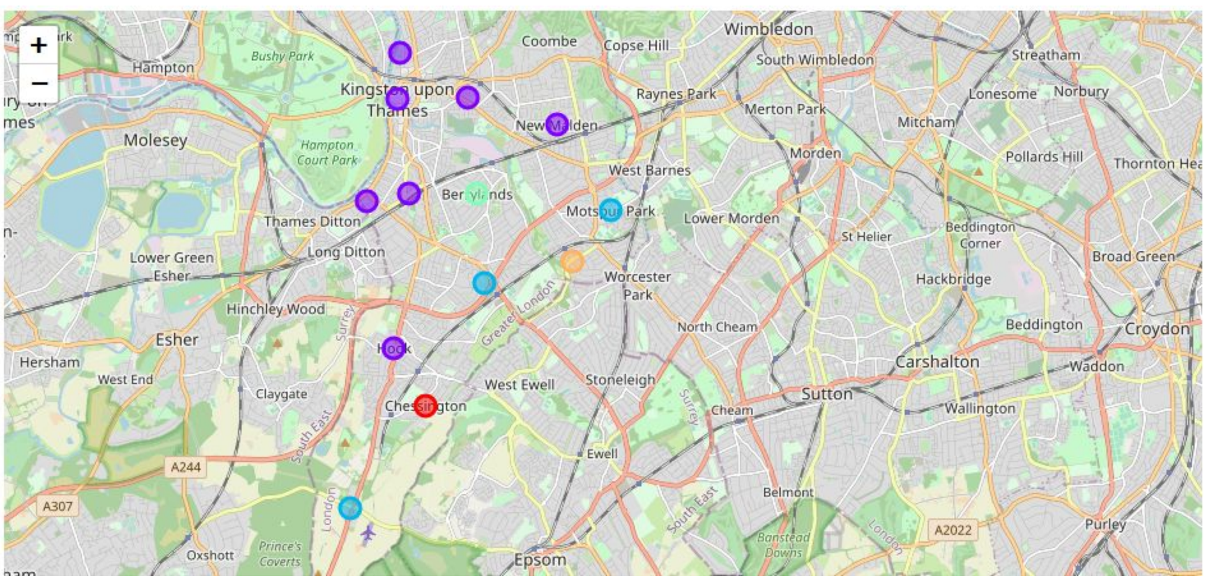
Visualize the clusters:

```
[ ] # create 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(kut_merged['Latitude'], kut_merged['Longitude'], kut_merged['Neighborhood'], kut_
        label = folium.Popup(str(poi) + ' Cluster ' + str(cluster), parse_html=True)
        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
```



Discussion

The project aims to help people who want to relocate to the safest borough in London, expats can chose the neighborhoods to which they want to relocate based on the most common venues in it. For example if a person is looking for a neighborhood with good connectivity and public transportation we can see that Clusters 3 and 4 have Train stations and Bus stops as the most common venues. If a person is looking for a neighborhood with stores and restaurants in a close proximity then the neighborhoods in the first cluster is suitable. For a family I feel that the neighborhoods in Cluster 4 are more suitable dues to the common venues in that cluster, these neighborhoods have common venues such as Parks, Gym/Fitness centers, Bus Stops, Restaurants, Electronics Stores and Soccer fields which is ideal for a family.

Conclusion

It is always helpful to make use of technology to stay one step ahead i.e. finding out more about places before moving into a neighborhood. We have just taken safety as a primary concern to shortlist the borough of London. The future of this project includes taking other factors such as cost of living in the areas into consideration to shortlist the borough based on safety and a predefined budget.
