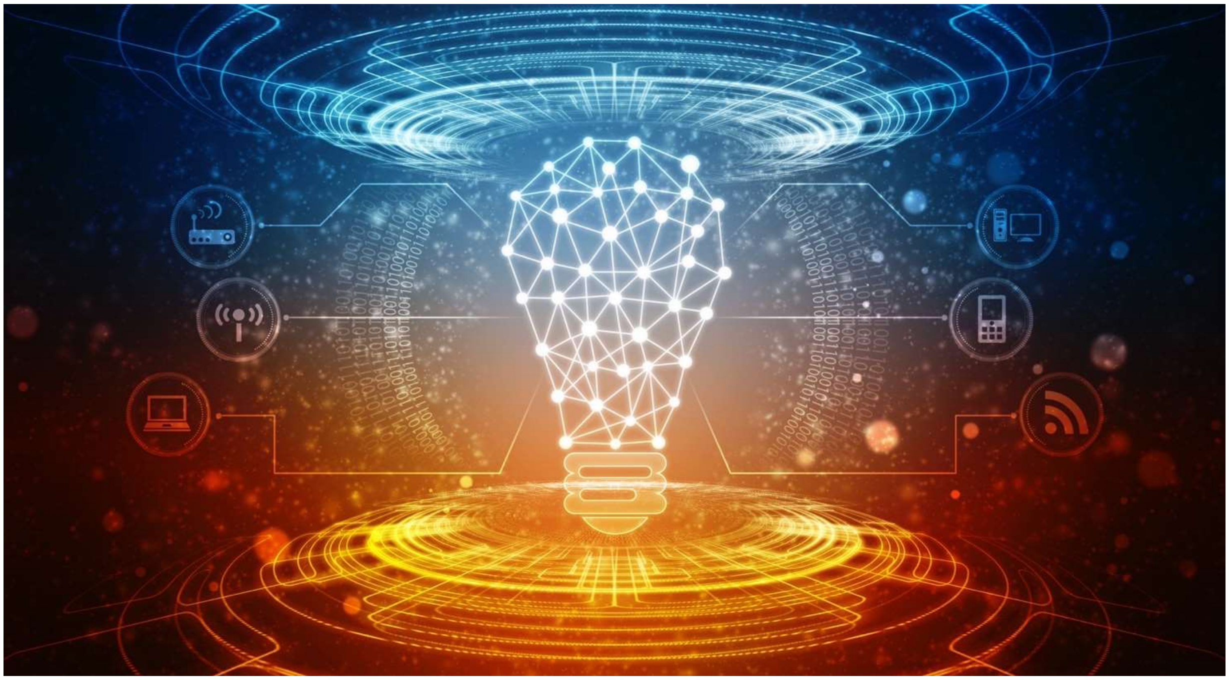
Capstone Project
The Battle of Neighborhoods
Final Report



Created Mar 04, 2021

Last Updated Mar 04, 2021

Version 0.1 English

Contents

Version	Name	Date	Description
0.1	Eddie Lau	20210304	Initial Version

1. Introduction

This document is the final report for Capstone Project - The Battle of Neighborhoods, which is final project for IBM Data Science Professional Certificate.

This report will cover the following topics:

1. Introduction the business problem and who would be interested in this project.
2. Data describe the data that will be used to solve the problem and the source of the data.
3. Methodology section to describe exploratory data analysis that you did, any inferential statistical testing that you performed, if any, and what machine learnings were used and why.
4. Results section to discuss the results.
5. Discussion section where you discuss any observations noted and any recommendations can make based on the results.
6. Conclusion section.

2. Business Requirements

2.1 Background

As the COVID-19 pandemic in England since 2020, the government trying the best the control the situation, sadly it still costing lives.

After the existence of the new variant found around Dec 2020, a new public health guidance and were expected to impose transit restrictions. By mid-December around two-thirds of the cases reported in London were the new variant. On 19 December it was announced that a new "tier four" measure would be applied to London, Kent, Essex, Bedfordshire, Buckinghamshire and Hertfordshire, and Christmas season relaxation would be limited to only Christmas Day.

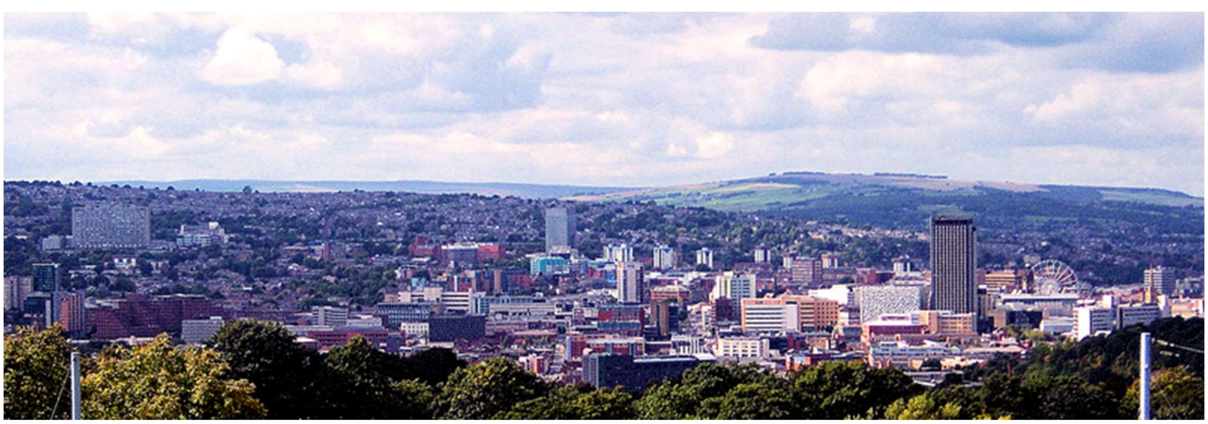
These attempts at controlling the second wave had limited success: the total number of hospital admissions rose again during December to more than 58,600, and deaths in hospital approached 10,600. Although almost 39,000 patients were discharged there were still more than 22,700 people in hospital on 31 December.

As the success of massive jab of vaccine during early Jan 2021, the government has released a the latest roadmap out of lockdown, since 8 March, people in England will see restrictions start to lift and the government's four-step roadmap offer a route back to a more normal life.

During this two year, many families, job and business, social relationships are destroyed and under siege.

As my long-term plan with wife, we are trying to move to Sheffield area for more quiet lives and plan for retires. For me, I studied and stayed in Sheffield in long time ago, I enjoyed the moment there.

Sheffield is a city and metropolitan borough in South Yorkshire, England, which is about the middle of England. It is surrounded by Manchester, Leeds, Derby, Birmingham. Unlike mega metropolitan like Manchester and Leeds, large part of the city connects with the Peak District in Derbyshire (South of Sheffield), so make she still a countryside style, even it is one of the large city in England.



2.2 Target Audience:

We would like to make use of latest technologies to seek a appropriate area for new home, in scientistic way, even we cannot visit there during lockdown.

2.3 Requirements

The following criterias affecting my choice:

1. It should be convenience to get foods and daily essentials.
2. Low crime rate.
3. High employment rate or moderate and above income.
4. With certain populations and avoid too countryside area.

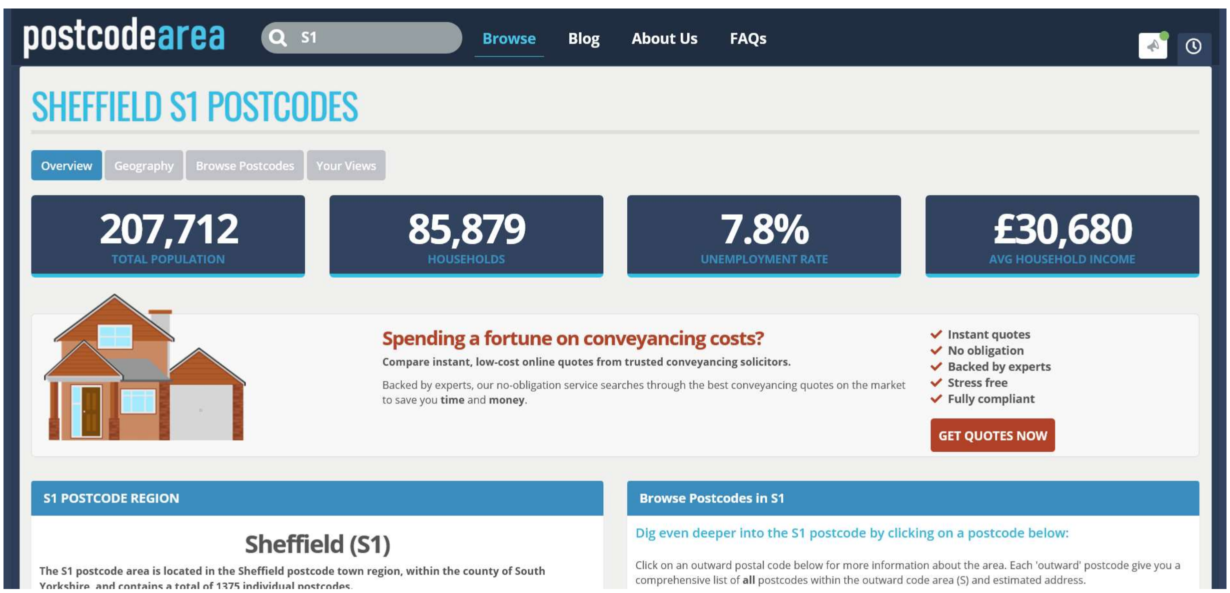
2.4 Success Criteria:

The success criteria of the project will be a good recommendation or living area.

3. Data

The following data will be used in the works:

1. Geo information from geolocator API
2. Sheffield postal area, https://www.postcodearea.co.uk/postaltowns/sheffield/



The postcode area provides information about Cities, Populations and Income.

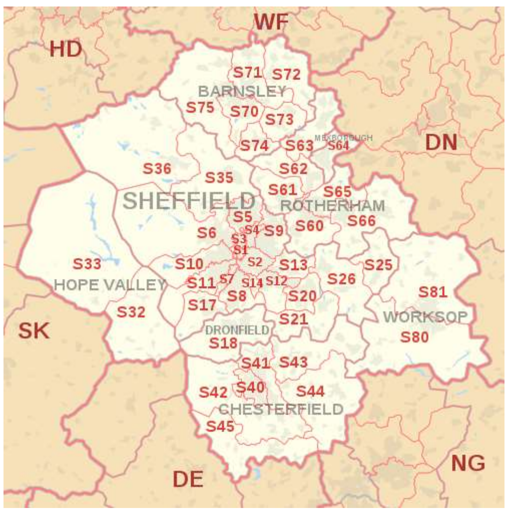
It forms the csv named: population.csv and sample as below.

```
City,Postal Code,Population,Household,Unemployment,Household Income
Sheffield ,S1,207712,85879,0.078,30680
Sheffield ,S2,143891,60186,0.105,27560
Sheffield ,S3,98762,41521,0.104,41080
Sheffield ,S4,189554,83361,0.18,27040
Sheffield ,S5,60467,24840,0.122,31200
Sheffield ,S6,301680,129156,0.057,32240
```

3. Sheffield postal code map, https://en.wikipedia.org/wiki/S_postcode_area

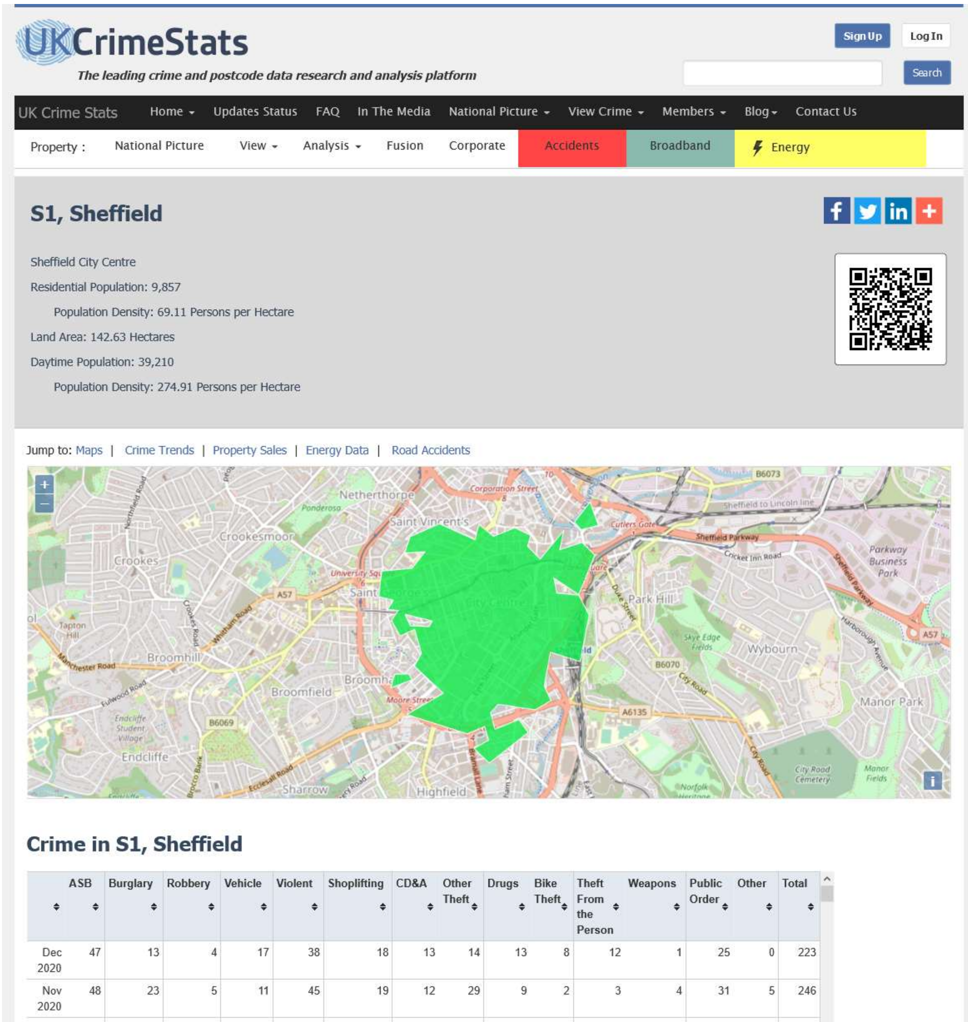
As there is no open data on surround area and towns, manual rendering data is needed. Based on the data, a Neighborhood.csv is formed and sample as below.

```
Postal Code,Area,Neighborhood
S1,Orchard Square,"S2, S3, S4, S6, S10, S11"
S2,Arbourthorne,"S14, S12, S13, S8, S7, S1, S9"
S3,Burngreave,"S1, S4, S5, S6, S10"
S4,Grimesthorpe,"S5, S3, S9"
S5,Firth Park,"S6, S35, S61, S4, S9"
```



4. UK Crime stats, https://www.ukcrimestats.com/

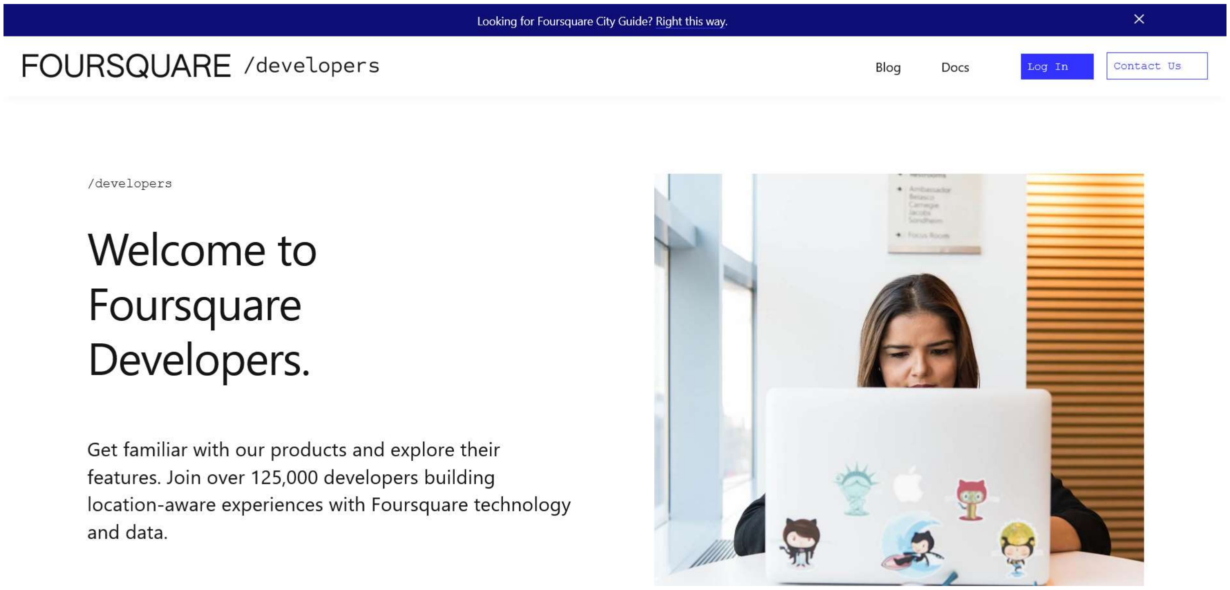
Abstract of the crime data is used form the Crime.csv and sample as below.




```
Postal Code,Crime Dec2020
S1,223
S2,585
S3,221
S4,222
```

5. Foursquare API

This is the API introduced in previous week for collecting the venue.



6. geolocator API

4. Methodologies

Up to now, we have three data sources, so we cut the process into three sections.

4.1 Part 1 Location Information

The first goal is trying to retrieve:

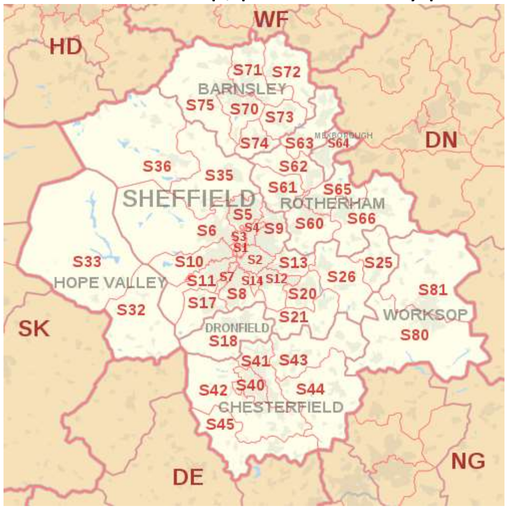
1. All area and their relationships by postal code in all Sheffield area.
2. Find all daily essentials in above area.

Implementaions

1. Visit the Sheffield postal code map, https://en.wikipedia.org/wiki/S_postcode_area. Render the list of postal code, area and their neighborhodd. Since it is not in a structured (json) data format, we need to render it into a csv named Neighborhood.csv, sample as below:

```
Postal Code,Area,Neighborhood
S1,Orchard Square,"S2, S3, S4, S6, S10, S11"
S2,Arbourthorne,"S14, S12, S13, S8, S7, S1, S9"
S3,Burngreave,"S1, S4, S5, S6, S10"
S4,Grimesthorpe,"S5, S3, S9"
S5,Firth Park,"S6, S35, S61, S4, S9"
```

2. Based on the Map, put the nearby postal code into the csv as well.



3. After loading it in Jupyter, transverse the data into appropriate format.

```
[9]: #Tranverse separated neighborhood into multiple column and map with area
df.rename(columns={"Postal Code": "postal_code", "Area": "area", "Neighborhood": "neighborhood"}, inplace=True)
df['neighborhood'] = df['neighborhood'].str.split(',')
#df=df.reset_index(['postal_code'])
df = df.explode('neighborhood')
df["neighborhood"]  = df["neighborhood"].str.strip()
#print(df['neighborhood'])

pd.options.mode.chained_assignment = None
df_neighborhood = pd.merge(df, df_area, left_on = 'neighborhood', right_on = 'postal_code',  how='left')
df_neighborhood.drop(['neighborhood'], axis=1, inplace=True)
df_neighborhood.rename(columns={"postal_code_x": "postal_code", "name": "neighborhood", 'postal_code_y' : "neighborhood_postal_code"}, inplace=True)

#df_neighborhood=df_neighborhood.reset_index([0, 'area'])
df_neighborhood.head()
```

[9]:

	postal_code	area	neighborhood_postal_code	neighborhood
0	S1	Orchard Square	S2	Arbourthorne
1	S1	Orchard Square	S3	Burngreave
2	S1	Orchard Square	S4	Grimesthorpe
3	S1	Orchard Square	S6	Hillsborough
4	S1	Orchard Square	S10	Fulwood

4. Use the python geocoder library, find Latitude and Longitude, and add data to list of area. Speical care is needed as the place name will appear in other cities.

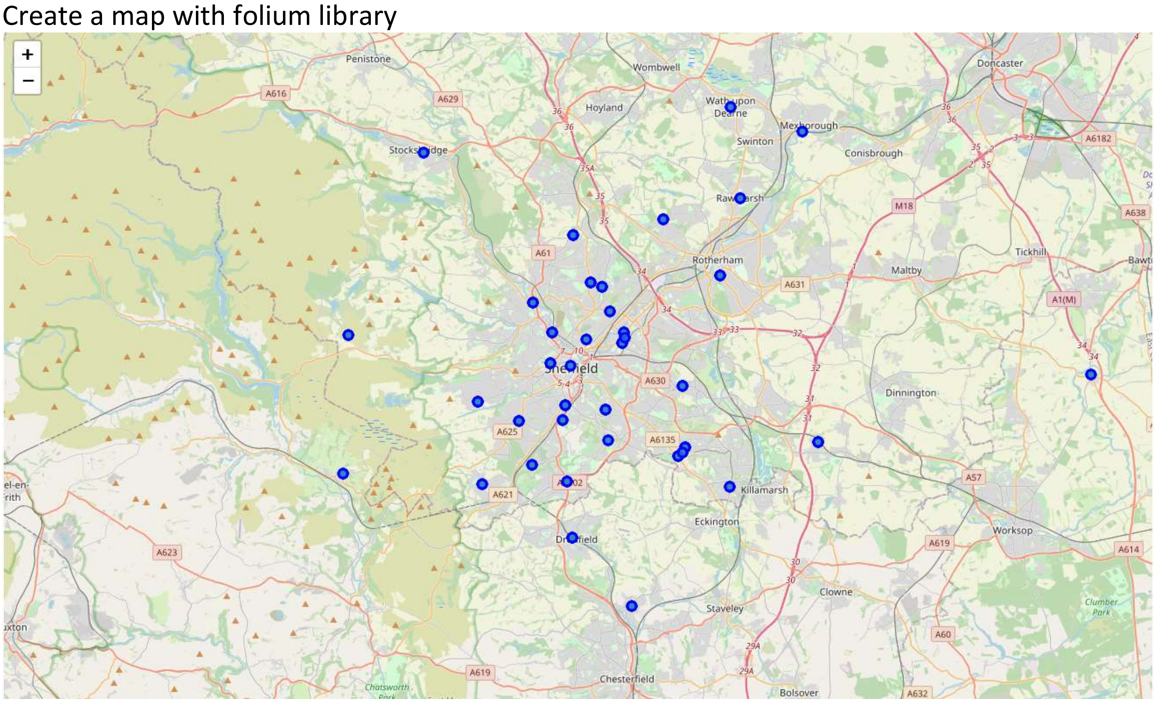
```
[12]: #Collect coordinate and merge back to df
latitude = []
longitude = []

geolocator = Nominatim(user_agent="ny_explorer")
for index, row in df_area_xy.iterrows():
    location = geolocator.geocode(row['area'] + ' ' + row['postal_code'] +', Sheffield, United Kingdom') #More accurate
    if location is None:
        location = geolocator.geocode(row['area']+ ' ' + row['postal_code'] + ', United Kingdom') #Handle exception case
    #print('The geographical coordinate of {}.'.format(row['neighbourhood']))
    if location is not None:
        latitude.append(location.latitude)
        longitude.append(location.longitude)
        print('The geograpical coordinate of {} are {}, {}.'.format(location, location.latitude, location.longitude))
    else:
        latitude.append(np.nan)
        longitude.append(np.nan)
df_area_xy['latitude'] = latitude
df_area_xy['longitude'] = longitude
df_area_xy.head
```

```
The geograpical coordinate of Orchard Square, City Centre, Sheffield, Yorkshire and the Humber, England, S1 2FB, United Kingdom are 53.3816738, -1.4705044.
The geograpical coordinate of Arbourthorne, Sheffield, Yorkshire and the Humber, England, S2 2FL, United Kingdom are 53.3613888, -1.4431226.
The geograpical coordinate of Burngreave, Sheffield, Yorkshire and the Humber, England, S4 7HQ, United Kingdom are 53.3939876, -1.45832.
The geograpical coordinate of Grimesthorpe, Sheffield, Yorkshire and the Humber, England, S4 8EZ, United Kingdom are 53.4070424, -1.4397035.
The geograpical coordinate of Firth Park, Sheffield, Yorkshire and the Humber, England, S5 6HB, United Kingdom are 53.4184353, -1.4462956.
The geograpical coordinate of Hillsborough Stadium, Penistone Road North, Middlewood, Wadsley Bridge, Sheffield, Yorkshire and the Humber, England, S6 1SW, United Kingdom are 53.4113729, -1.5006348419862217.
The geograpical coordinate of Beauchief, Sheffield, Yorkshire and the Humber, England, S8 0ER, United Kingdom are 53.3353094, -1.5008867.
The geograpical coordinate of The Norton, 337, Norton Lane, Meadowhead, Sheffield, Yorkshire and the Humber, England, S8 7UP, United Kingdom are 53.3274213, -1.4734325.
The geograpical coordinate of Attercliffe, Chippingham Street, Stadia Technology Park, Attercliffe, Sheffield, Yorkshire and the Humber, England, S9 3SP, United Kingdom are 53.3922699, -1.4302672.
The geograpical coordinate of Fulwood, Sheffield, Yorkshire and the Humber, England, S10 3RN, United Kingdom are 53.3650134, -1.5432431.
The geograpical coordinate of Ecclesall, Sheffield, Yorkshire and the Humber, England, S11 9GE, United Kingdom are 53.3557738, -1.5110371.
The geograpical coordinate of Hackenthorpe, Sheffield Road, Birley, Sheffield, Yorkshire and the Humber, England, S20 6RG, United Kingdom are 53.3436065, -1.3810491.
The geograpical coordinate of Handsworth, Sheffield, Yorkshire and the Humber, England, S13 9BZ, United Kingdom are 53.3723004, -1.3830121.
The geograpical coordinate of Rollestone Wood, Herdings, Sheffield, Yorkshire and the Humber, England, United Kingdom are 53.34689625, -1.4410266841411292.
The geograpical coordinate of Dore, Sheffield, Yorkshire and the Humber, England, S17 3GD, United Kingdom are 53.3262748, -1.540347.
The geograpical coordinate of Dronfield, Sheffield Road, Coal Aston, Dronfield, North East Derbyshire, Derbyshire, East Midlands, England, S18 2DH, United Kingdom are 53.301481, -1.4694368.
```

5. After clean up, the location fact is as shown below.

	postal_code	area	neighborhood_postal_code	neighborhood	latitude	longitude
0	S1	Orchard Square	S2	Arbourthorne	53.361389	-1.443123
1	S1	Orchard Square	S3	Burngreave	53.393988	-1.458320
2	S1	Orchard Square	S4	Grimesthorpe	53.407042	-1.439704
3	S1	Orchard Square	S6	Hillsborough	53.411373	-1.500635
4	S1	Orchard Square	S10	Fulwood	53.365013	-1.543243
5	S1	Orchard Square	S11	Ecclesall	53.355774	-1.511037
6	S2	Arbourthorne	S14	Rollestone	53.346896	-1.441027
7	S2	Arbourthorne	S12	Hackenthorpe	53.343606	-1.381049
8	S2	Arbourthorne	S13	Handsworth	53.372300	-1.383012
9	S2	Arbourthorne	S8	Norton Woodseats	53.327421	-1.473432
10	S2	Arbourthorne	S7	Beauchief	53.335309	-1.500887
11	S2	Arbourthorne	S1	Orchard Square	53.381674	-1.470504
12	S2	Arbourthorne	S9	Attercliffe	53.392270	-1.430267
13	S3	Burngreave	S1	Orchard Square	53.381674	-1.470504
14	S3	Burngreave	S4	Grimesthorpe	53.407042	-1.439704

6. Create a map with folium library



7. Find nearby venues for every area by Foursquare API (radius 1000 m) and create a sorted list of area with places similar to restaurants, cafes, etc.

```
[28]: print(df_venues.shape)
      df_venues.head()
```

```
(375, 7)
```

[28]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	S1	53.381674	-1.470504	Marmadukes Cafe Deli	53.381121	-1.468207	Café
1	S1	53.381674	-1.470504	Peace Gardens	53.379860	-1.469582	Garden
2	S1	53.381674	-1.470504	Crucible Theatre	53.381021	-1.466653	Theater
3	S1	53.381674	-1.470504	Couch	53.383272	-1.471219	Café
4	S1	53.381674	-1.470504	Cafe Piazza	53.382512	-1.470573	Mediterranean Restaurant

```
[29]: df_venues.groupby('Neighborhood').count()
```

[29]:

	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
Neighborhood						
S1	100	100	100	100	100	100
S10	4	4	4	4	4	4
S11	3	3	3	3	3	3
S12	2	2	2	2	2	2
S13	4	4	4	4	4	4
S14	3	3	3	3	3	3
S17	4	4	4	4	4	4
S18	9	9	9	9	9	9
S2	5	5	5	5	5	5
S21	19	19	19	19	19	19

8. Since we focus on foods and daily essentials, we pick the interest categories and ignore the irrevelent one.

```
df_venues['Venue Category'].unique()[:100]
```

```
array(['Café', 'Garden', 'Theater', 'Mediterranean Restaurant', 'Bar',
       'Movie Theater', 'Coffee Shop', 'Sushi Restaurant', 'Pub',
       'Concert Hall', 'Noodle House', 'Bagel Shop', 'Cuban Restaurant',
       'Pizza Place', 'Cosmetics Shop', 'Butcher', 'Department Store',
       'Caribbean Restaurant', 'Italian Restaurant', 'Bubble Tea Shop',
       'Art Gallery', 'Plaza', 'Asian Restaurant', 'Multiplex',
       'Hobby Shop', 'BBQ Joint', 'Hotel',
       'Vegetarian / Vegan Restaurant', 'Food Court', 'Bookstore',
       'English Restaurant', 'Indian Restaurant', 'Chinese Restaurant',
       'Burrito Place', 'Casino', 'Bakery', 'Restaurant',
       'Fried Chicken Joint', 'Comedy Club', 'Clothing Store',
       'Turkish Restaurant', 'Discount Store', 'Stationery Store',
       'Sandwich Place', 'Portuguese Restaurant', 'Chocolate Shop',
       'Steakhouse', 'Pharmacy', 'American Restaurant', 'Nightclub',
       'Music Venue', 'Rock Club', 'Grocery Store', 'Tram Station',
       'Breakfast Spot', 'Gym / Fitness Center', 'Supermarket',
       'Halal Restaurant', 'Burger Joint', 'Bowling Alley', 'Pet Store',
       'Park', 'Gym', 'Soccer Stadium', 'Fast Food Restaurant', 'Trail',
       'Wine Shop', 'Shopping Plaza', 'Convenience Store',
       'Sporting Goods Shop', 'Newsstand', 'Dam', 'Bus Stop',
       'Construction & Landscaping', 'Diner', 'Light Rail Station',
       'Scenic Lookout', 'Train Station', 'Farm', 'Toy / Game Store',
       'Tapas Restaurant', 'Beer Store', 'Betting Shop', 'Flea Market',
       'Tea Room', 'Pool', 'Outdoor Supply Store', 'Pool Hall',
       'Shopping Mall', 'Flower Shop', 'Electronics Store',
       'Arts & Entertainment', 'Business Service', 'Bus Station',
       'Falafel Restaurant', 'Museum', 'College Auditorium', 'Wine Bar',
       'Design Studio', 'Gelato Shop'], dtype=object)
```

```
my_concern = postal_code_grouped[["Neighborhood","Pharmacy","Convenience Store","Supermarket","Grocery Store",'English Restaurant', 'Indian Restaurant', 'Chinese Restaurant',
                                  'Pub', 'Asian Restaurant','Bar', 'Fast Food Restaurant','Burger Joint', 'Diner', 'Restaurant', 'Fried Chicken Joint', 'Fast Food Restaurant',
                                  'Flea Market'
                                 ]]
my_concern
```

9. We can base on previous learned knowledge to display the ten most frequent places.

```
num_top_venues = 5

for hood in my_concern['Neighborhood']:
    print("----"+hood+"----")
    temp = my_concern[my_concern['Neighborhood'] == hood].T.reset_index()
    temp.columns = ['venue','freq']
    temp = temp.iloc[1:]
    temp['freq'] = temp['freq'].astype(float)
    temp = temp.round({'freq': 2})
    print(temp.sort_values('freq', ascending=False).reset_index(drop=True).head(num_top_venues))
    print('\n')
```

```
----S1----
               venue  freq
0                Pub  0.09
1                Bar  0.09
2   Indian Restaurant  0.02
3  Chinese Restaurant  0.02
4          Restaurant  0.02


----S10----
               venue  freq
0       Grocery Store  0.25
1            Pharmacy  0.00
2   Convenience Store  0.00
3         Supermarket  0.00
4  English Restaurant  0.00
```

```
num_top_venues = 10

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

# create columns according to number of top venues
columns = ['Neighborhood']
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))

# create a new dataframe
neighborhoods_venues_sorted = pd.DataFrame(columns=columns)
neighborhoods_venues_sorted['Neighborhood'] = my_concern['Neighborhood']

for ind in np.arange(postal_code_grouped.shape[0]):
    neighborhoods_venues_sorted.iloc[ind, 1:] = return_most_common_venues(my_concern.iloc[ind, :], num_top_venues)

neighborhoods_venues_sorted
```

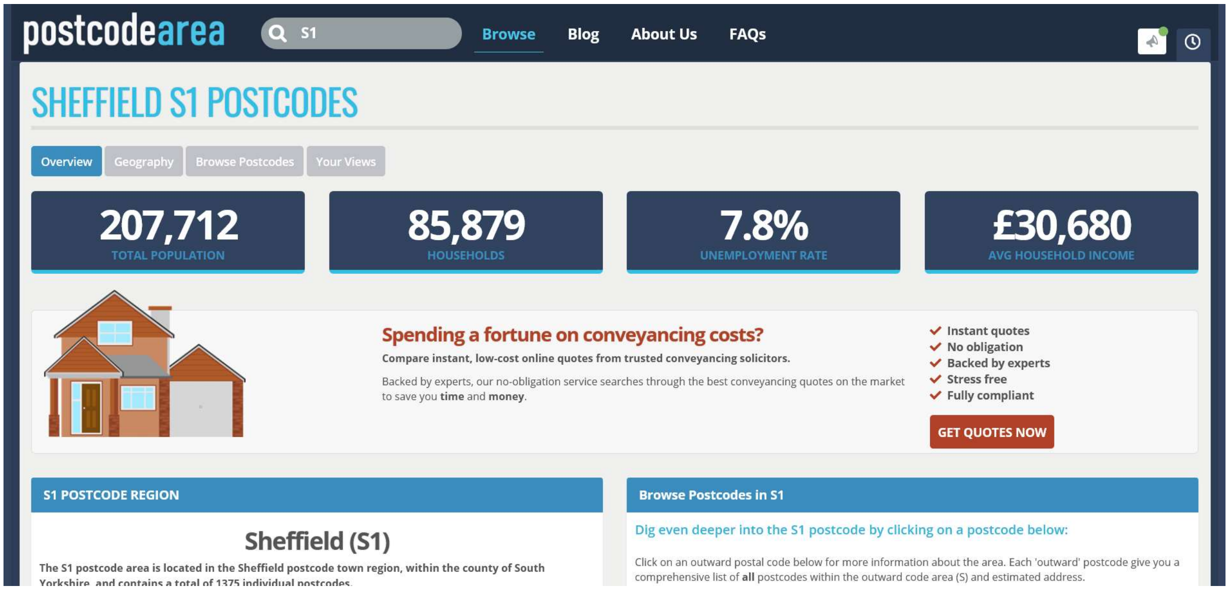
	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue	9th Most Common Venue	10th Most Common Venue
0	S1	Bar	Pub	Restaurant	Chinese Restaurant	Indian Restaurant	Fried Chicken Joint	Asian Restaurant	English Restaurant	Pharmacy	Flea Market
1	S10	Grocery Store	Flea Market	Fast Food Restaurant	Fried Chicken Joint	Restaurant	Fast Food Restaurant	Bar	Asian Restaurant	Pub	Chinese Restaurant
2	S11	Pub	Flea Market	Fast Food Restaurant	Fried Chicken Joint	Restaurant	Fast Food Restaurant	Bar	Asian Restaurant	Chinese Restaurant	Indian Restaurant
3	S12	Flea Market	Fast Food Restaurant	Fried Chicken Joint	Restaurant	Fast Food Restaurant	Bar	Asian Restaurant	Pub	Chinese Restaurant	Indian Restaurant

10. So now it is ready for next step.

4.2 Part 2 Population

We collect and prepare the population data, since massive area in Sheffield is country side and green area. We try to avoid to pick those wild life areas. Meanwhile, unemployment rate and household income, household size also affect.

1. We collect data from Sheffield postal area, https://www.postcodearea.co.uk/postaltowns/sheffield/



The postcode area provides information about Cities, Populations and Income.

2. It forms the csv named: population.csv and sample as below.

```
City,Postal Code,Population,Household,Unemployment,Household Income
Sheffield ,S1,207712,85879,0.078,30680
Sheffield ,S2,143891,60186,0.105,27560
Sheffield ,S3,98762,41521,0.104,41080
Sheffield ,S4,189554,83361,0.18,27040
Sheffield ,S5,60467,24840,0.122,31200
Sheffield ,S6,301680,129156,0.057,32240
```

3. Import the data

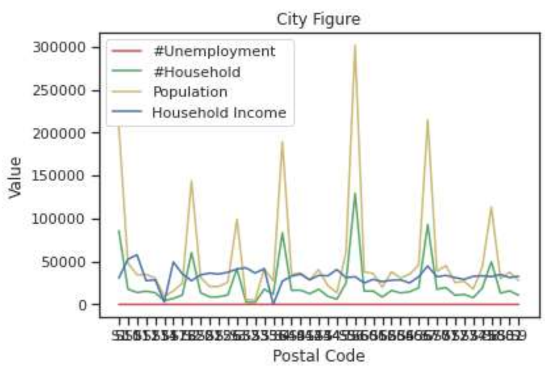
```python
#Read data
df=pd.read_csv('Population.csv')

#Collect Postal Code mapping
df_population = df[{'Postal Code', 'Population', 'Household', 'Unemployment', 'Household Income'}]
df_population.set_index('Postal Code')
df_population.rename(columns={"Postal Code": "postal_code"}, inplace=True)
df_population
```

	Unemployment	Household	postal_code	Population	Household Income
0	0.0780	85879	S1	207712	30680
1	0.1050	60186	S2	143891	27560
2	0.1040	41521	S3	98762	41080
3	0.1800	83361	S4	189554	27040
4	0.1220	24840	S5	60467	31200
5	0.0570	129156	S6	301680	32240
6	0.0630	92883	S7	214929	44720

4. We plot a graph to illustrate the figures

```
sns.set(style = 'ticks') #white background

sns.lineplot(x = 'postal_code', y = 'Unemployment', data = df_population, color = 'r', label = '#Unemployment')
sns.lineplot(x = 'postal_code', y = 'Household', data = df_population, color = 'g', label = '#Household')
sns.lineplot(x = 'postal_code', y = 'Population', data = df_population, color = 'y', label = 'Population')
sns.lineplot(x = 'postal_code', y = 'Household Income', data = df_population, color = 'b', label = 'Household Income')

plt.title ('City Figure')
plt.xlabel ('Postal Code')
plt.ylabel ('Value')
```

```
Text(0, 0.5, 'Value')
```



5. We calculate the sum and then the percentage value.

```
#Calculate the mean
df_population_mean = df_population[{'Household','Population','Household Income'}]
df_population_mean = df_population_mean.sum().reset_index().T
df_population_mean = df_population_mean.rename(columns=df_population_mean.iloc[0]).drop(df_population_mean.index[0])
df_population_mean
```

	Household Income	Household	Population
0	1474720	1033258	2418344

```
df_population['Household_mean'] = df_population['Household'] / df_population_mean['Household'][0]
df_population['Population_mean'] = df_population['Population'] / df_population_mean['Population'][0]
df_population['Household_Income_mean'] = df_population['Household Income'] / df_population_mean['Household Income'][0]
df_population.drop(['Household','Population','Household Income'], axis=1, inplace=True)
df_population
```

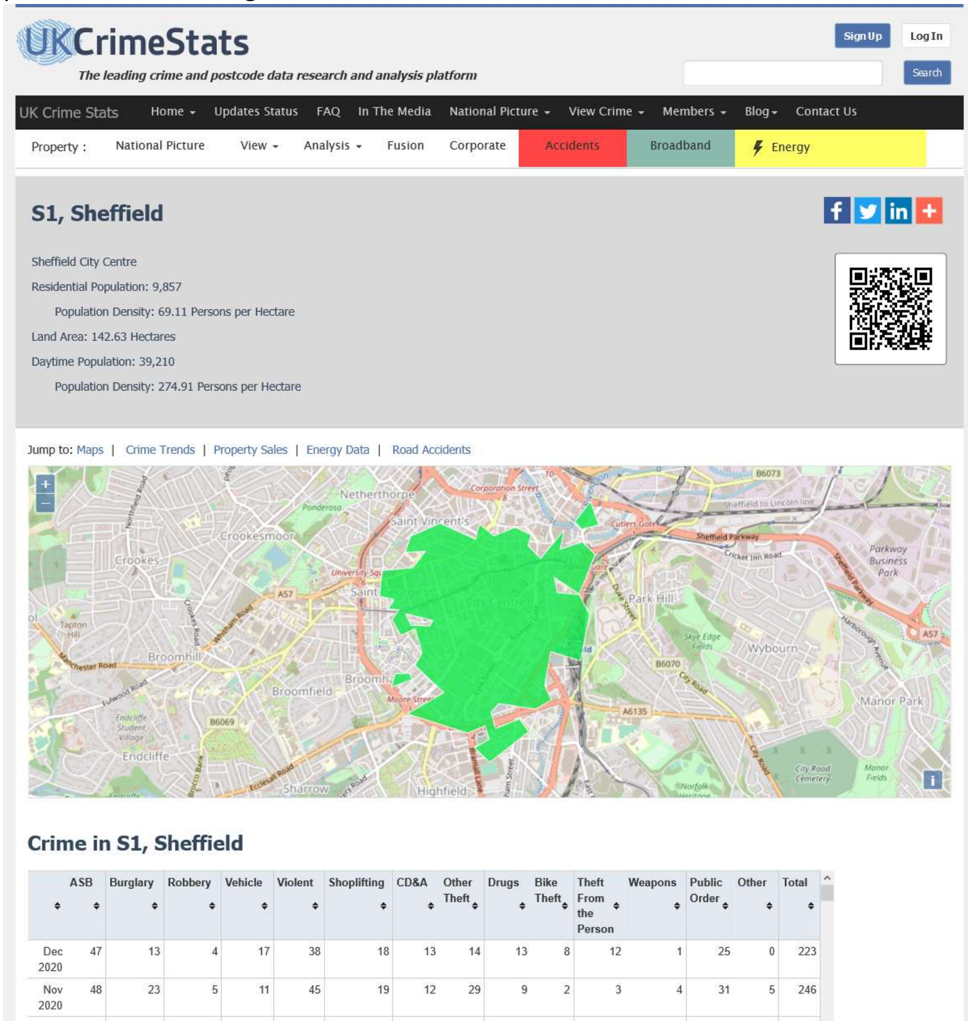
	Unemployment	postal_code	Household_mean	Population_mean	Household_Income_mean
0	0.0780	S1	0.083115	0.085890	0.020804
1	0.1050	S2	0.058249	0.059500	0.018688
2	0.1040	S3	0.040185	0.040839	0.027856
3	0.1800	S4	0.080678	0.078382	0.018336
4	0.1220	S5	0.024040	0.025003	0.021157
5	0.0570	S6	0.124999	0.124747	0.021862
6	0.0630	S7	0.089893	0.088874	0.030324

6. We merge back to Part 1 data and now it is ready for next step.

4.3 Part 3 Crime Rate

Apart from the bright side, every city also has their dark side, so we need to consider the crime rate as well.

1. We collect data from UK Crime map, https://www.ukcrimestats.com/. To make it simple, we pick the Dec 2020 figure.



2. It forms the csv named: population.csv and sample as below.

```
Postal Code,Crime Dec2020
S1,223
S2,585
S3,221
S4,222
```

3. Import the data

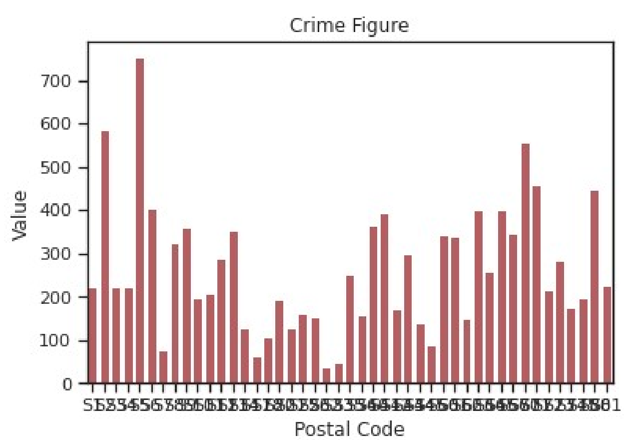
```
#Read data
df=pd.read_csv('Crime.csv')

#Collect Postal Code mapping
df_crime = df[{'Postal Code', 'Crime Dec2020'}]
df_crime.set_index('Postal Code')
df_crime.rename(columns={"Postal Code": "postal_code"}, inplace=True)
df_crime
```

	postal_code	Crime Dec2020
0	S1	223
1	S2	585
2	S3	221
3	S4	222
4	S5	752
5	S6	405

4. We plot a graph to illustrate the figures

```
sns.set(style = 'ticks') #white background

sns.barplot(x = 'postal_code', y = 'Crime Dec2020', data = df_crime, color = 'r', label = '#Crime')

plt.title ('Crime Figure')
plt.xlabel ('Postal Code')
plt.ylabel ('Value')
```

```
Text(0, 0.5, 'Value')
```



5. We calculate the sum and then the percentage value.

```
df_crime_mean = df_crime[{'Crime Dec2020'}]
df_crime_mean = df_crime_mean.sum().reset_index().T
df_crime_mean = df_crime_mean.rename(columns=df_crime_mean.iloc[0]).drop(df_crime_mean.index[0])
df_crime_mean
```

	Crime Dec2020
0	11905

```
df_crime['crime_mean'] = (1 - (df_crime['Crime Dec2020'] / df_crime_mean['Crime Dec2020'][0])) /1000
df_crime.drop(['Crime Dec2020'], axis=1, inplace=True)
df_crime
```

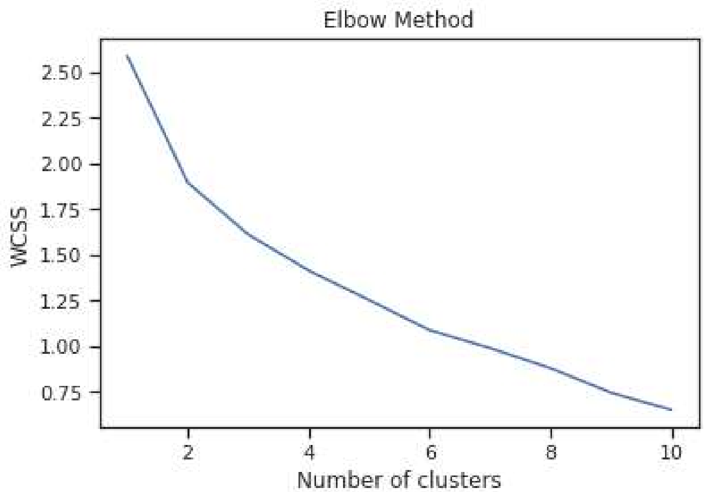
	postal_code	crime_mean
0	S1	0.000981
1	S2	0.000951
2	S3	0.000981
3	S4	0.000981
4	S5	0.000937
5	S6	0.000966

6. We merge back to Part 1 data and now it is ready for next step.

4.4 Part 4 Clustering

We use Elbow method to estimate best-k value for our dataset.

```python
import matplotlib.pyplot as plt
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10, random_state=0)
    kmeans.fit(postal_code_grouped_clustering)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```



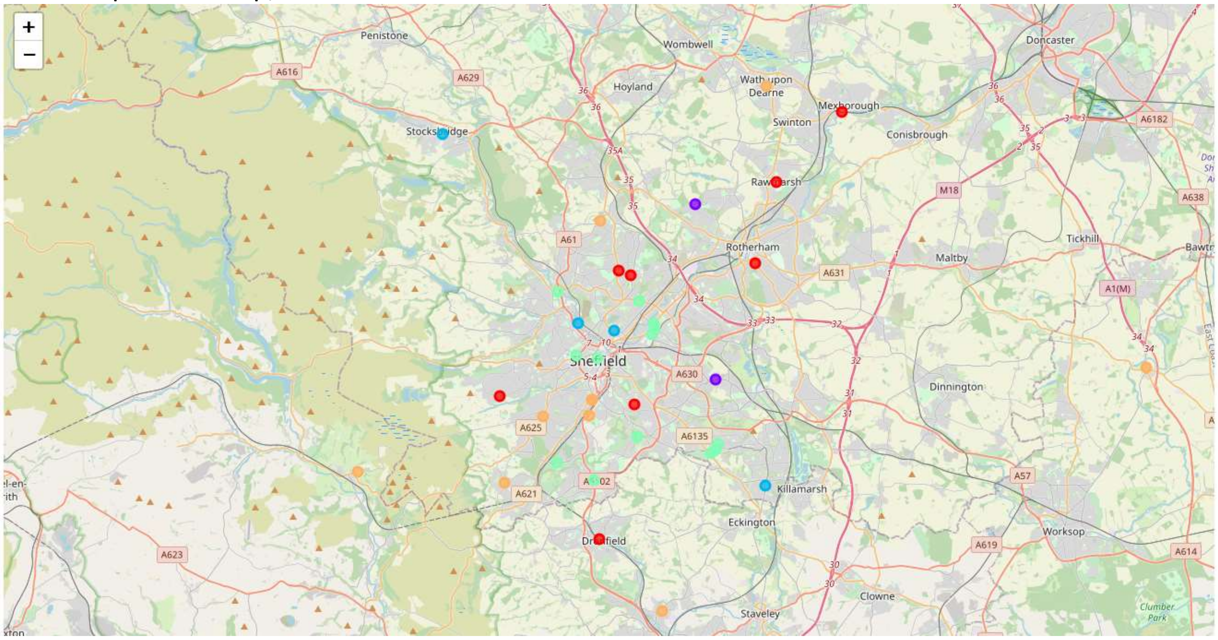
The best k value is 5, so we run the k-means clustering to group them into 5 categories.

```python
# add clustering labels
final_result = postal_code_merged.copy()
final_result.insert(0, 'Cluster Labels', kmeans.labels_)
```

```python
final_result.sort_values(by=['Cluster Labels','Neighborhood'], inplace=True)
final_result
```

	Cluster Labels	Neighborhood	Pharmacy	Convenience Store	Supermarket	Grocery Store	English Restaurant	Indian Restaurant	Chinese Restaurant	Pub	Asian Restaurant	Bar	Fast Food Restaurant	Restaurant	Fried Chicken Joint	Fast Food Restaurant	Flea Market	Pizza Place
1	0	S10	0.000000	0.000000	0.000000	0.250000	0.00	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	0.000000	0.000000
7	0	S18	0.000000	0.000000	0.000000	0.125000	0.00	0.000000	0.125000	0.000000	0.000000	0.125000	0.000000	0.000000	0.00	0.000000	0.000000	0.125000
8	0	S2	0.000000	0.000000	0.000000	0.166667	0.00	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	0.000000	0.166667
21	0	S5	0.000000	0.000000	0.000000	0.250000	0.00	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	0.000000	0.000000
23	0	S60	0.000000	0.000000	0.000000	0.250000	0.00	0.000000	0.250000	0.250000	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	0.000000	0.000000
25	0	S62	0.000000	0.000000	0.000000	0.250000	0.00	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	0.000000	0.000000
27	0	S64	0.000000	0.000000	0.125000	0.125000	0.00	0.000000	0.000000	0.125000	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	0.000000	0.000000
31	0	S70	0.000000	0.000000	0.000000	0.250000	0.00	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	0.000000	0.000000
4	1	S13	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	0.000000	0.250000	0.000000	0.250000	0.000000	0.000000	0.00	0.000000	0.000000	0.000000
24	1	S61	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	0.000000	0.333333	0.000000	0.333333	0.000000	0.000000	0.00	0.000000	0.000000	0.000000

After we plot the map, we can see the result



And we can have a separated cluster examination

Cluster 1

```
final_result.loc[final_result['Cluster Labels'] == 0, final_result.columns[[0] + [1] + list(range(5, final_result.shape[1]))]]
```

	area	postal_code	Pharmacy	Convenience Store	Supermarket	Grocery Store	English Restaurant	Indian Restaurant	Chinese Restaurant	Pub	Asian Restaurant	Bar	Fast Food Restaurant	Fast Food Restaurant	Restaurant	Fried Chicken Joint	Fast Food Restaurant	Fast Food Restaurant	M
45	Fulwood	S10	0.0	0.0	0.000	0.250000	0.0	0.0	0.000	0.000	0.0	0.000	0.0	0.0	0.0	0.0	0.0	0.0	
77	Dronfield	S18	0.0	0.0	0.000	0.125000	0.0	0.0	0.125	0.000	0.0	0.125	0.0	0.0	0.0	0.0	0.0	0.0	
6	Arbourthorne	S2	0.0	0.0	0.000	0.166667	0.0	0.0	0.000	0.000	0.0	0.000	0.0	0.0	0.0	0.0	0.0	0.0	
21	Firth Park	S5	0.0	0.0	0.000	0.250000	0.0	0.0	0.000	0.000	0.0	0.000	0.0	0.0	0.0	0.0	0.0	0.0	
140	Moorgate, Rotherham	S60	0.0	0.0	0.000	0.250000	0.0	0.0	0.250	0.250	0.0	0.000	0.0	0.0	0.0	0.0	0.0	0.0	
151	Rawmarsh	S62	0.0	0.0	0.000	0.250000	0.0	0.0	0.000	0.000	0.0	0.000	0.0	0.0	0.0	0.0	0.0	0.0	
159	Mexborough	S64	0.0	0.0	0.125	0.125000	0.0	0.0	0.000	0.125	0.0	0.000	0.0	0.0	0.0	0.0	0.0	0.0	
171	Barnsley	S70	0.0	0.0	0.000	0.250000	0.0	0.0	0.000	0.000	0.0	0.000	0.0	0.0	0.0	0.0	0.0	0.0	

5. Results

The following is the main research results:

1. The best place to live is S10, S18, S2, S5 which are nearest the city center

6. Discussions

The statistic result shows that S10, S18, S2, S5 is the best choice with the balance between shops, population, houseolds, housesold income, employment rate and crime.

I suggest to have further factors to inject in the model:

1. Public transport routes,
2. Distance from train stations, inter-city major routes (M1 highway)
3. More finite grouping on shops by weighting (i.e. supermarket has higher weighting than restaurant)

4. More finite grouping on household age group
5. More detail distribution on house size, house type and price range
6. Distance to school (By driving time/ public transport)
7. School ranking

7. Conclusions

The primary target is to find a suitable living place in Sheffield, UK for my new home. Our requirements are:

1. It should be convenince to get foods and daily essentials.
2. Low crime rate
3. High employment rate or moderate and above income.
4. With certain populations and avoid too countryside area.

As a result, area S10, S18, S2, S5 is the best choice with the balance between shops, population, houseolds, housesold income, employment rate and crime.

Although there still some options, there are too far away from city center.