GitHub Repository

what are the spatial patterns of overall crime rates and the rates of the three crime categories in London?

Introduction

This notebook aimes to find spatial patterns of crime in London. Including 3 typess of crimes. I use a serious ways to validate what I found in clustering analysis.

Literature Review

In both academic and practical settings, the most popular methods for detecting crime hot spots are based on spatial autocorrelation and kernel density analysis(Murray and Grubesic, 2013). In contrast, non-hierarchical clustering techniques like k-means are less frequently utilized in various scenarios(Murray and Grubesic, 2013). However, Alkhaibari and Chung(2017) employed clustering algorithms like K-Means and agglomerative clustering on the 2015 NYPD Stop, Question and Frisk Report Database to analyze crime locations and stop reasons, aiming to reduce urban crime rates. By applying internal validation methods to compare K-Means and agglomerative clustering, it was determined that the K-Means algorithm performed the best among all tested clustering algorithms(Alkhaibari and Chung2017). This contradiction inspired me to use the K-Means algorithm to detect clusters of crime in London and their spatial distribution, while also attempting to extract useful insights from the cluster distributions obtained through analysis.

In addition, In criminology, a widely acknowledged aspect is the consistent age pattern observed in criminal activities, which has remained invariant across various social conditions(Hirschi and Gottfredson, 1983). Steffensmeier et al(1989) utilized arrest data from the FBI's Uniform Crime Reports analyzing parameters including the peak age of criminality, the overall shape of the age-crime curve, and the rate of decline from the peak age and their findings indicate that although crime rates generally decline throughout life after initially rising during adolescence, some types of crimes may peak later in life, decline more slowly, or both. Therefore, I believe it is meaningful to include the proportion of adolescents in each LSOA (Lower Layer Super Output Area) when conducting cluster analysis on crime rates in London. This also suggests that different types of crimes may

exhibit distinct characteristics, hence I will perform separate cluster analyses for different types of crimes.

According to the economic theory of crime, an increase in unemployment rates leads to a rise in property crime rates. One finding indicates that unemployment has a significant positive impact on property crime rates, and this result is consistent across different model specifications but the evidence for violent crime is relatively weak(Raphael and Winter-Ebmer, 2001). Another study shows that unemployment significantly positively affects certain types of property crimes, such as burglary, car theft, and bicycle theft(Edmark, 2005). Therefore, I plan to incorporate unemployment rates into my study to examine whether clusters with high crime rates also have correspondingly high unemployment rates. This will help further understand the relationship between crime rates and economic conditions.

Based on the previous research applied to the context of London crime data, I plan to categorize all types of crimes into three main categories: Public Order, Violent, and Property. (Xu, Pennington-Gray and Kim, 2019;Flatley, 2017)

Research Question

Therefore, based on the discussion above, my research question is: Considering unemployment rates and the proportion of adolescents, what are the spatial patterns of overall crime rates and the rates of the three crime categories (Public Order, Violent, Property) in London? How can we interpret the formation of these clusters, and what insights can be gained from them?

Methodology

The mean that I will use to detect clustering is K-Means. K-means and its variants are commonly used for unsupervised clustering in pattern recognition and machine learning. However, K-means does not fully conform to the standards of unsupervised clustering methods because its effectiveness largely depends on the choice of initial cluster centers and the predetermined number of clusters(K. P. Sinaga and M. -S. Yang, 2020).

- How K-Means Works:

1. Initialization: Start by selecting K initial points as the centroids of the clusters. These points can be chosen randomly from the dataset .
2. Assignment: Assign each data point to the nearest cluster by calculating the distance from each data point to each centroid. The most common distance metric used is Euclidean distance.
3. Update: Once all points have been assigned to clusters, recalculate the centroids of these clusters. The new centroid is typically the mean of all points

assigned to that cluster.
4. Iteration: Repeat the assignment and update steps until the centroids no longer change significantly, indicating that the clusters have stabilized, or a predetermined number of iterations is reached.

- Advantages of K-Means:

1. Efficiency: K-means is computationally efficient, making it suitable for large datasets, although its computational cost grows as the number of clusters increases(Govender and Sivakumar, 2020).
2. Simplicity: The algorithm is straightforward to implement and understand, which makes it a good baseline method for clustering tasks.
3. Scalability: It scales well to large numbers of samples and has been implemented in various platforms that support big data technologies.

- Disadvantages of K-Means:

1. Sensitivity to Initial Centroids: The initial selection of centroids can affect the final clusters. Poor initial centroids can result in sub-optimal clustering, which can be mitigated by multiple runs of the algorithm with different initializations.
2. Fixed Number of Clusters: The algorithm requires the number of clusters to be specified beforehand, which is not always practical when prior knowledge of the data is limited(Ahmed, Seraj and Islam, 2020).
3. Incapable of handling clusters of a non-convex shape: For non-convex functions, the situation with local minima and global minima is more complex. K-means tends to fall into the trap of local minima, overlooking the global minimum.
4. Sensitive to outliers, which can result in inaccurate clusters

Data loading and Exploration

These are the packages will be used.

In [ ]:
```
import os
from requests import get
from urllib.parse import urlparse
import numpy as np
import pandas as pd
import geopandas as gpd
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels
import statsmodels.api as sm
from sklearn.preprocessing import RobustScaler, MinMaxScaler
import plotly.express as px
import matplotlib.image as mpimg
from math import ceil
import networkx as nx
```

```
from sklearn.cluster import KMeans, DBSCAN, OPTICS, AgglomerativeClustering
from esda.adbscan import ADBSCAN

from scipy.cluster.hierarchy import dendrogram

import spopt
from spopt.region import MaxPHeuristic as MaxP


import libpysal
import warnings
```

Now let us load and explain the datasets.

In [ ]:
```
def cache_data(src: str, dest: str) -> str:
    """
    Downloads a file from the specified URL to a destination directory if not al

    Parameters:
        src (str): The source URL from which the file will be downloaded.
        dest (str): The destination directory where the file will be stored.

    Returns:
        str: The path to the downloaded or existing file.

    This function checks if the file exists in the specified destination directo
    If it does not exist, the function downloads the file. If the directory does
    it creates the directory (and any required parent directories).
    """

    # Parse the URL to extract the file name
    url = urlparse(src)
    fn = os.path.split(url.path)[-1]  # Extract the filename from the URL path

    # Combine the destination directory and filename to form the full path
    dfn = os.path.join(dest, fn)

    # Check if the file already exists at the destination
    if not os.path.isfile(dfn):
        print(f"{dfn} not found, downloading!")

        # Ensure the destination directory exists, create if it doesn't
        if not os.path.exists(dest):
            os.makedirs(dest, exist_ok=True)

        # Download the file and write to the destination filename
        with open(dfn, "wb") as file:
            response = get(src)  # Download the file
            file.write(response.content)  # Write the content to file

        print("\tDone downloading...")
    else:
        print(f"Found {dfn} locally!")

    return dfn  # Return the path to the file
```

In [ ]:
```
ddir  = os.path.join('data') # destination directory
spath = 'https://www.dropbox.com/scl/fi/' # source path for dropbox
```

This CSV file contains the attributes of each Lower Layer Super Output Area (LSOA) in England and Wales. It is based on the data from Census 2021 carried by Office for National Statistics, by the mean that aski people questions(Home - Office for National Statistics, no date). This is not raw data, as due to some limitations, only a portion of the data can be downloaded at a time. Therefore, I have downloaded the data multiple times in advance and then merged them to create this more complete file of LSOA attributes.

In [ ]:
```
lsoa_stat = pd.read_csv(cache_data(spath + 'qqo8gajxuo7ceentfw1wz/lsoa_full.csv?
```

```
Found data\lsoa_full.csv locally!
```

In [ ]:
```
lsoa_stat
```

Out[ ]:

	LSOA code	local authority code	local authority name	All households	Detached	Semi-detached	Terra
0	E01000001	E09000001	City of London	838	0	3	
1	E01000002	E09000001	City of London	824	1	2	
2	E01000003	E09000001	City of London	1018	1	0	
3	E01000005	E09000001	City of London	478	0	0	
4	E01032739	E09000001	City of London	882	2	7	
...	...	...	...	...	...	...	
4989	E01035718	E09000033	Westminster	732	11	30	
4990	E01035719	E09000033	Westminster	634	0	1	
4991	E01035720	E09000033	Westminster	587	2	2	
4992	E01035721	E09000033	Westminster	1302	4	22	
4993	E01035722	E09000033	Westminster	1178	6	2	

4994 rows × 235 columns

These CSV files provide street-level crime, outcome, and stop and search information, broken down by police force and 2021 lower layer super output area (LSOA)(About | data.police.uk, no date).

- Columns in the CSV files

1. Reported by: The force that provided the data about the crime.
2. Falls within: At present, also the force that provided the data about the crime. This is currently being looked into and is likely to change in the near future.
3. Longitude and Latitude: The anonymised coordinates of the crime.
4. LSOA code and LSOA name: References to the Lower Layer Super Output Area that the anonymised point falls into, according to the LSOA boundaries provided by the Office for National Statistics.
5. Crime type: One of the crime types listed in the Police.UK FAQ.
6. Last outcome category: A reference to whichever of the outcomes associated with the crime occurred most recently. For example, this crime's 'Last outcome category' would be 'Formal action is not in the public interest'.
7. Context: A field provided for forces to provide additional human-readable data about individual crimes. Currently, for newly added CSVs, this is always empty.

In [ ]:
```
crime = pd.read_csv(cache_data(spath + '5ckyvra71cdmzjqb2jvs4/2023london_crime.c
```

```
Found data\2023london_crime.csv locally!
```

In [ ]:
```
crime
```

Out[ ]:

	Crime ID	Reported by	Falls
0	8536e93fb3ce916daa4251bd53c1a4416ba4159a938340...	Metropolitan Police Service	Metro
1	483d52d514591a895c829dece6091c31f797b7dcfd0735...	Metropolitan Police Service	Metro
2	63343c1f1236bad8ce08d130f37760172dc33b20af2b56...	Metropolitan Police Service	Metro
3	a3d980f554d3ece9e8dcda8518ae87bfa9c75d62396105...	Metropolitan Police Service	Metro
4	bfb1d1da32341b7129e789130001d96f7e603088593dc5...	Metropolitan Police Service	Metro
...	...	...	
1048476	6ea18f57499bbe2b7d241605c9f7b7233378799f4ef2df...	City of London Police	I
1048477	93f09bc79a85358faa143cf835f86b9f6fbcf08b6976bf...	City of London Police	I
1048478	37022111f232b614c500c223664310f45148fb7393e06f...	City of London Police	I
1048479	8fd906ccd41a2c7d0f87a695c10fd58118005620e56860...	City of London Police	I
1048480	fd8da55614c44df3762dbbdd98d2d79d3117741bcb17ef...	City of London Police	I

1048481 rows × 11 columns

It is a GeoPackage containing detailed geospatial data about Lower Layer Super Output Areas (LSOAs) in London.

In [ ]:
```
lsoa = gpd.read_file(cache_data(spath+'6yc108b81708sgyetr3kt/lsoa_london.gpkg?rl
```

```
Found data\lsoa_london.gpkg locally!
```

Convert the crime data with coordinates into a GeoDataFrame to prepare for analyzing the number of crimes in each Lower Layer Super Output Area (LSOA) in

London.

In [ ]:
```
gdf_crime = gpd.GeoDataFrame(crime,
      geometry=gpd.points_from_xy(crime.Longitude, crime.Latitude, crs='epsg:432
```

This describes how I categorized all types of crimes into three major categories.

Public order: public order, drugs, possession of weapons, anti-social behavior.

Violent(involve force): robbery, violence and sexual offenses.

Property(without force): burglary, criminal damage and arson, shoplifting, theft, vehicle crime.

In [ ]:
```
crime_public_order = ['Public order', 'Drugs', 'Anti-social behaviour','Possessi
crime_violent = ['Robbery', 'Violence and sexual offences']
crime_property = ['Burglary','Theft from the person', 'Bicycle theft' 'Criminal
```

In [ ]:
```
gdf_crime = gdf_crime[~gdf_crime['Crime type'].isin(['Other crime'])]
```

In [ ]:
```
gdf_crime['crime_category'] = np.select(
    [gdf_crime['Crime type'].isin(crime_public_order),
     gdf_crime['Crime type'].isin(crime_violent),
     gdf_crime['Crime type'].isin(crime_property)],
    ['public order', 'violent', 'property'],
    default='other'
)
```

In [ ]:
```
gdf_crime.head()
```

Out[ ]:

	Crime ID	Reported by	Falls within
0	8536e93fb3ce916daa4251bd53c1a4416ba4159a938340...	Metropolitan Police Service	Metropolitan Police Service
1	483d52d514591a895c829dece6091c31f797b7dcfd0735...	Metropolitan Police Service	Metropolitan Police Service
2	63343c1f1236bad8ce08d130f37760172dc33b20af2b56...	Metropolitan Police Service	Metropolitan Police Service
3	a3d980f554d3ece9e8dcda8518ae87bfa9c75d62396105...	Metropolitan Police Service	Metropolitan Police Service
4	bfb1d1da32341b7129e789130001d96f7e603088593dc5...	Metropolitan Police Service	Metropolitan Police Service

In [ ]:
```
print(f"Data frame is {gdf_crime.shape[0]:,} x {gdf_crime.shape[1]}")
```

```
Data frame is 1,038,304 x 13
```

In [ ]:
```
gdf_crime = gdf_crime.to_crs('epsg:27700')
```

In [ ]:
```
gdf_crime_pu = gdf_crime[gdf_crime['crime_category']=='public order']
gdf_crime_vi = gdf_crime[gdf_crime['crime_category']=='violent']
gdf_crime_pr = gdf_crime[gdf_crime['crime_category']=='property']
```

Assign each crime to the corresponding Lower Layer Super Output Area (LSOA) based on its location and calculate the total number of crimes as well as the count for each of the three major crime categories.

In [ ]:
```
lsoa_merge = lsoa

# Spatial crime listings and lsoa, summarise by count
sjoin_lsoa = gpd.sjoin(gdf_crime, lsoa_merge)
count_dict = sjoin_lsoa['LSOA21CD'].value_counts().to_dict() # count the values
lsoa_merge['crime_count'] = lsoa_merge['LSOA21CD'].map(count_dict) # map it back
```

In [ ]:
```
sjoin_lsoa = gpd.sjoin(gdf_crime_pu, lsoa_merge)
count_dict = sjoin_lsoa['LSOA21CD'].value_counts().to_dict() # count the values
lsoa_merge['public_order_crime_count'] = lsoa_merge['LSOA21CD'].map(count_dict)
```

In [ ]:
```
sjoin_lsoa = gpd.sjoin(gdf_crime_vi, lsoa_merge)
count_dict = sjoin_lsoa['LSOA21CD'].value_counts().to_dict() # count the values
lsoa_merge['violent_crime_count'] = lsoa_merge['LSOA21CD'].map(count_dict) # map
```

In [ ]:
```
sjoin_lsoa = gpd.sjoin(gdf_crime_pr, lsoa_merge)
count_dict = sjoin_lsoa['LSOA21CD'].value_counts().to_dict() # count the values
lsoa_merge['property_crime_count'] = lsoa_merge['LSOA21CD'].map(count_dict) # ma
```

In [ ]:
```
lsoa_merge
```

Out[ ]:

	OBJECTID	LSOA21CD	LSOA21NM	GlobalID	geometry	crime_coun
0	1	E01000001	City of London 001A	f1865556-4e62-48e3-a025-d93a40f15e46	POLYGON ((532105.312 182010.574, 532162.491 18...	150.(
1	2	E01000002	City of London 001B	83e78aad-ee05-44a5-a8d3-077de6ed4053	POLYGON ((532634.497 181926.016, 532619.141 18...	382.(
2	3	E01000003	City of London 001C	69c457df-229e-4446-9577-95ac1c9be694	POLYGON ((532135.138 182198.131, 532158.250 18...	97.(
3	4	E01000005	City of London 001E	d537d59f-237a-45ed-a514-b0ab2a93e510	POLYGON ((533808.018 180767.774, 533649.037 18...	712.(
4	5	E01000006	Barking and Dagenham 016A	8d0081fe-69e2-48ac-aaf9-fd270d60e339	POLYGON ((545122.049 184314.931, 545271.849 18...	91.(
...	...	...	...	...	...	.
4989	33712	E01035718	Westminster 019G	8ce44e30-7ce1-4ce5-b99c-22fd920bafc8	POLYGON ((528427.719 180115.109, 528410.126 17...	1811.(
4990	33713	E01035719	Westminster 021F	4a204f9d-15c0-4043-848d-14bc504402e3	POLYGON ((530276.465 178962.701, 530238.780 17...	88.(
4991	33714	E01035720	Westminster 021G	9282c1d4-8e39-49a8-86a7-f85e97211fe5	POLYGON ((529867.785 178609.306, 529892.537 17...	156.(
4992	33715	E01035721	Westminster 023H	20805ec5-7f21-4ecb-b469-ecf528636354	POLYGON ((528674.775 178693.078, 528612.868 17...	875.(
4993	33716	E01035722	Westminster 024G	e5eec9e8-84b6-46aa-9764-dd651919e998	POLYGON ((529651.808 178210.675, 529692.695 17...	164.(

4994 rows × 9 columns

Calculate the variables I need and merge them by LSOA code

When calculating the unemployment rate, we use the number of people aged 16 and older as the denominator(Home - Office for National Statistics, no date).

In [ ]:
```
lsoa_stat['unemployed_rate'] = (lsoa_stat['Economically active: Unemployed'])/ls
lsoa_stat['teenage_rate'] = (lsoa_stat['Aged 16 to 17'] + lsoa_stat['Aged 15 '])
lsoa_stat_selected = lsoa_stat[['LSOA code','local authority code','All usual re
```

In [ ]:
```
# Merge selected columns from lsoa_stat to lsoa_merge by LSOA21CD
lsoa_merge1 = lsoa_merge.merge(lsoa_stat_selected, left_on = 'LSOA21CD',right_on

# Add density for each column in lsoa_merge using 'All usual residents' in lsoa_
lsoa_merge1['crime_rate'] = lsoa_merge1['crime_count']/lsoa_merge1['All usual re
lsoa_merge1['public_order_crime_rate'] = lsoa_merge1['public_order_crime_count']
lsoa_merge1['violent_crime_rate'] = lsoa_merge1['violent_crime_count']/lsoa_merg
lsoa_merge1['property_crime_rate'] = lsoa_merge1['property_crime_count']/lsoa_me
```

In [ ]:
```
lsoa_merge1
```

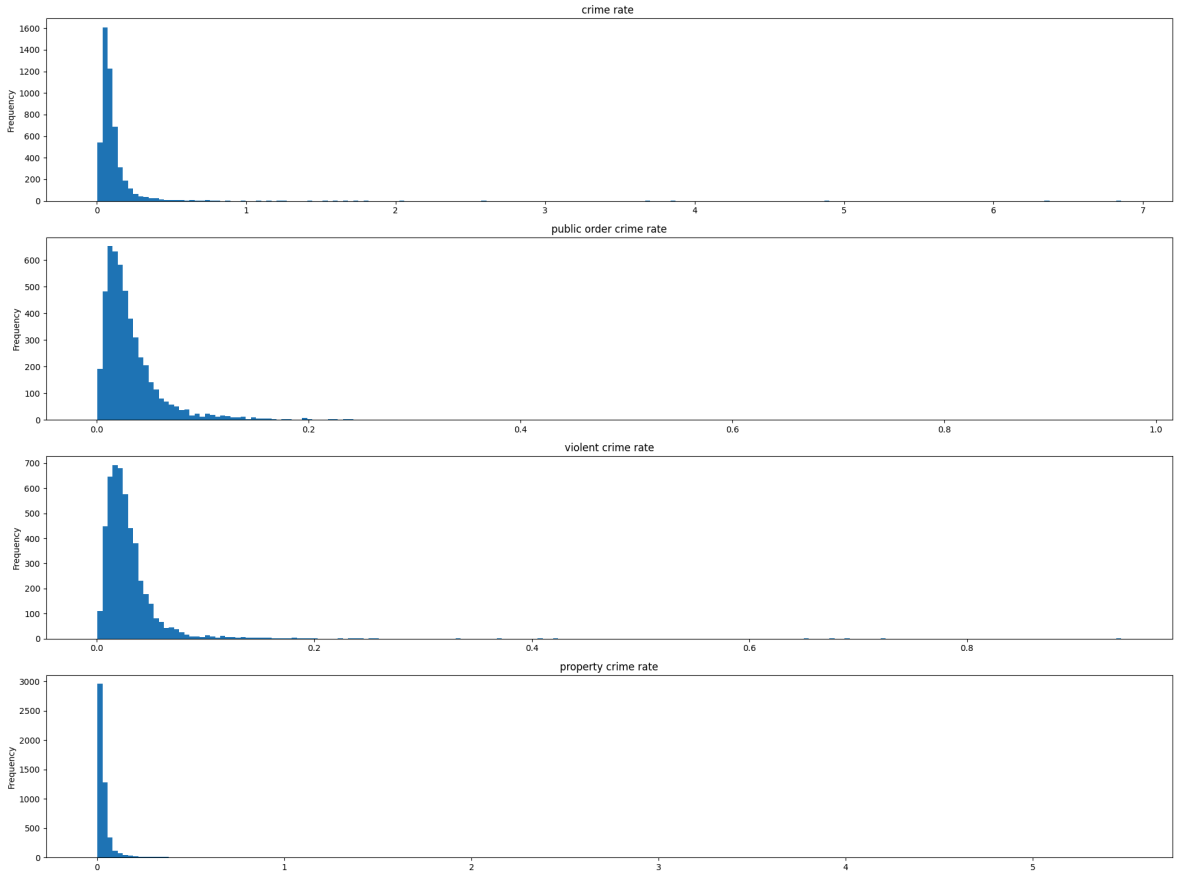
Out[ ]:

	OBJECTID	LSOA21CD	LSOA21NM	GlobalID	geometry	crime_coun
0	1	E01000001	City of London 001A	f1865556-4e62-48e3-a025-d93a40f15e46	POLYGON ((532105.312 182010.574, 532162.491 18...	150.(
1	2	E01000002	City of London 001B	83e78aad-ee05-44a5-a8d3-077de6ed4053	POLYGON ((532634.497 181926.016, 532619.141 18...	382.(
2	3	E01000003	City of London 001C	69c457df-229e-4446-9577-95ac1c9be694	POLYGON ((532135.138 182198.131, 532158.250 18...	97.(
3	4	E01000005	City of London 001E	d537d59f-237a-45ed-a514-b0ab2a93e510	POLYGON ((533808.018 180767.774, 533649.037 18...	712.(
4	5	E01000006	Barking and Dagenham 016A	8d0081fe-69e2-48ac-aaf9-fd270d60e339	POLYGON ((545122.049 184314.931, 545271.849 18...	91.(
...	...	...	...	...	...	.
4989	33712	E01035718	Westminster 019G	8ce44e30-7ce1-4ce5-b99c-22fd920bafc8	POLYGON ((528427.719 180115.109, 528410.126 17...	1811.(
4990	33713	E01035719	Westminster 021F	4a204f9d-15c0-4043-848d-14bc504402e3	POLYGON ((530276.465 178962.701, 530238.780 17...	88.(
4991	33714	E01035720	Westminster 021G	9282c1d4-8e39-49a8-86a7-f85e97211fe5	POLYGON ((529867.785 178609.306, 529892.537 17...	156.(
4992	33715	E01035721	Westminster 023H	20805ec5-7f21-4ecb-b469-ecf528636354	POLYGON ((528674.775 178693.078, 528612.868 17...	875.(
4993	33716	E01035722	Westminster 024G	e5eec9e8-84b6-46aa-9764-dd651919e998	POLYGON ((529651.808 178210.675, 529692.695 17...	164.(

4994 rows × 17 columns

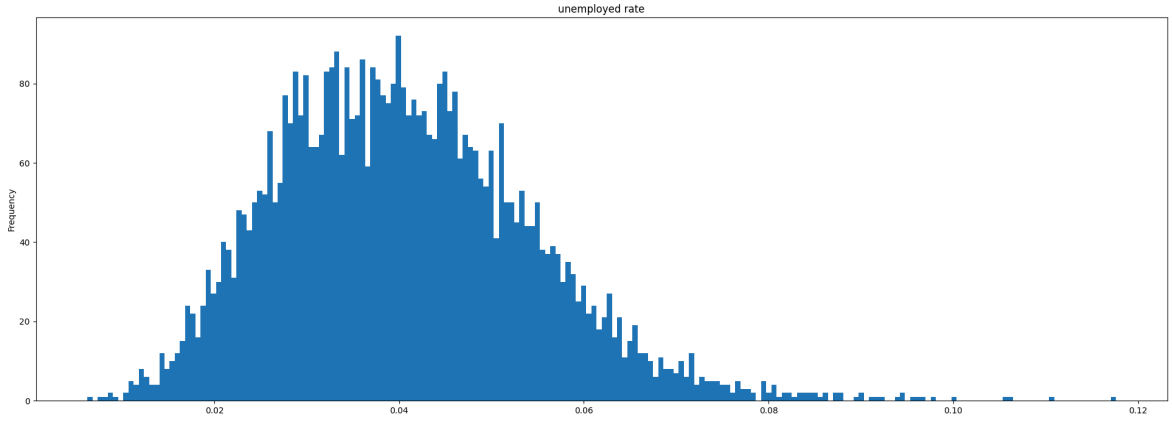
Look at the distribution of data

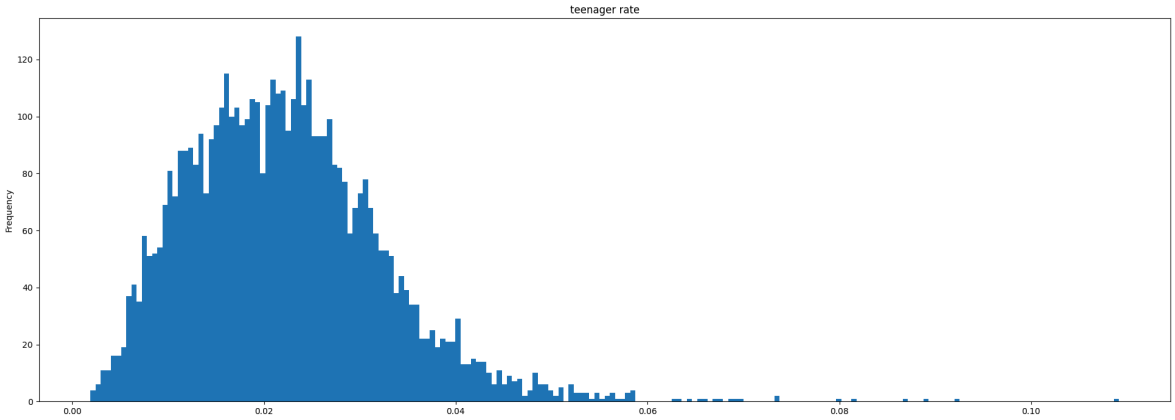
In [ ]:
```
fig, (ax1,ax2,ax3,ax4) = plt.subplots(4,1, figsize=(24,18))
lsoa_merge1.crime_rate.plot.hist(ax=ax1,bins=200)
lsoa_merge1.public_order_crime_rate.plot.hist(ax=ax2,bins=200)
lsoa_merge1.violent_crime_rate.plot.hist(ax=ax3,bins=200)
lsoa_merge1.property_crime_rate.plot.hist(ax=ax4,bins=200)
ax1.set_title('crime rate ')
ax2.set_title('public order crime rate')
ax3.set_title('violent crime rate')
ax4.set_title('property crime rate')
plt.show()
```



All histograms are positively skewed, meaning they have long tails to the right of the peak. This skewness indicates that the majority of observations are concentrated on the lower end of the crime rate spectrum, with a relatively small number of observations showing higher crime rates.

In [ ]:
```
fig, (ax1,ax2) = plt.subplots(2,1, figsize=(24,18))
lsoa_merge1.unemployed_rate.plot.hist(ax=ax1,bins=200)
lsoa_merge1.teenage_rate.plot.hist(ax=ax2,bins=200)

ax1.set_title('unemployed rate')
ax2.set_title('teenager rate')

plt.show()
```



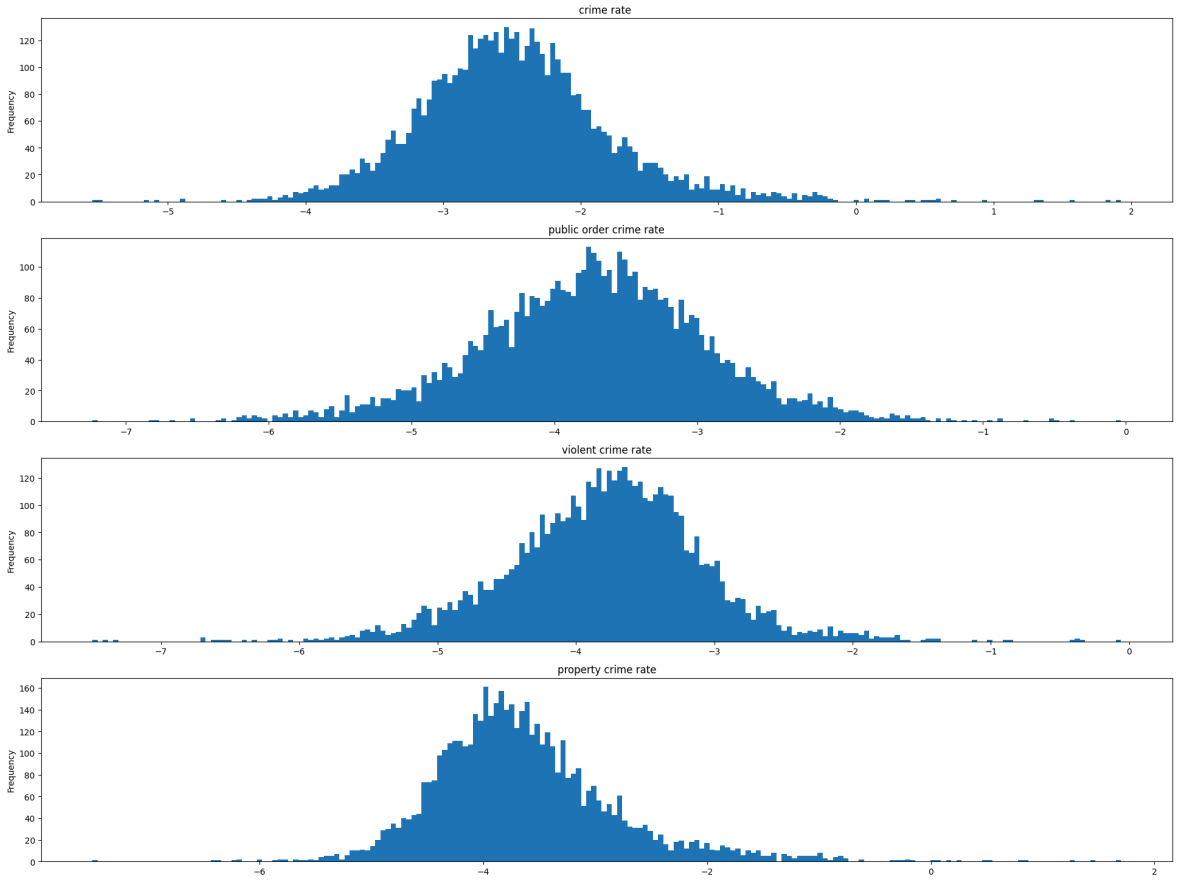


Because one of the drawbacks of the k-means algorithm is its sensitivity to outliers, and given that the crime rate data is very positively skewed, outliers have a significant impact. A log transformation is an effective method to reduce data variability and achieve data normality(Changyong et al., 2014).

In [ ]:
```
lsoa_merge1['crime_rate_log'] = np.log(lsoa_merge1['crime_rate'])
lsoa_merge1['public_order_crime_rate_log'] = np.log(lsoa_merge1['public_order_cr
lsoa_merge1['violent_crime_rate_log'] = np.log(lsoa_merge1['violent_crime_rate']
lsoa_merge1['property_crime_rate_log'] = np.log(lsoa_merge1['property_crime_rate
```

In [ ]:
```
fig, (ax1,ax2,ax3,ax4) = plt.subplots(4,1, figsize=(24,18))
fig.suptitle('After log transformation', size = 24)
lsoa_merge1.crime_rate_log.plot.hist(ax=ax1,bins=200)
lsoa_merge1.public_order_crime_rate_log.plot.hist(ax=ax2,bins=200)
lsoa_merge1.violent_crime_rate_log.plot.hist(ax=ax3,bins=200)
lsoa_merge1.property_crime_rate_log.plot.hist(ax=ax4,bins=200)
ax1.set_title('crime rate ')
ax2.set_title('public order crime rate')
ax3.set_title('violent crime rate')
ax4.set_title('property crime rate')
plt.show()
```

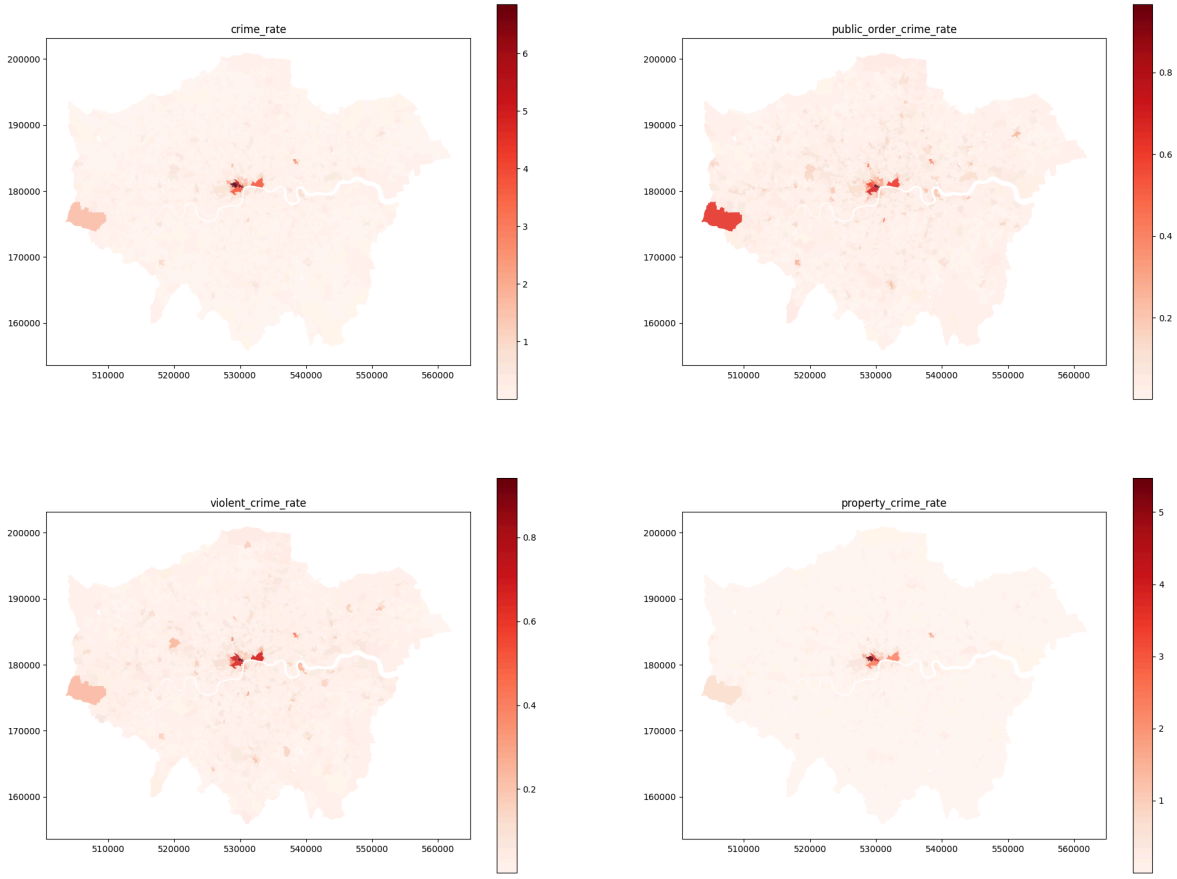
After log transformation



In [ ]:
```
lsoa_merge1.dropna(axis=0, inplace=True)
```

The images below are distributions of crime rates.

In [ ]:
```
fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2, figsize=(24, 18))
axes = [ax1, ax2, ax3, ax4]

for i, crime_type in enumerate(['crime_rate', 'public_order_crime_rate', 'violen
    lsoa_merge1.plot(ax=axes[i], column=crime_type, legend=True, cmap='Reds', al
    axes[i].set_title(f'{crime_type}')

plt.show()
```



Clustering!

In [ ]:
```
def mapping_clusters(labels_cluster):
    lsoa_merge1['cluster_nm'] = labels_cluster
    lsoa_merge1.plot(column='cluster_nm', categorical=True, legend=True, figsize
```

In [ ]:
```
# adapted from this tutorial: https://towardsdatascience.com/how-to-make-stunnin
def radar_plot_cluster_centroids(df_cluster_centroid):
    # parameters
    # df_cluster_centroid: a dataframe with rows representing a cluster centroid

    # add an additional element to both categories and restaurants that's identi
    # manually 'close' the line
    categories = df_cluster_centroid.columns.values.tolist()
    categories = [*categories, categories[0]]

    label_loc = np.linspace(start=0, stop=2 * np.pi, num=len(categories))

    plt.figure(figsize=(12, 8))
    plt.subplot(polar=True)
    for index, row in df_cluster_centroid.iterrows():
        centroid = row.tolist()
        centroid = [*centroid, centroid[0]]
        label = "Cluster {}".format(index)
        plt.plot(label_loc, centroid, label=label)
    plt.title('Cluster centroid comparison', size=20, y=1.05)
    lines, labels = plt.thetagrids(np.degrees(label_loc), labels=categories)
    plt.legend()
    plt.show()
```

In [ ]:
```
rs = RobustScaler(quantile_range=(10.0, 90.0))
```

In [ ]:
```
df1 = lsoa_merge1[['LSOA21CD','crime_rate_log','teenage_rate','unemployed_rate']
df1.set_index('LSOA21CD', inplace=True)
```

In [ ]:
```
df1.head()
```

Out[ ]:

LSOA21CD	crime_rate_log	teenage_rate	unemployed_rate
E01000001	-2.284421	0.014257	0.025811
E01000002	-1.285866	0.023878	0.024596
E01000003	-2.811140	0.022319	0.045213
E01000005	-0.444938	0.041404	0.061203
E01000006	-3.006661	0.025543	0.040645

Standardisation

We will use the `RobustScaler` class from the `sklearn` package for standardising the dataset.

The RobustScaler is robust to outlier. It removes the median and scales the data according to the quantile range (defaults to between 25 quantile and 75 quantile, also called the Interquartile Range or IQR).

In [ ]:
```
normed = df1.copy()
for c in df1.columns.values:
    normed[c] = rs.fit_transform(df1[c].values.reshape(-1,1))
    print("The range of {} is [{}, {}]".format(c, normed[c].min(), normed[c].max
normed.head()
```

```
The range of crime_rate_log is [-1.8316381716076273, 2.693144110293424]
The range of teenage_rate is [-0.7901207284399858, 3.571072852694266]
The range of unemployed_rate is [-0.9648933603301447, 2.263373197800955]
```

Out[ ]:

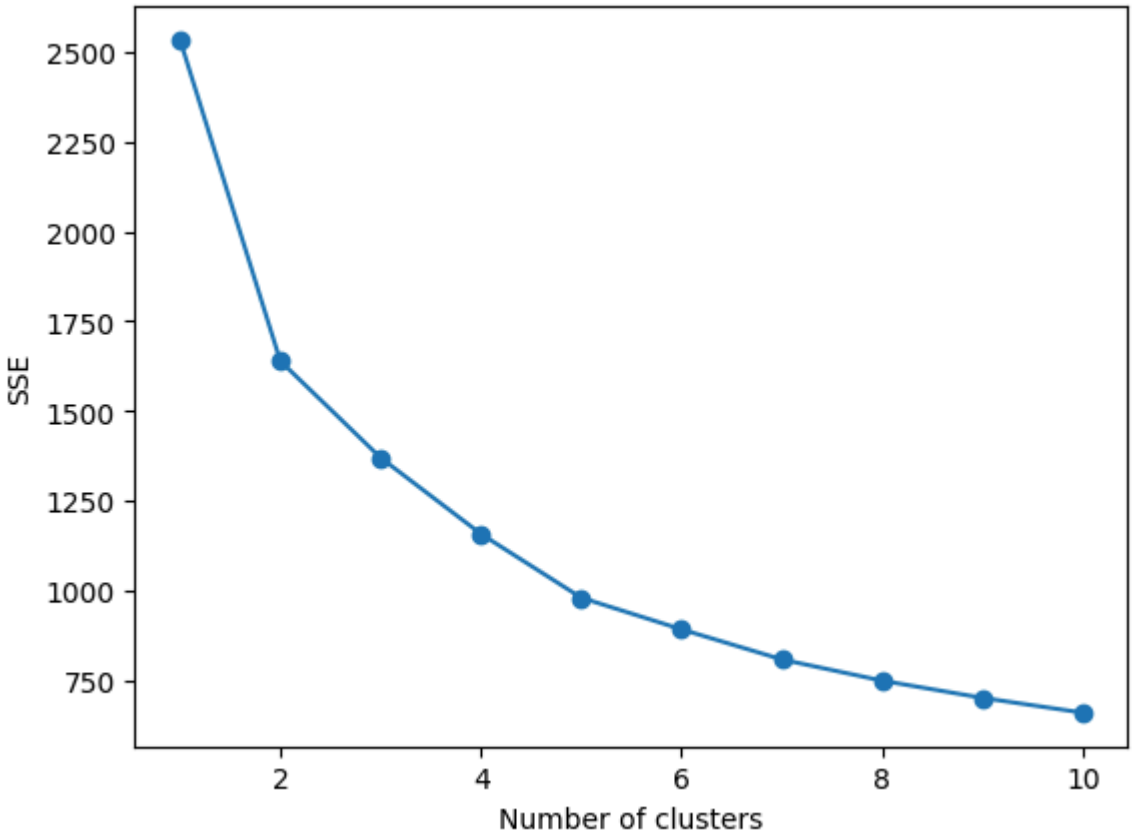
LSOA21CD	crime_rate_log	teenage_rate	unemployed_rate
E01000001	0.144478	-0.287620	-0.397777
E01000002	0.749115	0.103626	-0.432994
E01000003	-0.174457	0.040202	0.164711
E01000005	1.258308	0.816263	0.628308
E01000006	-0.292848	0.171331	0.032275

Find the best number of clusters

The Elbow method is a heuristic used in determining the number of clusters in a data set. The method consists of plotting the explained variation as a function of

the number of clusters and picking the elbow of the curve as the number of clusters to use. Generally, it's the point at which the rate of decrease sharply changes.

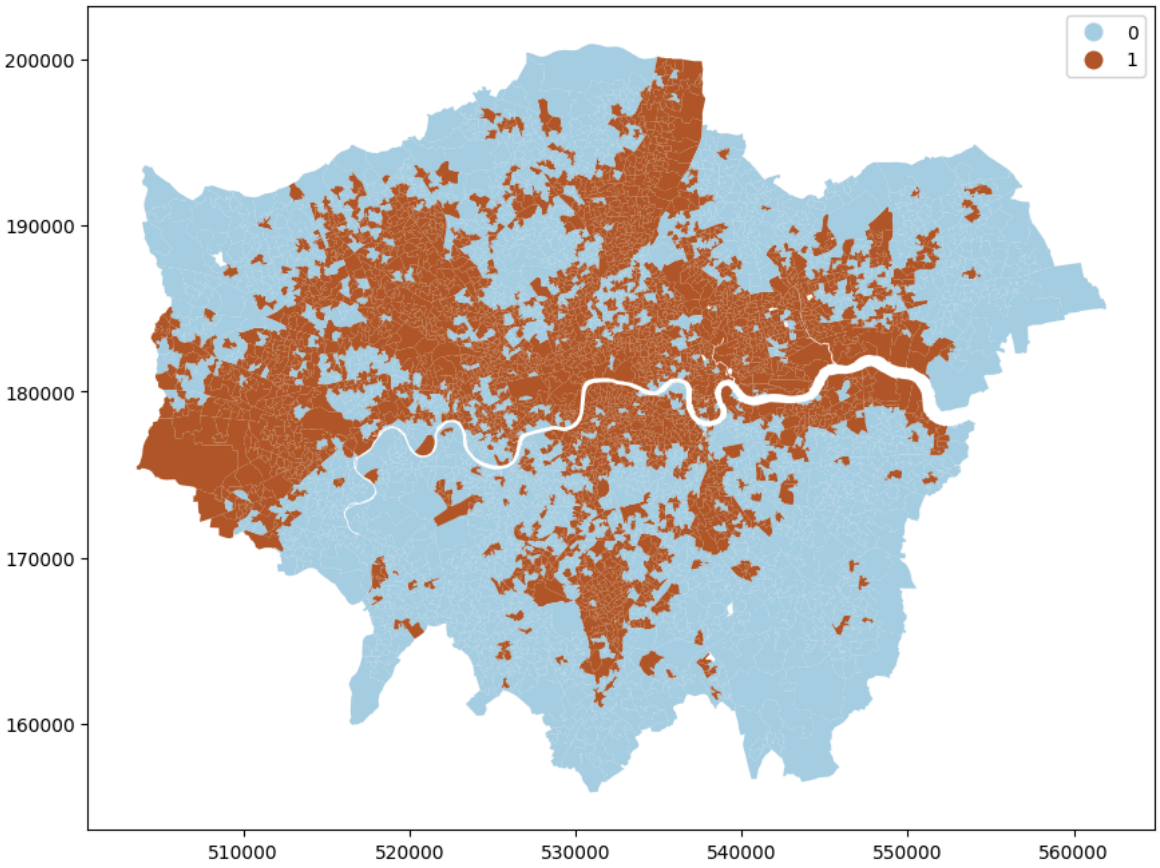
In [ ]:
```
# calculate SSE for a range of number of cluster
list_SSE = []
min_k = 1
max_k = 10
range_k = range(min_k, max_k+1)
for i in range_k:
    km = KMeans(
        n_clusters=i, init='random',
        n_init=10, max_iter=300,
        tol=1e-04, random_state=0
    )
    km.fit(normed)
    # inertia is a concept from physics. Roughly it means SSE of clustering.
    list_SSE.append(km.inertia_)

# plot
plt.plot(range_k, list_SSE, marker='o')
plt.xlabel('Number of clusters')
plt.ylabel('SSE')
plt.show()
```



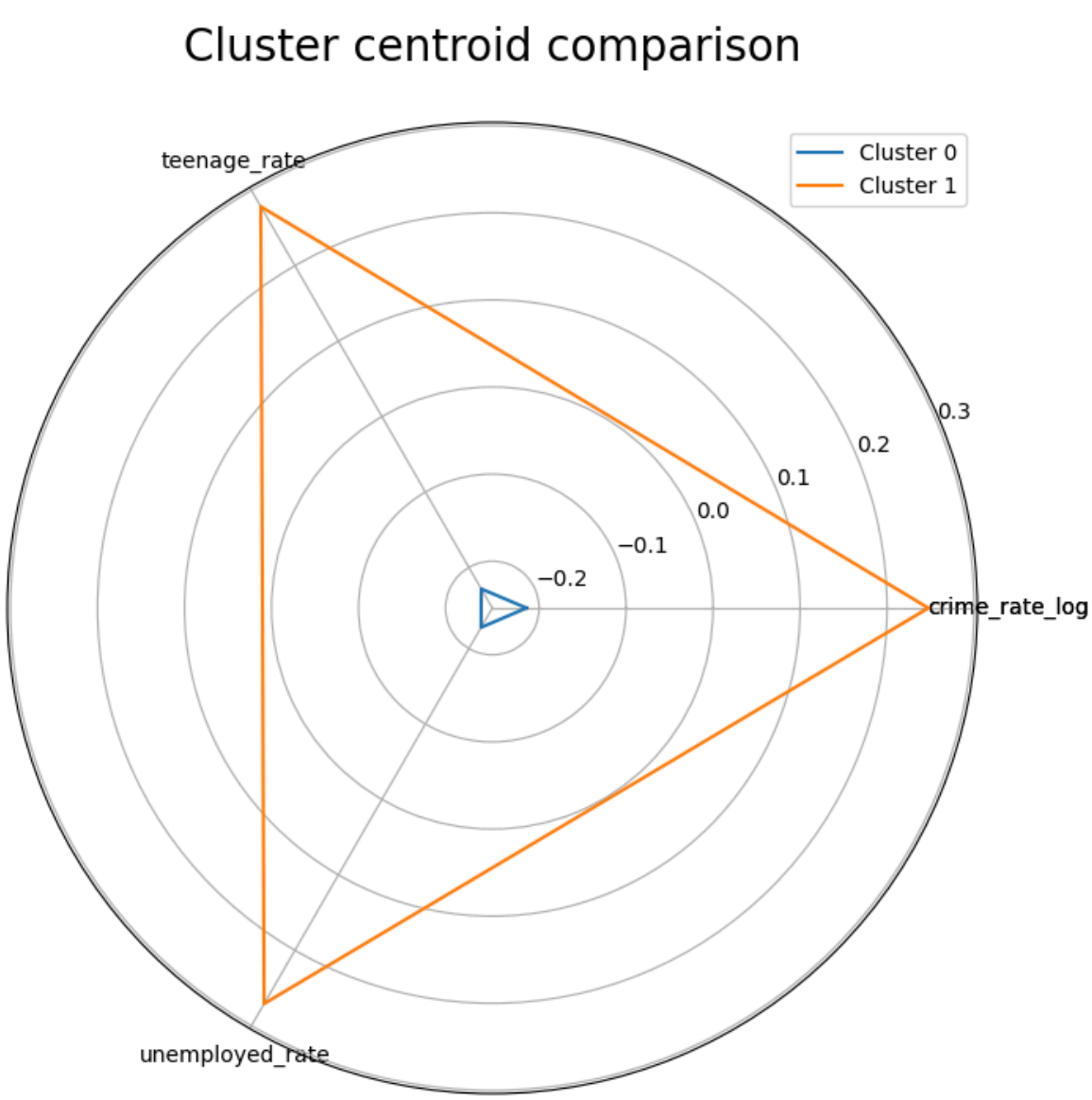
K = 2 is a good choice

In [ ]:
```
k_cluster = 2
random_seed = 1
kmeans_method = KMeans(n_clusters=k_cluster,random_state=random_seed)
kmeans_method.fit(normed)
mapping_clusters(kmeans_method.labels_);
```

```
c:\Users\17197\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn
\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init` will chang
e from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  super()._check_params_vs_input(X, default_n_init=10)
```



In [ ]:
```
df_cluster_centroid = pd.DataFrame(kmeans_method.cluster_centers_, columns=norme
radar_plot_cluster_centroids(df_cluster_centroid)
```



The only measure that can help is the Silhouette Score, which calculates how close points are on average to points their clustered with, relative to points they are not clustered with.

The `scikit-learn` algorithm (docs) for Silhouette Score simply takes the data and the generated labels. A score closer to one indicates strong clustering, negative scores indicate poor clustering.

In [ ]:
```
from sklearn import metrics
metrics.silhouette_score(normed, kmeans_method.labels_)
```

Out[ ]:
```
0.3182008649011523
```

In [ ]:
```
df2 = lsoa_merge1[['LSOA21CD','public_order_crime_rate_log','teenage_rate','unem
df2.set_index('LSOA21CD', inplace=True)
```

In [ ]:
```
normed1 = df2.copy()
for c in df2.columns.values:
    normed1[c] = rs.fit_transform(df2[c].values.reshape(-1,1))
    print("The range of {} is [{}, {}]".format(c, normed1[c].min(), normed1[c].m
normed1.head()
```

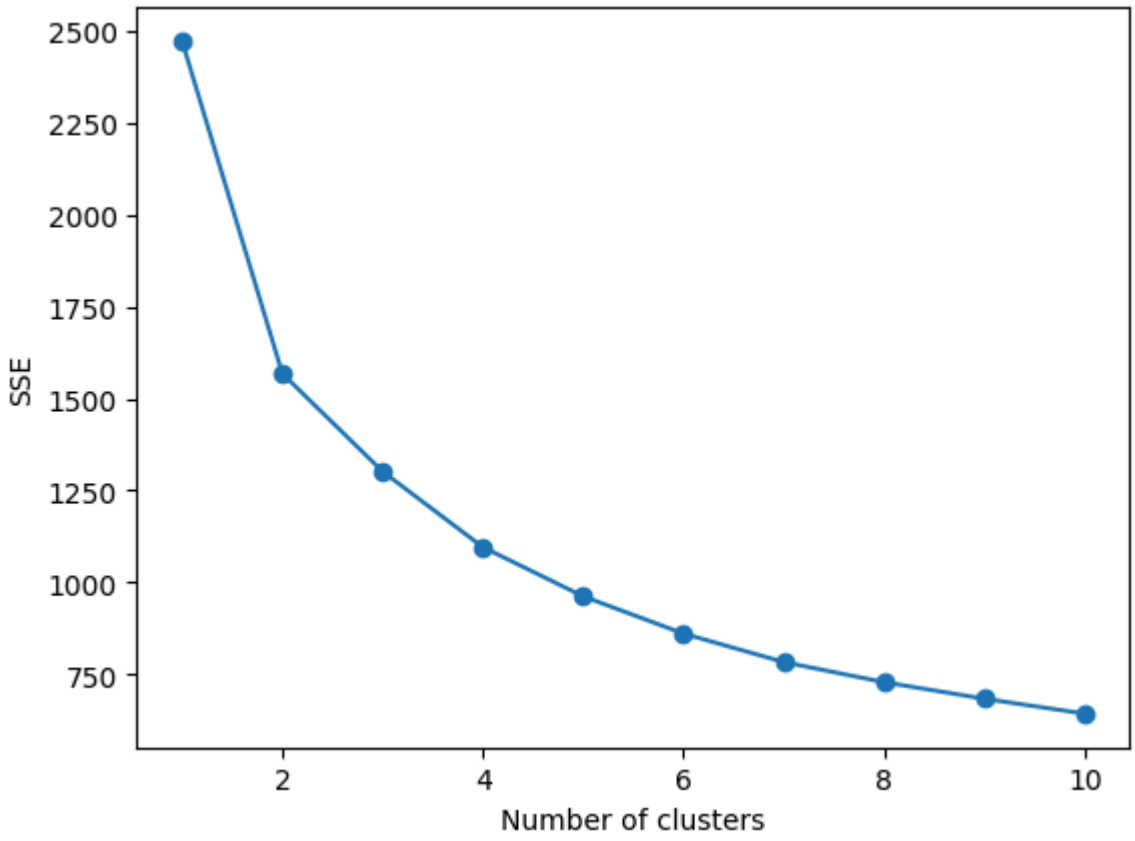
```
The range of public_order_crime_rate_log is [-1.792018133903522, 1.87059858137675
7]
The range of teenage_rate is [-0.7901207284399858, 3.571072852694266]
The range of unemployed_rate is [-0.9648933603301447, 2.263373197800955]
```

Out[ ]:

LSOA21CD	public_order_crime_rate_log	teenage_rate	unemployed_rate
E01000001	-0.029281	-0.287620	-0.397777
E01000002	0.199325	0.103626	-0.432994
E01000003	-0.296891	0.040202	0.164711
E01000005	0.855417	0.816263	0.628308
E01000006	-0.241198	0.171331	0.032275

In [ ]:

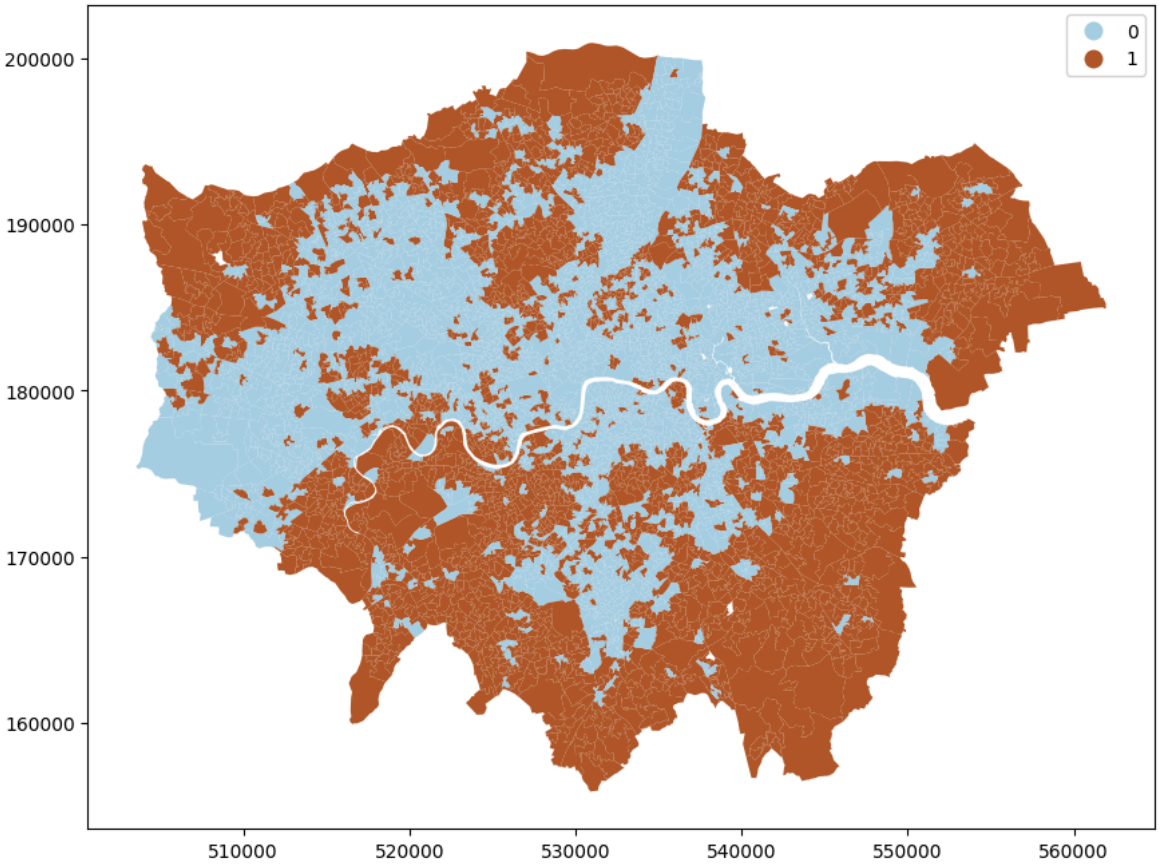
```python
# calculate SSE for a range of number of cluster
list_SSE = []
min_k = 1
max_k = 10
range_k = range(min_k, max_k+1)
for i in range_k:
    km = KMeans(
        n_clusters=i, init='random',
        n_init=10, max_iter=300,
        tol=1e-04, random_state=0
    )
    km.fit(normed1)
    # inertia is a concept from physics. Roughly it means SSE of clustering.
    list_SSE.append(km.inertia_)

# plot
plt.plot(range_k, list_SSE, marker='o')
plt.xlabel('Number of clusters')
plt.ylabel('SSE')
plt.show()
```



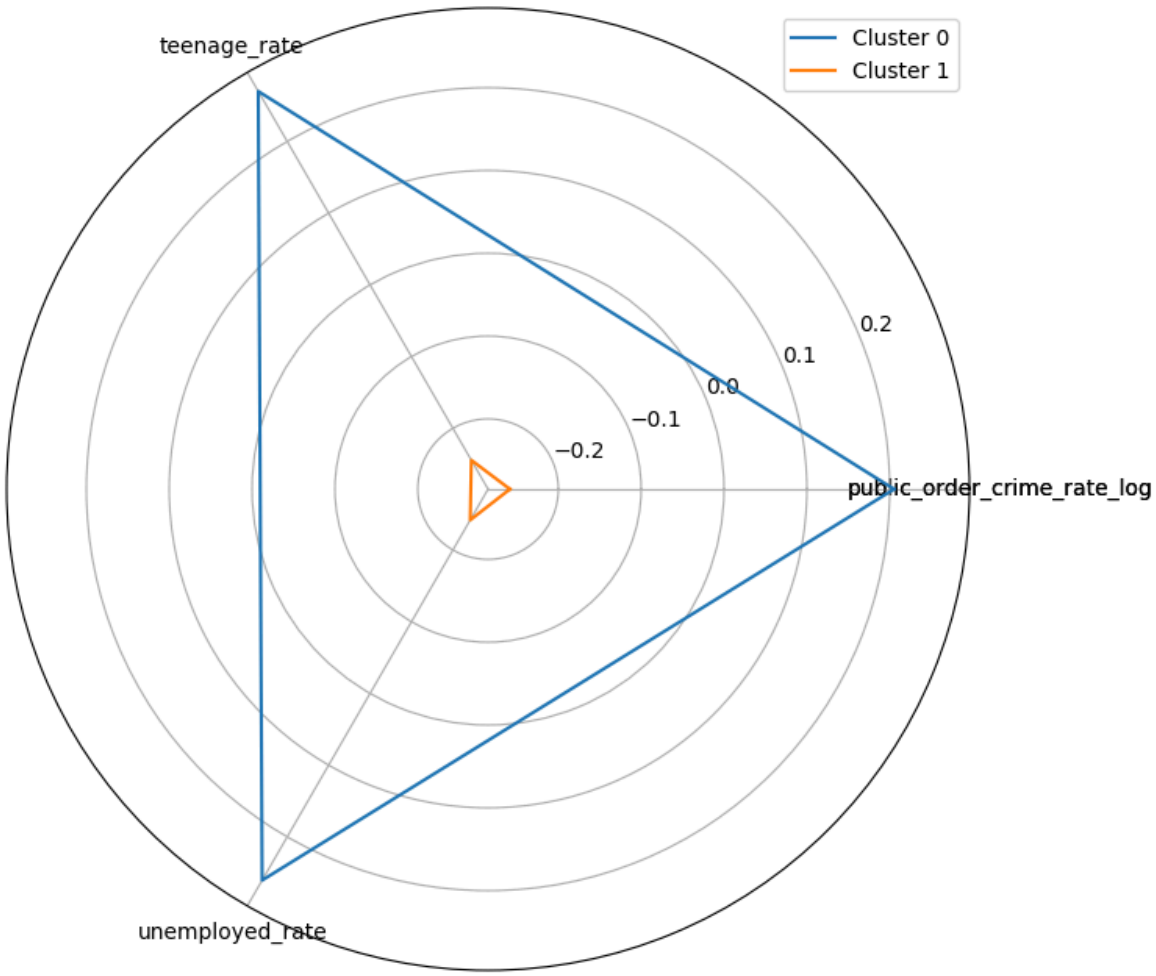
K = 2 is a good choice

In [ ]:
```
k_cluster = 2
random_seed = 1
kmeans_method = KMeans(n_clusters=k_cluster,random_state=random_seed)
kmeans_method.fit(normed1)
mapping_clusters(kmeans_method.labels_);
```

```
c:\Users\17197\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn
\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init` will chang
e from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  super()._check_params_vs_input(X, default_n_init=10)
```



In [ ]:
```
df_cluster_centroid = pd.DataFrame(kmeans_method.cluster_centers_, columns=norme
radar_plot_cluster_centroids(df_cluster_centroid)
```



In [ ]:
```
metrics.silhouette_score(normed1, kmeans_method.labels_)
```

Out[ ]:
```
0.32430523245049947
```

In [ ]:
```
df3 = lsoa_merge1[['LSOA21CD','violent_crime_rate_log','teenage_rate','unemploye
df3.set_index('LSOA21CD', inplace=True)
```

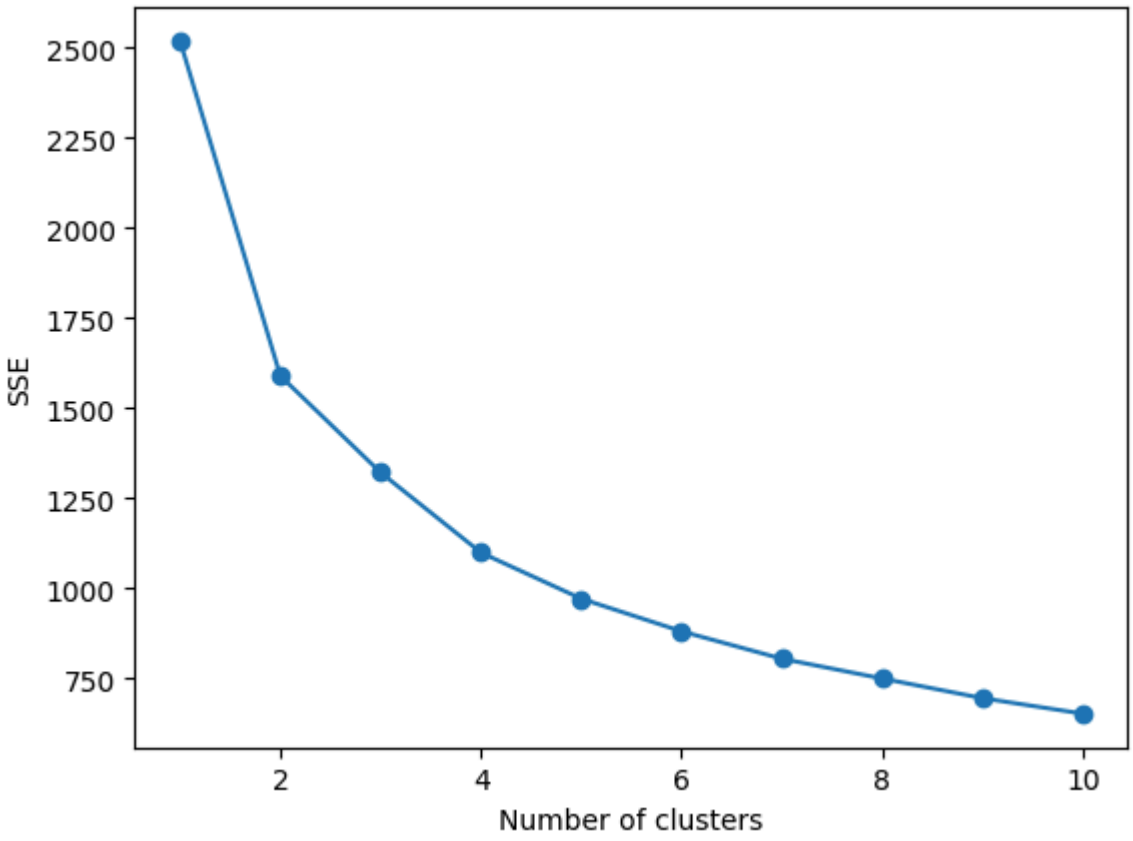
In [ ]:
```
normed2 = df3.copy()
for c in df3.columns.values:
    normed2[c] = rs.fit_transform(df3[c].values.reshape(-1,1))
    print("The range of {} is [{}, {}]".format(c, normed2[c].min(), normed2[c].m
normed2.head()
```

```
The range of violent_crime_rate_log is [-2.2163622311197204, 2.18713893742722]
The range of teenage_rate is [-0.7901207284399858, 3.571072852694266]
The range of unemployed_rate is [-0.9648933603301447, 2.263373197800955]
```

Out[ ]:

LSOA21CD	violent_crime_rate_log	teenage_rate	unemployed_rate
E01000001	-0.386417	-0.287620	-0.397777
E01000002	0.323687	0.103626	-0.432994
E01000003	-0.245605	0.040202	0.164711
E01000005	1.067417	0.816263	0.628308
E01000006	-0.060183	0.171331	0.032275

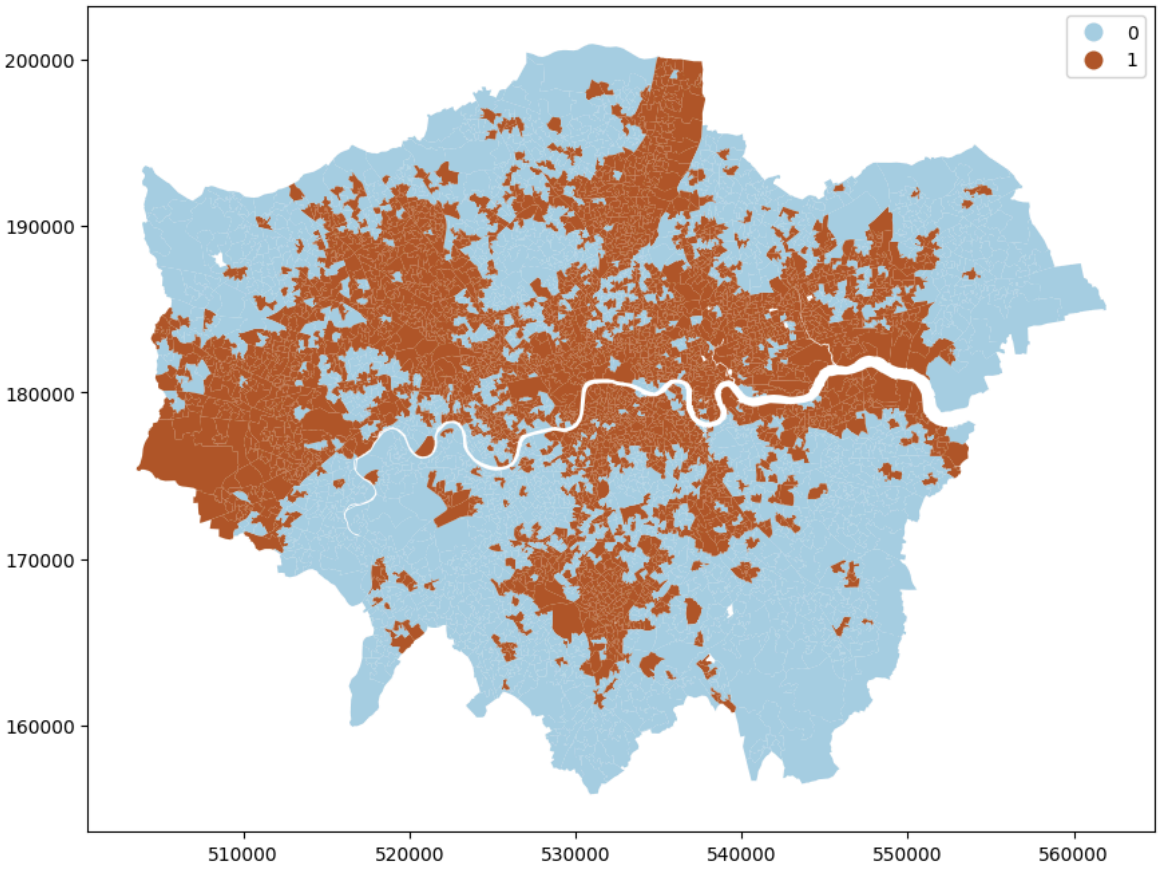
In [ ]:
```
# calculate SSE for a range of number of cluster
list_SSE = []
min_k = 1
max_k = 10
range_k = range(min_k, max_k+1)
for i in range_k:
    km = KMeans(
        n_clusters=i, init='random',
        n_init=10, max_iter=300,
        tol=1e-04, random_state=0
    )
    km.fit(normed2)
    # inertia is a concept from physics. Roughly it means SSE of clustering.
    list_SSE.append(km.inertia_)

# plot
plt.plot(range_k, list_SSE, marker='o')
plt.xlabel('Number of clusters')
plt.ylabel('SSE')
plt.show()
```



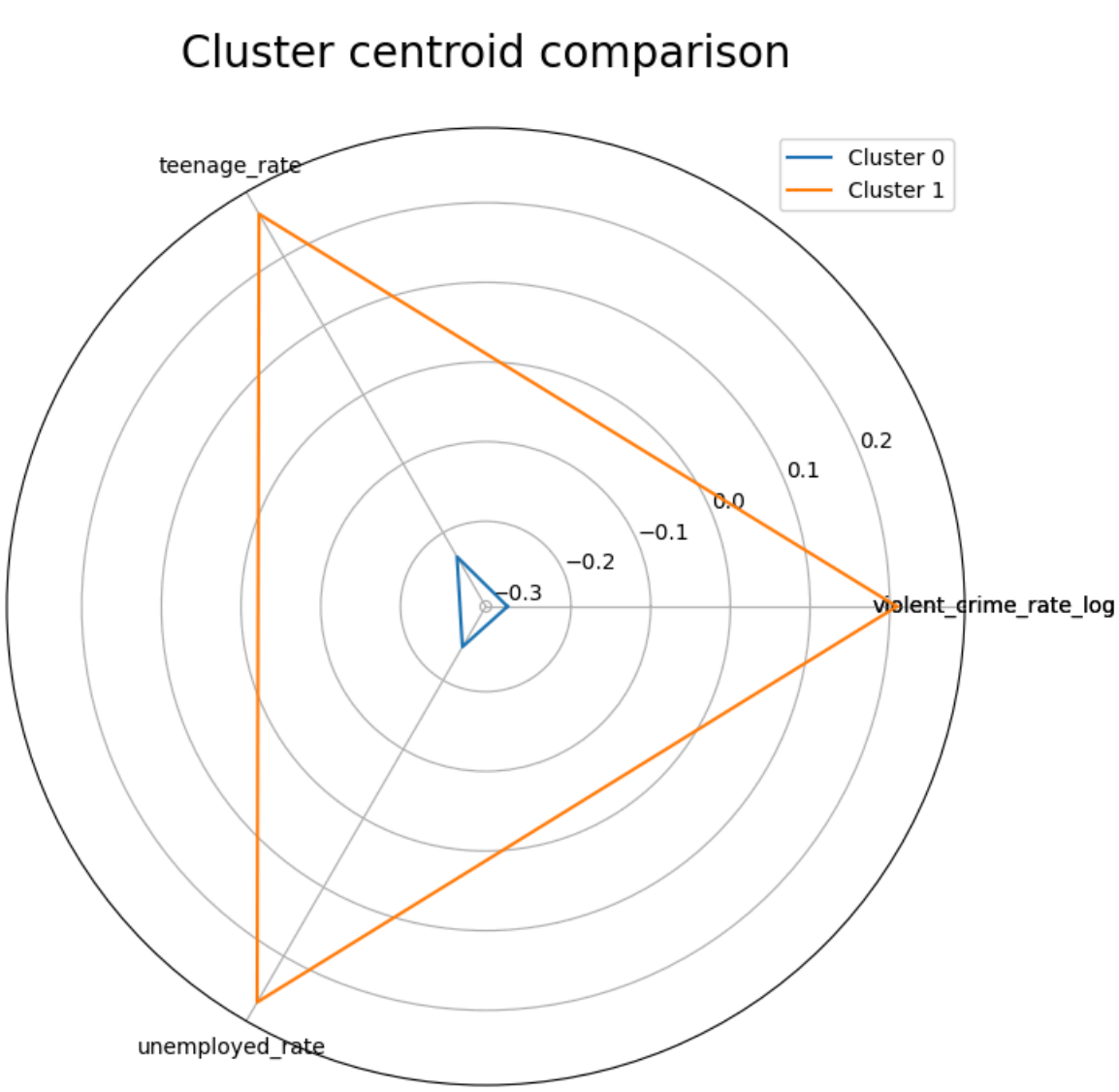
K = 2 is a good choice

In [ ]:
```
k_cluster = 2
random_seed = 1
kmeans_method = KMeans(n_clusters=k_cluster,random_state=random_seed)
kmeans_method.fit(normed2)
mapping_clusters(kmeans_method.labels_);
```

```
c:\Users\17197\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn
\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init` will chang
e from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  super()._check_params_vs_input(X, default_n_init=10)
```



In [ ]:
```
df_cluster_centroid = pd.DataFrame(kmeans_method.cluster_centers_, columns=norme
radar_plot_cluster_centroids(df_cluster_centroid)
```



In [ ]:
```
metrics.silhouette_score(normed2, kmeans_method.labels_)
```

Out[ ]: 0.3280668341764684

In [ ]:
```
df4 = lsoa_merge1[['LSOA21CD','property_crime_rate_log','teenage_rate','unemploy
df4.set_index('LSOA21CD', inplace=True)
```

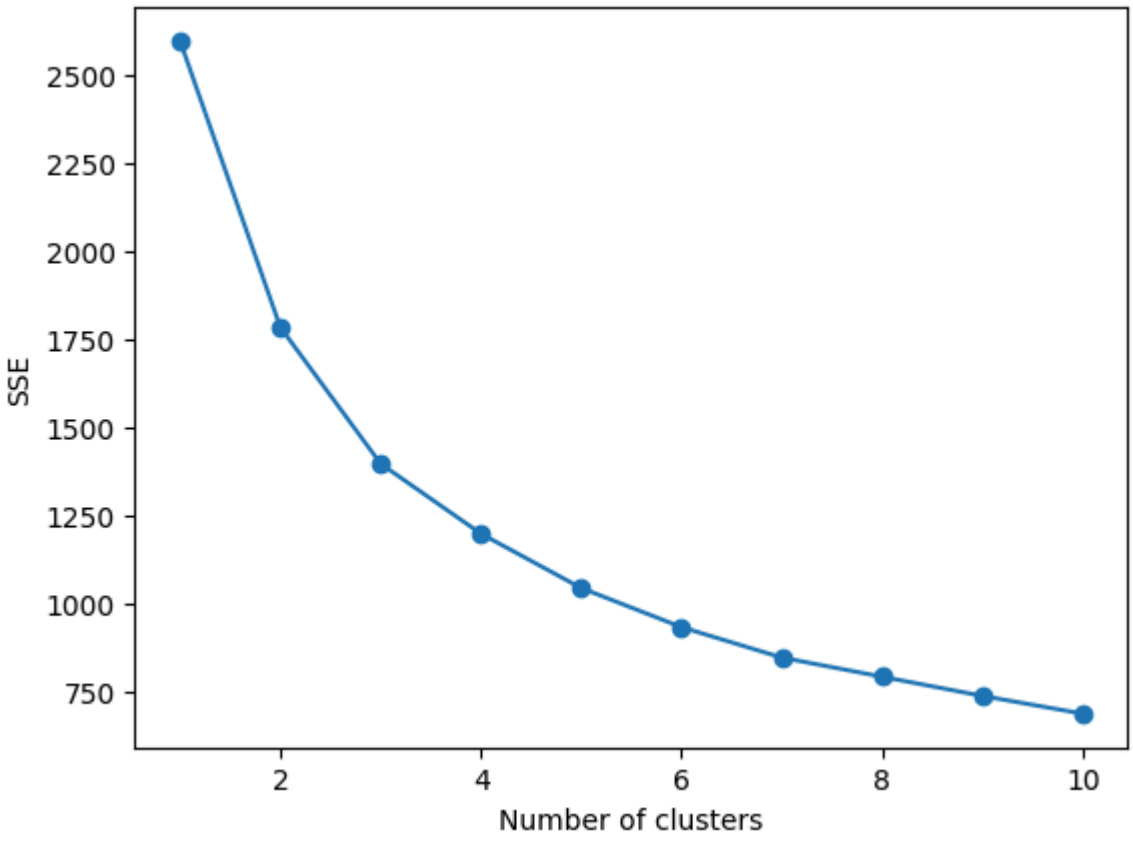
In [ ]:
```
normed3 = df4.copy()
for c in df4.columns.values:
    normed3[c] = rs.fit_transform(df4[c].values.reshape(-1,1))
    print("The range of {} is [{}, {}]".format(c, normed3[c].min(), normed3[c].m
normed3.head()
```

```
The range of property_crime_rate_log is [-2.0335012999130795, 2.944430182367487]
The range of teenage_rate is [-0.7901207284399858, 3.571072852694266]
The range of unemployed_rate is [-0.9648933603301447, 2.263373197800955]
```

Out[ ]:

LSOA21CD	property_crime_rate_log	teenage_rate	unemployed_rate
E01000001	0.411857	-0.287620	-0.397777
E01000002	1.076175	0.103626	-0.432994
E01000003	-0.049965	0.040202	0.164711
E01000005	1.422847	0.816263	0.628308
E01000006	-0.481310	0.171331	0.032275

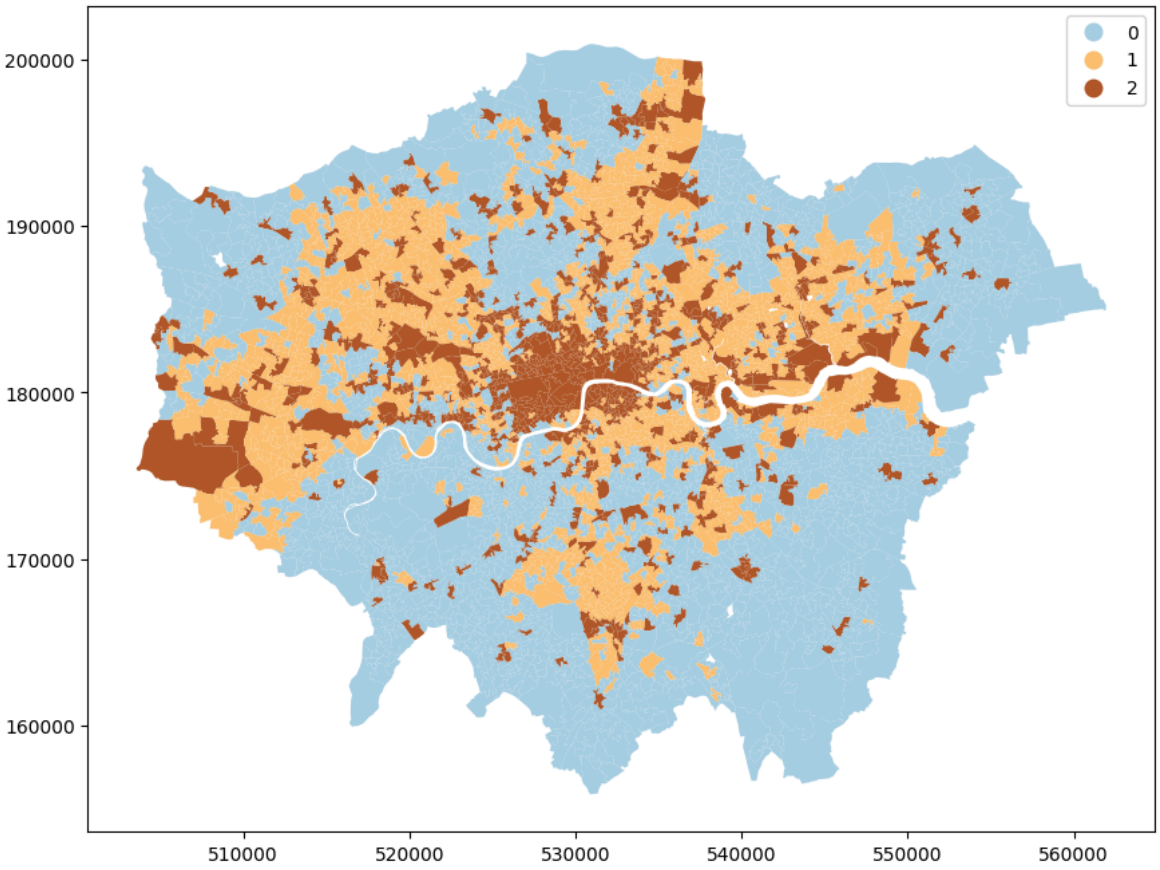
In [ ]:
```
# calculate SSE for a range of number of cluster
list_SSE = []
min_k = 1
max_k = 10
range_k = range(min_k, max_k+1)
for i in range_k:
    km = KMeans(
        n_clusters=i, init='random',
        n_init=10, max_iter=300,
        tol=1e-04, random_state=0
    )
    km.fit(normed3)
    # inertia is a concept from physics. Roughly it means SSE of clustering.
    list_SSE.append(km.inertia_)

# plot
plt.plot(range_k, list_SSE, marker='o')
plt.xlabel('Number of clusters')
plt.ylabel('SSE')
plt.show()
```



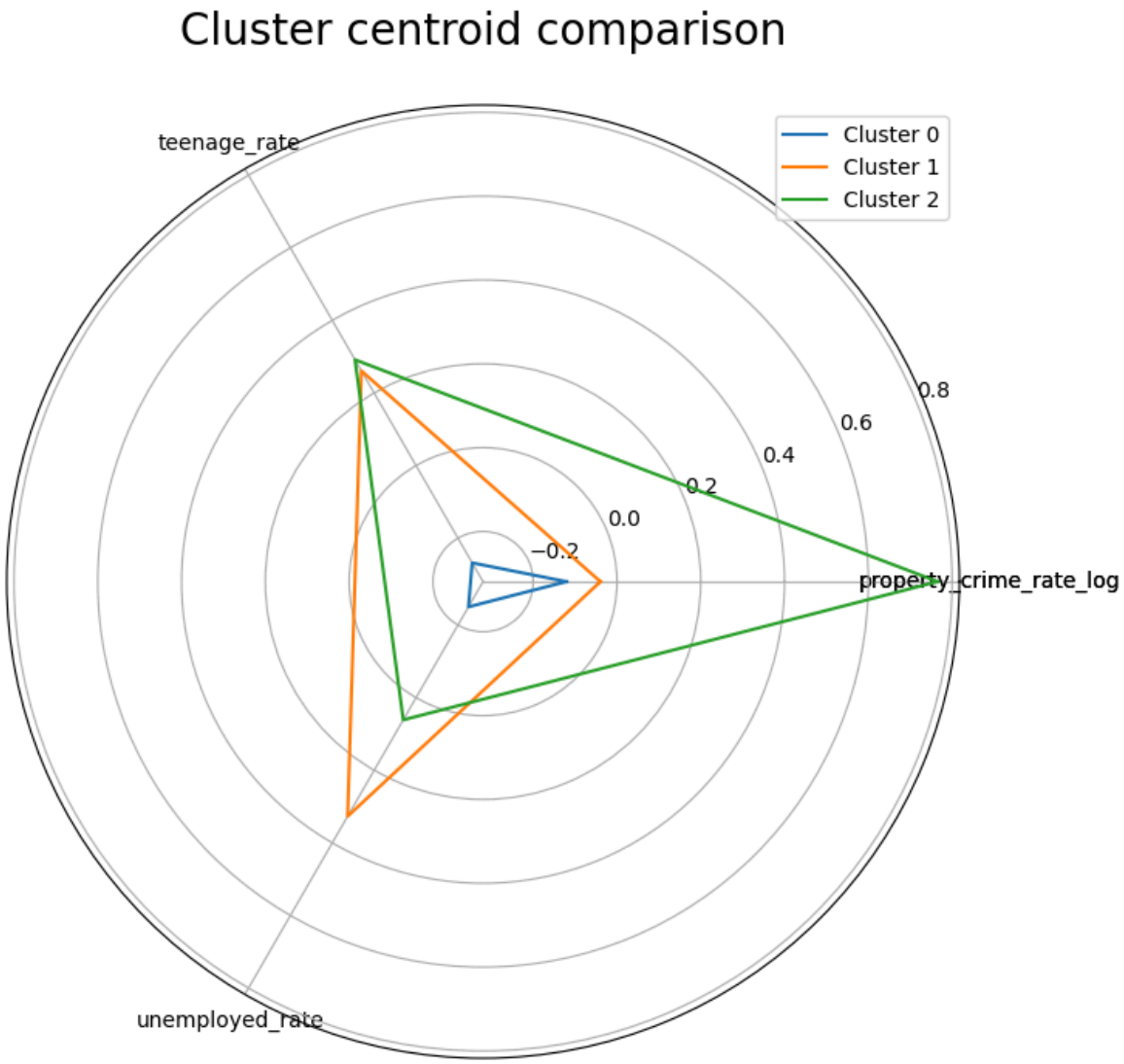
K = 3 is a good choice

In [ ]:
```
k_cluster = 3
random_seed = 1
kmeans_method = KMeans(n_clusters=k_cluster,random_state=random_seed)
kmeans_method.fit(normed3)
mapping_clusters(kmeans_method.labels_);
```

```
c:\Users\17197\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn
\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init` will chang
e from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  super()._check_params_vs_input(X, default_n_init=10)
```



In [ ]:
```
df_cluster_centroid = pd.DataFrame(kmeans_method.cluster_centers_, columns=norme
radar_plot_cluster_centroids(df_cluster_centroid)
```



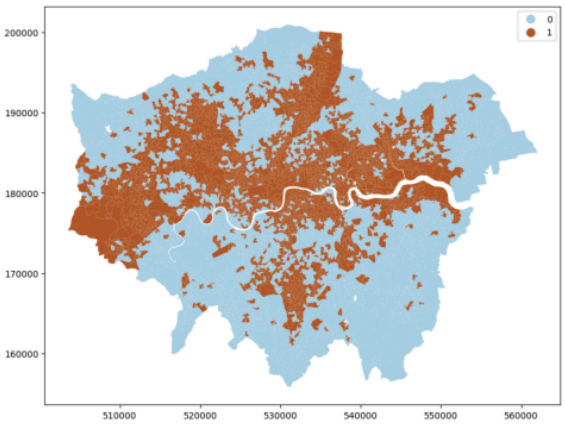
In [ ]:
```
metrics.silhouette_score(normed3, kmeans_method.labels_)
```

Out[ ]:
```
0.30193016900250047
```

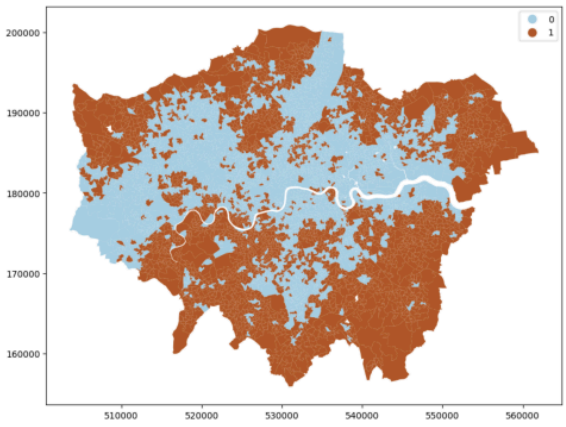
Results

In [ ]:
```
import requests
from PIL import Image
from io import BytesIO



def load_image_from_url(url):
    response = requests.get(url)
    img = Image.open(BytesIO(response.content))
    return np.array(img)

# URLs of the images
urls = [
    'https://github.com/zzzzzoy/data_sci/blob/main/output/output1.1.png?raw=true
    'https://github.com/zzzzzoy/data_sci/blob/main/output/output2.1.png?raw=true
    'https://github.com/zzzzzoy/data_sci/blob/main/output/output3.1.png?raw=true
    'https://github.com/zzzzzoy/data_sci/blob/main/output/output4.1.png?raw=true
]
```

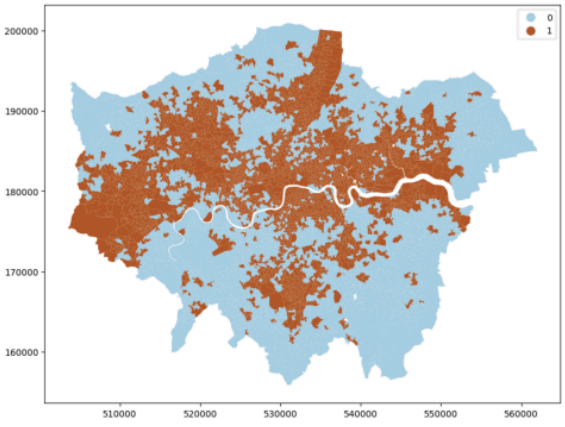
```
# Load images
images = [load_image_from_url(url) for url in urls]

# Create a figure and a 2x2 grid of subplots
fig, axs = plt.subplots(2, 2, figsize=(10, 10))

# Titles for each image
titles = ["crime_rate_log", "public_order_crime_rate_log", "violent_crime_rate_l

# Show each image in a subplot and add a name below
for img, ax, title in zip(images, axs.flat, titles):
    ax.imshow(img)
    ax.axis('off')  # Hide axis
    ax.text(0.5, -0.1, title, transform=ax.transAxes,
            ha='center', va='top', fontsize=12)

# Adjust the spacing between subplots
plt.tight_layout()
plt.show()
```



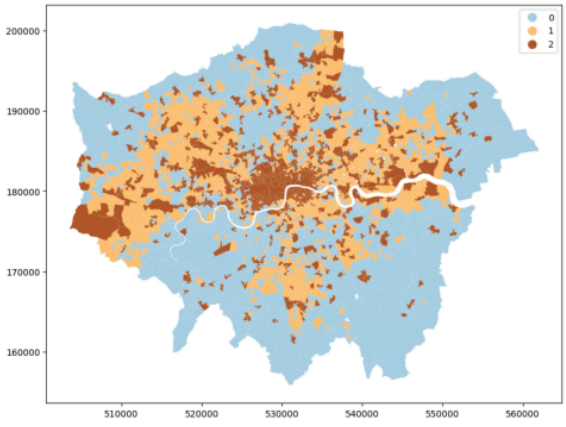
crime_rate_log



public_order_crime_rate_log



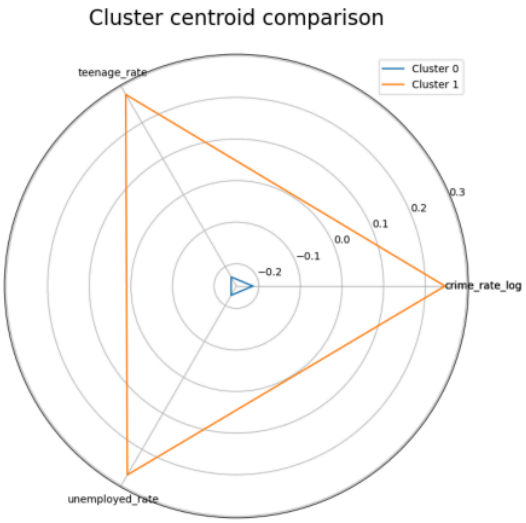
violent_crime_rate_log

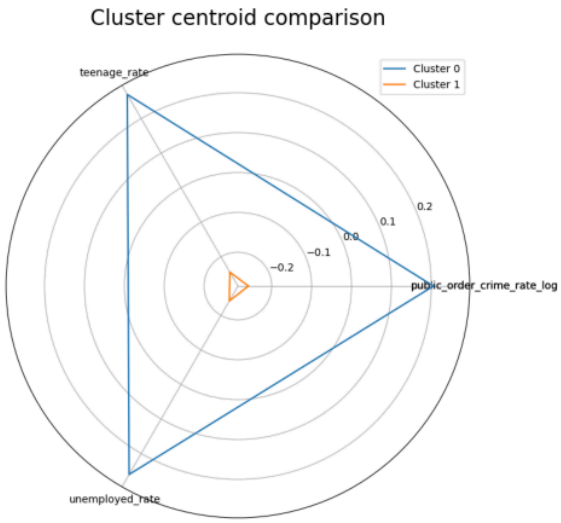


property_crime_rate_log

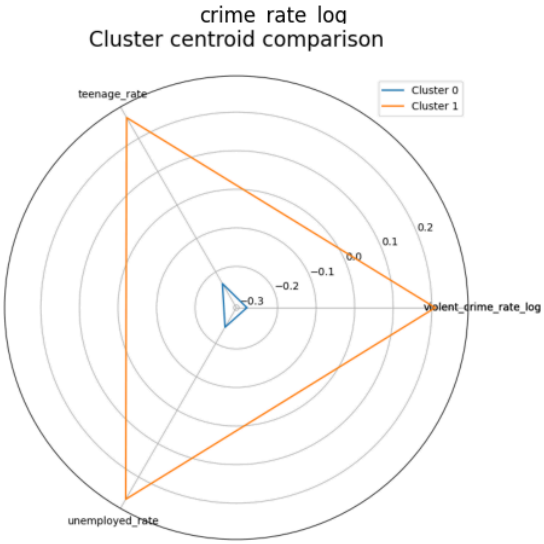
In [ ]:
```
urls = [
    'https://github.com/zzzzzoy/data_sci/blob/main/output/output1.2.png?raw=true
    'https://github.com/zzzzzoy/data_sci/blob/main/output/output2.2.png?raw=true
    'https://github.com/zzzzzoy/data_sci/blob/main/output/output3.2.png?raw=true
    'https://github.com/zzzzzoy/data_sci/blob/main/output/output4.2.png?raw=true
]
```

```
# Load images
images = [load_image_from_url(url) for url in urls]

# Create a figure and a 2x2 grid of subplots
fig, axs = plt.subplots(2, 2, figsize=(10, 10))

# Titles for each image
titles = ["crime_rate_log", "public_order_crime_rate_log", "violent_crime_rate_l

# Show each image in a subplot and add a name below
for img, ax, title in zip(images, axs.flat, titles):
    ax.imshow(img)
    ax.axis('off')  # Hide axis
    ax.text(0.5, -0.1, title, transform=ax.transAxes,
            ha='center', va='top', fontsize=12)

# Adjust the spacing between subplots
plt.tight_layout()
plt.show()
```



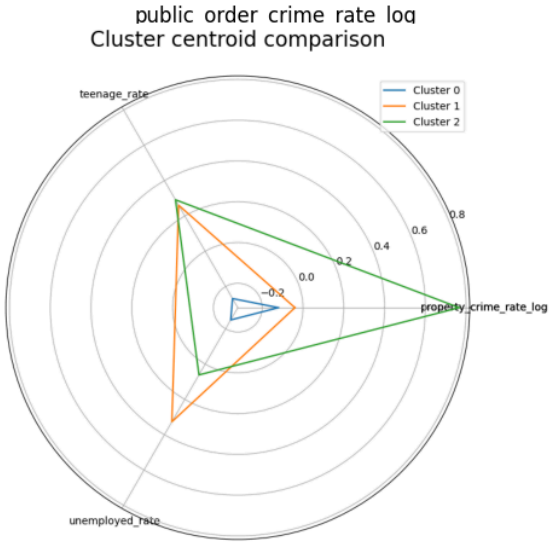


crime_rate_log



violent_crime_rate_log

public_order_crime_rate_log



property_crime_rate_log

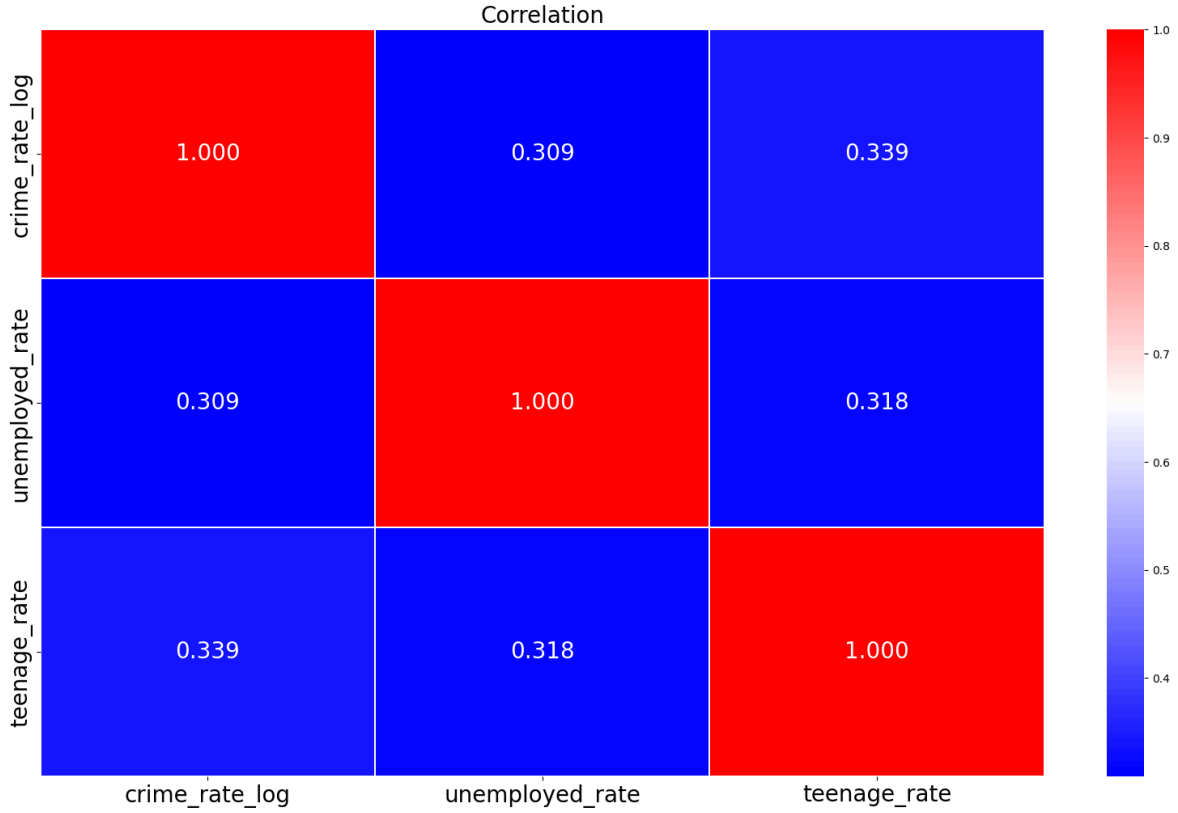
Discussion and Conclusion

Based on the spatial distribution of crime rates from the four graphs, the distribution of clusters with higher crime rates is very similar across all types. Therefore, I believe that there is no significant spatial variation in the types of crime; each LSOA does not have a clear tendency towards any particular type of crime. From the first three radar plots, it is not difficult to conclude that clusters with high crime rates also have a high proportion of youth and unemployment rates. This makes me wonder if there is a positive relationship between unemployment rates, the proportion of youth, and crime rates. Next, I will use linear regression to test this hypothesis.

In [ ]:
```
name_list = ['LSOA21CD','crime_rate_log' ,'unemployed_rate', 'teenage_rate']
df = lsoa_merge1[name_list]
# make the correlation matrix
df_numeric = df.loc[:, 'crime_rate_log':'teenage_rate']
```

In [ ]:
```
df_cormatrix = df_numeric.corr()# add a list to record the correlation coefficie
corcoef_list1 = df_cormatrix['crime_rate_log'].tolist()
```

In [ ]:
```
fig, axes = plt.subplots(figsize = (20,12))
sns.heatmap(df_cormatrix, cmap ='bwr', annot = True, fmt='.3f', linewidths = 0.1
axes.tick_params(labelsize = 20)
axes.set_title('Correlation', fontsize = 20)
plt.show()
```



In [ ]:
```
df.dropna(axis=0, inplace=True)
```

```
C:\Users\17197\AppData\Local\Temp\ipykernel_44720\3603100996.py:1: SettingWithCop
yWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  df.dropna(axis=0, inplace=True)
```

In [ ]:
```
selected_features = ['unemployed_rate', 'teenage_rate']
OSLmodel1 = sm.OLS(endog=df[['crime_rate_log']], exog=sm.add_constant(df[selecte

test_rainbow1 = statsmodels.stats.diagnostic.linear_rainbow(OSLmodel1)
# This function returns a tuple consisting of two values: the test statistic bas
# Note that these two values are not named. Therefore, you need to know the orde
print("The p value of the rainbow test: {:.4f}".format(test_rainbow1[1]))

test_dw1 = statsmodels.stats.stattools.durbin_watson(OSLmodel1.resid)
print("Durbin-Watson test statistic is: {:.4f}".format(test_dw1))
test_JB1 = statsmodels.stats.stattools.jarque_bera(OSLmodel1.resid)
print("The p value of the Jarque Bera test: {:.4f}".format(test_JB1[1]))
a= statsmodels.stats.diagnostic.het_goldfeldquandt(OSLmodel1.model.endog, OSLmod
print("The p value of the Goldfeld-Quandt homoskedasticity test: {:.4f}".format(
OSLmodel1.summary()
```

```
The p value of the rainbow test: 0.0000
Durbin-Watson test statistic is: 1.5979
The p value of the Jarque Bera test: 0.0000
The p value of the Goldfeld-Quandt homoskedasticity test: 0.0000
```

Out[ ]:

OLS Regression Results

Dep. Variable:	crime_rate_log	**R-squared:**	0.160
Model:	OLS	**Adj. R-squared:**	0.159
Method:	Least Squares	**F-statistic:**	472.6
Date:	Mon, 22 Apr 2024	**Prob (F-statistic):**	1.20e-188
Time:	16:07:59	**Log-Likelihood:**	-4832.8
No. Observations:	4979	**AIC:**	9672.
Df Residuals:	4976	**BIC:**	9691.
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-3.3477	0.031	-109.637	0.000	-3.408	-3.288
unemployed_rate	11.3141	0.693	16.321	0.000	9.955	12.673
teenage_rate	18.3107	0.938	19.513	0.000	16.471	20.150

Omnibus:	619.982	**Durbin-Watson:**	1.598
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	1757.552
Skew:	0.673	**Prob(JB):**	0.00
Kurtosis:	5.581	**Cond. No.**	109.

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Although this is not an optimal model, based on the Rainbow Test, the model may suffer from non-linearity. From the Jarque-Bera Test, the model does not follow a normal distribution. From the Goldfeld-Quandt Test, the variance of the errors is not constant across all levels of the independent variables. However, the p-values for youth rate and unemployment rate are extremely low, and their coefficients are positive, indicating a definite positive correlation between youth rate, unemployment rate, and crime rate.

However, when we notice the fourth radar chart, we find something strange. Although cluster 2 has a property crime rate significantly higher than that of cluster 1, the proportion of youth in both clusters is almost the same, and even the unemployment rate in cluster 2 is lower than in cluster 1. Does this contradict our previous conclusions? I believe this is due to human mobility. To explain this result, let's first go back to the data itself. The crime data records the location of the crime, which means that the criminal does not necessarily reside in the LSOA where the crime took place. Meanwhile, the attributes of an LSOA recorded

include the population residing in that LSOA, so the unemployment rate describes the unemployment rate of residents within that LSOA, not the crime rate. This needs to take into account people living outside the LSOA who come to commit crimes in that LSOA, which causes some areas to have a high calculated crime rate while having a low unemployment rate.

I will use the passenger flow of the London Underground to support my point.

In [ ]:
```
G = nx.read_graphml(cache_data('https://github.com/zzzzzoy/data_sci/blob/main/da
```

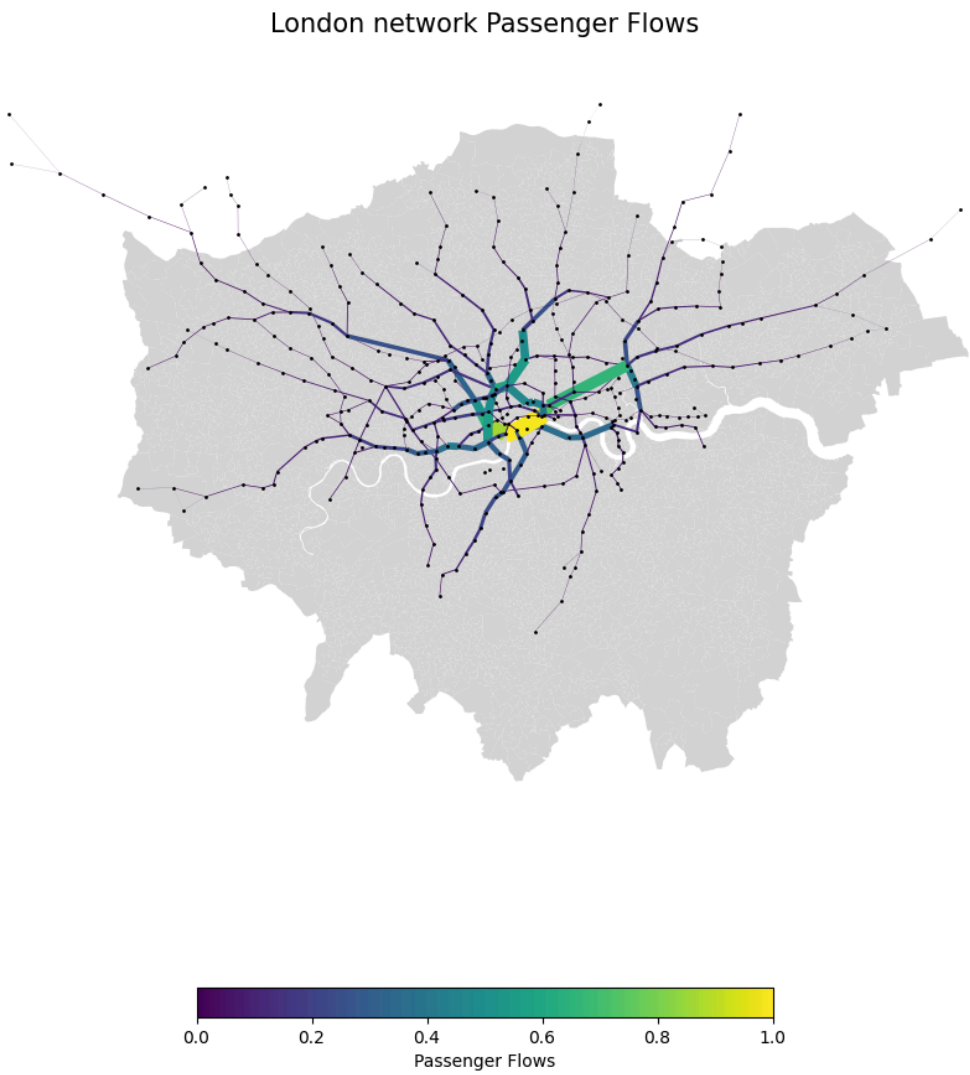
```
Found data\london1.graph locally!
```

In [ ]:
```
#since coords tuples are stored as string, need to convert them back to tuples u
for node in G.nodes():
    G.nodes[node]['coords'] = eval(G.nodes[node]['coords'])
```

In [ ]:
```
pos = nx.get_node_attributes(G, 'coords')
```

In [ ]:
```
flows = nx.get_edge_attributes(G,'flows')
```

In [ ]:
```
flows_values = flows.values()
flow_color=[(i[2]['flows']/max(flows_values)) for i in G.edges(data=True)]
flow_width=[(i[2]['flows']/max(flows_values)*10) for i in G.edges(data=True)]
# Plot graph
fig, ax = plt.subplots(figsize=(12,12))

lsoa.plot(ax=ax, color='lightgrey')
edg=nx.draw_networkx_edges(G, pos,edge_color=flow_color, width=flow_width)

nx.draw_networkx_nodes(G,
        pos = pos,
        node_color= 'black',
        node_size= 1)


plt.colorbar(edg,label="Passenger Flows",orientation="horizontal", shrink=0.5)
plt.axis("off")
plt.title("London network Passenger Flows",fontsize=15)
plt.show()
```



From this chart, we can see that areas with high passenger flow overlap significantly with the location of cluster 2. Therefore, this might explain why cluster 2 has a high crime rate and a low unemployment rate, which could be attributed to the influx of people from outside the area.

Therefore, in all, to reduce the unemployment rate, my suggestion is to lower the unemployment rate, educate the youth on crime prevention, and also pay attention to the impact of human flows on the crime rate.

Reference

K. P. Sinaga and M. -S. Yang (2020) 'Unsupervised K-Means Clustering Algorithm', *IEEE Access*, 8, pp. 80716–80727. Available at: https://doi.org/10.1109/ACCESS.2020.2988796.

Edmark, K. (2005) 'Unemployment and crime: Is there a connection?', *Scandinavian Journal of Economics*, 107(2), pp. 353–373.

Xu, Y.-H., Pennington-Gray, L. and Kim, J. (2019) 'The Sharing Economy: A Geographically Weighted Regression Approach to Examine Crime and the Shared

Lodging Sector', *Journal of Travel Research*, 58(7), pp. 1193–1208. Available at: https://doi.org/10.1177/0047287518797197.

Ahmed, M., Seraj, R. and Islam, S.M. (2020) 'The k-means Algorithm: A Comprehensive Survey and Performance Evaluation', *Electronics*, 9(8). Available at: https://doi.org/10.3390/electronics9081295.

Changyong, F. et al. (2014) 'Log-transformation and its implications for data analysis', *Shanghai archives of psychiatry*, 26(2), p. 105.

Raphael, S. and Winter-Ebmer, R. (2001) 'Identifying the effect of unemployment on crime', *The journal of law and economics*, 44(1), pp. 259–283.

Home - Office for National Statistics (no date). Available at: https://www.ons.gov.uk/ (Accessed: 22 April 2024).

Murray, A.T. and Grubesic, T.H. (2013) 'Exploring Spatial Patterns of Crime Using Non-hierarchical Cluster Analysis', in M. Leitner (ed.) *Crime Modeling and Mapping Using Geospatial Technologies*. Dordrecht: Springer Netherlands, pp. 105–124. Available at: https://doi.org/10.1007/978-94-007-4997-9_5.

Data downloads | data.police.uk (no date). Available at: https://data.police.uk/data/ (Accessed: 22 April 2024).

Flatley, J. (2017) 'Crime in England and Wales: year ending Sept 2016', *Office for National Statistics-Crime Survey of England and Wales (CSEW), London* [Preprint].

Alkhaibari, A.A. and Chung, P.-T. (2017) 'Cluster analysis for reducing city crime rates', in. *2017 IEEE Long Island Systems, Applications and Technology Conference (LISAT)*, IEEE, pp. 1–6.

Govender, P. and Sivakumar, V. (2020) 'Application of k-means and hierarchical clustering techniques for analysis of air pollution: A review (1980–2019)', *Atmospheric Pollution Research*, 11(1), pp. 40–56. Available at: https://doi.org/10.1016/j.apr.2019.09.009.

Hirschi, T. and Gottfredson, M. (1983) 'Age and the Explanation of Crime', *American Journal of Sociology*, 89(3), pp. 552–584

Steffensmeier, D.J. et al. (1989) 'Age and the distribution of crime', *American journal of Sociology*, 94(4), pp. 803–831

About | data.police.uk (no date). Available at: https://data.police.uk/about/ (Accessed: 22 April 2024).