**Code for London data analysis**

|  |
| --- |
| import pandas as pd  import matplotlib.pyplot as plt  import folium  import os, re  from sklearn.preprocessing import StandardScaler  from sklearn.preprocessing import normalize  from IPython.display import IFrame  from sklearn.cluster import AgglomerativeClustering  import scipy.cluster.hierarchy as shc  %matplotlib inline  import warnings  warnings.filterwarnings('ignore') |

|  |
| --- |
| path\_to\_data = '/content/sample\_data/london\_crime.csv'  cd = os.path.dirname(os.path.abspath(path\_to\_data))  i=0  columns = range(1,100)  dfList = []  for root, dirs, files in os.walk(cd):  for fname in files:  if re.match("^.\*.csv$", fname):  frame = pd.read\_csv(os.path.join(root, fname))  frame['key'] = "file{}".format(i)  dfList.append(frame)  i += 1  dataset = pd.concat(dfList) |

|  |
| --- |
| dataset.head() |

|  |
| --- |
| print(dataset.shape) |

|  |
| --- |
| crime= 'london\_crime.csv'  dataset.to\_csv(crime, index=False) |

|  |
| --- |
| data = pd.read\_csv(crime) |

|  |
| --- |
| data['Crime type'].value\_counts() |

|  |
| --- |
| data['LSOA name'].value\_counts() |

|  |
| --- |
| data['Month'].value\_counts() |

|  |
| --- |
| data['town'] = data['LSOA name'].str.split(' ').str[0] |

|  |
| --- |
| data.head() |

|  |
| --- |
| towns = ['City']  filtered\_data = data[data.town.str.contains('|'.join(towns), na=False)]  filtered\_data.head() |

|  |
| --- |
| filtered\_data['LSOA code'].value\_counts().nlargest(10) |

|  |
| --- |
| filtered\_important\_data = filtered\_data[['LSOA code','Crime type']]  filtered\_important\_data = pd.get\_dummies(filtered\_important\_data, columns=['Crime type'])  clustering\_data = filtered\_important\_data.groupby(['LSOA code']).agg(  {'Crime type\_Anti-social behaviour':'sum',  'Crime type\_Bicycle theft':'sum',  'Crime type\_Burglary':'sum',  'Crime type\_Criminal damage and arson':'sum',  'Crime type\_Drugs':'sum',  'Crime type\_Other crime':'sum',  'Crime type\_Other theft':'sum',  'Crime type\_Possession of weapons':'sum',  'Crime type\_Public order':'sum',  'Crime type\_Robbery':'sum',  'Crime type\_Shoplifting':'sum',  'Crime type\_Theft from the person':'sum',  'Crime type\_Vehicle crime':'sum',  'Crime type\_Violence and sexual offences':'sum'  }  ).reset\_index()  clustering\_data[:5] |

|  |
| --- |
| clustering\_data\_original = clustering\_data.copy()  clustering\_data\_original.head() |

|  |
| --- |
| data\_scaled = normalize(clustering\_data)  data\_scaled = pd.DataFrame(data\_scaled, columns=clustering\_data.columns)  data\_scaled.head() |

|  |
| --- |
| plt.figure(figsize=(10, 7))  plt.title("Dendrograms")  dend = shc.dendrogram(shc.linkage(data\_scaled, method='ward')) |

|  |
| --- |
| plt.figure(figsize=(10, 7))  plt.title("Dendrograms")  dend = shc.dendrogram(shc.linkage(data\_scaled, method='ward'))  plt.axhline(y=1.25, color='r', linestyle='--') |

|  |
| --- |
| cluster = AgglomerativeClustering(n\_clusters=4, affinity='euclidean', linkage='ward')  cluster\_ids = cluster.fit\_predict(data\_scaled) |

|  |
| --- |
| clustering\_data['cluster'] = cluster\_ids  clustering\_data.head() |

|  |
| --- |
| hierarchical\_cluster = pd.DataFrame(round(clustering\_data.groupby('cluster').mean(),1))  hierarchical\_cluster |

|  |
| --- |
| clustering\_data\_original['cluster'] = cluster\_ids  clusters = clustering\_data\_original[['LSOA code', 'cluster']] |

|  |
| --- |
| clustered\_full = pd.merge(filtered\_data, clusters, on='LSOA code')  clustered\_full.head() |

|  |
| --- |
| def get\_color(cluster\_id):  if cluster\_id == 1:  return 'green'  if cluster\_id == 2:  return 'black'  if cluster\_id == 0:  return 'red'  if cluster\_id == 3:  return 'blue' |

|  |
| --- |
| #create a map  this\_map = folium.Map(location =[clustered\_full["Latitude"].mean(),  clustered\_full["Longitude"].mean()], zoom\_start=5)  def plot\_dot(point):  '''input: series that contains a numeric named latitude and a numeric named longitude  this function creates a CircleMarker and adds it to your this\_map'''  folium.CircleMarker(location=[point.Latitude, point.Longitude],  radius=2,  color=point.color,  weight=1).add\_to(this\_map)  clustered\_full["color"] = clustered\_full["cluster"].apply(lambda x: get\_color(x))  #use df.apply(,axis=1) to iterate through every row in your dataframe  clustered\_full.apply(plot\_dot, axis = 1)  #Set the zoom to the maximum possible  this\_map.fit\_bounds(this\_map.get\_bounds())  #Save the map to an HTML file  this\_map.save(os.path.join('the\_map.html'))  #IFrame(src='Crime\_map.html', width=1000, height=600) |

|  |
| --- |
| import folium  # group the clustered\_full dataframe by Crime type and count the number of crimes in each category  crime\_counts = clustered\_full.groupby('Crime type').size()  # create a map centered on the City of London  m = folium.Map(location=[51.5074, -0.1278], zoom\_start=13)  # add markers for each crime type, scaled by the number of crimes  for i, crime\_type in enumerate(crime\_counts.index):  # get the number of crimes for this crime type  count = crime\_counts[i]  # create a marker at a random location within the City of London  lat, lon = (51.5108, -0.1180) # replace with the coordinates of a suitable location  marker = folium.Marker(location=[lat, lon], popup=f'{crime\_type}: {count}', tooltip=crime\_type,  icon=folium.Icon(icon='info-sign', color='red'))  # scale the marker size based on the number of crimes  marker.radius = max(5, count // 20)  marker.add\_to(m) |

|  |
| --- |
| # display the map |

This code examines a dataset of criminal incidents in London. It begins by importing the necessary libraries before reading a CSV file containing the crime data. The code then pre-processes and filters the data to concentrate on the City of London. Following that, it organises the data by LSOA code and crime type, and then uses hierarchical clustering to divide the LSOAs into clusters based on the various types of crimes.

Following clustering, the code displays the findings on two separate maps. The first map depicts the clustered LSOAs, with various colours denoting distinct clusters. The second map shows markers for each crime type in the City of London, with the size of the markers scaled by the number of offences.

In brief, the code reads in a dataset of London crimes, pre-processes the data, clusters the LSOAs based on crime types, and then visualises the clustered LSOAs and crime counts on two distinct maps.