CASA0006_Final_Assessment

April 20, 2024

1 Traffic Collision Analysis of Accident 'Black Spots' in London

1.1 through Spatial Clustering & Supervised Learning

London Road Safety Dataset: London Collisions 2022

London Spatial Dataset: London Boundaries

1.2 1 Introduction

Black spots are areas on the public road network where the number of accidents is significantly higher than expected (Karamanlis, I. et al., 2023). Globally, road safety issues are becoming increasingly severe. According to the "Global Status Report on Road Safety 2018" released by the World Health Organization, approximately 1.35 million people die each year due to road traffic accidents, making it the leading cause of death for individuals aged 5 to 29. These figures not only highlight the severity of the issue but also remind us of the urgent need to take measures to improve road safety. Indeed, the United Nations Environment Programme (UNEP, 2003) also indicates that the socio-economic costs of road traffic accidents are among the most significant of all human activities. Therefore, identifying and improving black spots is an essential strategy for enhancing road safety and reducing accident occurrence.

1.3 2 Literature Review

Research on traffic accident black spots has been enriched by a range of influential studies, each employing distinct methodologies. Karamanlis et al. (2023) emphasize the critical role of black spot analysis in enhancing road safety. Additionally, Anderson (2005) explores the spatial variations in road collisions in London, attributing these to factors such as engineering flaws and personal mobility.

Furthermore, Szénási and Csiba (2014) investigate the application of DBSCAN clustering analysis for pinpointing black spots, showcasing how this method effectively aggregates locations prone to frequent accidents. Siamidoudaran and Iscioglu (2019) demonstrate the application of supervised learning in predicting the severity of injuries from accidents, thereby improving the accuracy of risk assessments in road safety.

Together, these studies highlight the importance of scientific approaches in the identification and management of traffic accident black spots, contributing significantly to the advancement of road safety.

1.4 3 Research Question

This report aims to address the following research questions: "Is it possible to categorize or describe the characteristics of severe traffic collision areas, known as 'Black Spots', in London for the year 2022?" and "Can lessons be learned from these clustering patterns to optimize urban traffic planning?"

[1]:
```
import warnings
warnings.filterwarnings("ignore")
```

[2]:
```
# First, load all the packages...
import pysal as ps
import numpy as np
import pandas as pd
import seaborn as sns
import geopandas as gpd
import matplotlib.pyplot as plt
import plotly.express as px

from math import ceil

from sklearn.neighbors import NearestNeighbors
from sklearn.model_selection import train_test_split
from sklearn.cluster import KMeans, DBSCAN, OPTICS, AgglomerativeClustering
from sklearn.preprocessing import RobustScaler, MinMaxScaler, StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score
from sklearn.multiclass import OneVsRestClassifier
from esda.adbscan import ADBSCAN

from scipy.cluster.hierarchy import dendrogram
from shapely.geometry import Point

import spopt
from spopt.region import MaxPHeuristic as MaxP
import matplotlib.pyplot as plt

import libpysal
```

1.5 4 Presentation of Data

London Road Safety Dataset (London Collisions 2022) encapsulates traffic collisions occurring throughout London in 2022, provided by official road safety data from the UK government. Additionally, it comprises detailed records of individual traffic incidents as reported to local authorities. Each entry includes specifics of where and when collisions occurred, the type of location, involved vehicles, severity of injuries, and more details.

1.5.1 4.1 Data Input

In this section, we load and process the 2022 London Road Safety data along with London's spatial boundary data, merging them to analyze accidents within specific geographic areas (MSOA) of the city.

[3]:
```
# read the road safety data
accidents = pd.read_csv(
    "https://data.dft.gov.uk/road-accidents-safety-data/
↪dft-road-casualty-statistics-collision-2022.csv",
    low_memory=False,
)
accidents.head()
```

[3]:
```
  accident_index  accident_year accident_reference  location_easting_osgr  \
0  2022010352073           2022          010352073               525199.0
1  2022010352573           2022          010352573               546214.0
2  2022010352575           2022          010352575               551119.0
3  2022010352578           2022          010352578               528889.0
4  2022010352580           2022          010352580               539773.0

   location_northing_osgr  longitude   latitude  police_force  \
0                177928.0  -0.198224  51.486454             1
1                179866.0   0.105042  51.498830             1
2                174789.0   0.173482  51.451924             1
3                192230.0  -0.139873  51.614153             1
4                190404.0   0.016495  51.595151             1

   accident_severity  number_of_vehicles  …  \
0                  3                   2  …
1                  3                   2  …
2                  3                   2  …
3                  3                   2  …
4                  3                   4  …

   pedestrian_crossing_physical_facilities light_conditions  \
0                                        0                4
1                                        4                4
2                                        0                4
3                                        0                4
4                                        0                4

   weather_conditions road_surface_conditions  special_conditions_at_site  \
0                   1                       1                           0
1                   1                       1                           0
2                   1                       1                           0
3                   1                       1                           0
4                   1                       1                           0
```

```
  carriageway_hazards urban_or_rural_area  \
0                   0                   1
1                   0                   2
2                   0                   1
3                   0                   1
4                   0                   1

   did_police_officer_attend_scene_of_accident  trunk_road_flag  \
0                                            1                2
1                                            1                2
2                                            1                2
3                                            1                2
4                                            1                2

   lsoa_of_accident_location
0                  E01001883
1                  E01033745
2                  E01000378
3                  E01001529
4                  E01003673

[5 rows x 36 columns]
```

[4]:
```
url = 'https://data.london.gov.uk/download/
↪statistical-gis-boundary-files-london/9ba8c833-6370-4b11-abdc-314aa020d5e0/
↪statistical-gis-boundaries-london.zip'
! wget $url
```

```
--2024-04-20 03:51:23--  https://data.london.gov.uk/download/statistical-gis-
boundary-files-london/9ba8c833-6370-4b11-abdc-314aa020d5e0/statistical-gis-
boundaries-london.zip
Resolving data.london.gov.uk (data.london.gov.uk)… 104.26.7.203,
172.67.72.228, 104.26.6.203, …
Connecting to data.london.gov.uk (data.london.gov.uk)|104.26.7.203|:443…
connected.
HTTP request sent, awaiting response… 302 Found
Location: https://airdrive-secure.s3-eu-
west-1.amazonaws.com/london/dataset/statistical-gis-boundary-files-
london/2016-10-03T13%3A52%3A28/statistical-gis-boundaries-london.zip?X-Amz-
Algorithm=AWS4-HMAC-SHA256&X-Amz-
Credential=AKIAJJDIMAIVZJDICKHA%2F20240420%2Feu-
west-1%2Fs3%2Faws4_request&X-Amz-Date=20240420T035128Z&X-Amz-Expires=300&X-Amz-S
ignature=9899c3b096c62637901b081f05953d658922856d8b4b2e3445a296409c77f34e&X-Amz-
SignedHeaders=host [following]
--2024-04-20 03:51:28--  https://airdrive-secure.s3-eu-
west-1.amazonaws.com/london/dataset/statistical-gis-boundary-files-
```

```
london/2016-10-03T13%3A52%3A28/statistical-gis-boundaries-london.zip?X-Amz-
Algorithm=AWS4-HMAC-SHA256&X-Amz-
Credential=AKIAJJDIMAIVZJDICKHA%2F20240420%2Feu-
west-1%2Fs3%2Faws4_request&X-Amz-Date=20240420T035128Z&X-Amz-Expires=300&X-Amz-S
ignature=9899c3b096c62637901b081f05953d658922856d8b4b2e3445a296409c77f34e&X-Amz-
SignedHeaders=host
Resolving airdrive-secure.s3-eu-west-1.amazonaws.com (airdrive-secure.s3-eu-
west-1.amazonaws.com)… 3.5.64.192, 3.5.72.138, 3.5.71.167, …
Connecting to airdrive-secure.s3-eu-west-1.amazonaws.com (airdrive-secure.s3-eu-
west-1.amazonaws.com)|3.5.64.192|:443… connected.
HTTP request sent, awaiting response… 200 OK
Length: 28666674 (27M) [application/zip]
Saving to: ‘statistical-gis-boundaries-london.zip’

statistical-gis-bou 100%[===================>]  27.34M  1.00MB/s    in 2m 0s

2024-04-20 03:53:31 (232 KB/s) - ‘statistical-gis-boundaries-london.zip’ saved
[28666674/28666674]
```

[5]:
```
# read the London's spatial boundary data (MSOA)
gdf_london_msoa = gpd.read_file(f"zip://statistical-gis-boundaries-london.zip!
 ↪statistical-gis-boundaries-london/ESRI/MSOA_2011_London_gen_MHW.shp")
gdf_london_msoa.head()
```

[5]:
```
     MSOA11CD                 MSOA11NM    LAD11CD              LAD11NM  \
0  E02000001       City of London 001  E09000001       City of London
1  E02000002  Barking and Dagenham 001  E09000002  Barking and Dagenham
2  E02000003  Barking and Dagenham 002  E09000002  Barking and Dagenham
3  E02000004  Barking and Dagenham 003  E09000002  Barking and Dagenham
4  E02000005  Barking and Dagenham 004  E09000002  Barking and Dagenham

     RGN11CD RGN11NM  USUALRES  HHOLDRES  COMESTRES  POPDEN  HHOLDS  \
0  E12000007  London      7375      7187        188    25.5    4385
1  E12000007  London      6775      6724         51    31.3    2713
2  E12000007  London     10045     10033         12    46.9    3834
3  E12000007  London      6182      5937        245    24.8    2318
4  E12000007  London      8562      8562          0    72.1    3183

   AVHHOLDSZ                                           geometry
0        1.6  MULTIPOLYGON (((531667.624 180534.992, 531647…
1        2.5  POLYGON ((548881.563 190845.265, 548881.125 19…
2        2.6  POLYGON ((549102.438 189324.625, 548954.500 18…
3        2.6  POLYGON ((551549.998 187364.637, 551478.000 18…
4        2.7  POLYGON ((549099.634 187656.076, 549161.375 18…
```

[6]:
```python
accidents_gdf = gpd.GeoDataFrame(
    accidents,
    geometry=gpd.points_from_xy(accidents.longitude, accidents.latitude)
)

# Set the coordinate reference system (CRS) to WGS84.
accidents_gdf.crs = {"init": "epsg:4326"}
gdf_london_msoa = gdf_london_msoa.to_crs("EPSG:4326")

# Spatially connect 'accidents_gdf' to 'gdf_london_msoa'
joined = gpd.sjoin(accidents_gdf, gdf_london_msoa, how="left", op='within')

# Copy MSOA code to new column 'msoa_of_accident_location'
accidents['msoa_of_accident_location'] = joined['MSOA11CD']

accidents.head()
```

[6]:
```
   accident_index  accident_year accident_reference  location_easting_osgr  \
0  2022010352073           2022          010352073               525199.0
1  2022010352573           2022          010352573               546214.0
2  2022010352575           2022          010352575               551119.0
3  2022010352578           2022          010352578               528889.0
4  2022010352580           2022          010352580               539773.0

   location_northing_osgr  longitude   latitude  police_force  \
0                177928.0  -0.198224  51.486454             1
1                179866.0   0.105042  51.498830             1
2                174789.0   0.173482  51.451924             1
3                192230.0  -0.139873  51.614153             1
4                190404.0   0.016495  51.595151             1

   accident_severity  number_of_vehicles  …  light_conditions  \
0                  3                   2  …                 4
1                  3                   2  …                 4
2                  3                   2  …                 4
3                  3                   2  …                 4
4                  3                   4  …                 4

  weather_conditions  road_surface_conditions special_conditions_at_site  \
0                  1                        1                          0
1                  1                        1                          0
2                  1                        1                          0
3                  1                        1                          0
4                  1                        1                          0

   carriageway_hazards urban_or_rural_area  \
0                    0                   1
```

```
1                    0                   2
2                    0                   1
3                    0                   1
4                    0                   1

  did_police_officer_attend_scene_of_accident  trunk_road_flag  \
0                                            1                2
1                                            1                2
2                                            1                2
3                                            1                2
4                                            1                2

   lsoa_of_accident_location  msoa_of_accident_location
0                  E01001883                  E02000388
1                  E01033745                  E02000314
2                  E01000378                  E02000083
3                  E01001529                  E02000308
4                  E01003673                  E02000757

[5 rows x 37 columns]
```

1.5.2 4.2 Data Pre-processing

[7]:
```python
# Pick the orriented varribles before exploration data analysis
accidents = accidents[['accident_index', 'accident_severity',␣
 ↪'number_of_vehicles', 'number_of_casualties', 'day_of_week',␣
 ↪'msoa_of_accident_location', 'first_road_class',
         'road_type', 'speed_limit', 'second_road_class',␣
 ↪'pedestrian_crossing_physical_facilities', 'light_conditions',␣
 ↪'weather_conditions', 'road_surface_conditions',
         'urban_or_rural_area', 'longitude', 'latitude']]

# Pick out the 'London' rows
gss_codes = gdf_london_msoa['MSOA11CD'].unique()
accidents = accidents[accidents['msoa_of_accident_location'].isin(gss_codes)]

# Drop the rows which have null value
accidents = accidents.dropna()
```

To analyze the numeric attributes of the road safety dataset, descriptive statistics have been computed as detailed in the provided code above. Key observations from this analysis include:

- The cleaned dataset comprises 23,443 records.
- The average number of vehicles involved in collisions (1.82) suggests that most accidents involve more than one vehicle.
- The mean accident severity rating is 2.83, indicating that most reported accidents are of moderate severity.

- The average number of casualties per accident is relatively low at 1.16, reflecting that most accidents result in fewer injuries.

[8]:
```python
# Check the numeric columns and descriptive statistics
print(accidents.describe(include = [np.number]))
```

```
       accident_severity  number_of_vehicles  number_of_casualties  \
count       23469.000000        23469.000000          23469.000000
mean            2.833056            1.820188              1.159956
std             0.384300            0.623840              0.543314
min             1.000000            1.000000              1.000000
25%             3.000000            1.000000              1.000000
50%             3.000000            2.000000              1.000000
75%             3.000000            2.000000              1.000000
max             3.000000           13.000000             16.000000

        day_of_week  first_road_class     road_type   speed_limit  \
count  23469.000000      23469.000000  23469.000000  23469.000000
mean       4.165154          3.911372      5.241084     26.342409
std        1.939557          1.248446      2.041853      7.603702
min        1.000000          1.000000      1.000000     20.000000
25%        3.000000          3.000000      3.000000     20.000000
50%        4.000000          3.000000      6.000000     30.000000
75%        6.000000          5.000000      6.000000     30.000000
max        7.000000          6.000000      9.000000     70.000000

       second_road_class  pedestrian_crossing_physical_facilities  \
count       23469.000000                             23469.000000
mean            4.043632                                 2.732669
std             2.377180                                 3.285093
min             0.000000                                 0.000000
25%             3.000000                                 0.000000
50%             5.000000                                 1.000000
75%             6.000000                                 5.000000
max             6.000000                                 9.000000

       light_conditions  weather_conditions  road_surface_conditions  \
count      23469.000000        23469.000000              23469.00000
mean           2.022626            1.986450                  1.55013
std            1.546107            2.379119                  1.66809
min            1.000000            1.000000                  1.00000
25%            1.000000            1.000000                  1.00000
50%            1.000000            1.000000                  1.00000
75%            4.000000            1.000000                  1.00000
max            7.000000            9.000000                  9.00000

       urban_or_rural_area     longitude      latitude
count         23469.000000  23469.000000  23469.000000
```

```
mean             1.028889    -0.121903    51.509996
std              0.167499     0.142148     0.068528
min              1.000000    -0.508592    51.294892
25%              1.000000    -0.204350    51.465912
50%              1.000000    -0.115516    51.514507
75%              1.000000    -0.037785    51.556095
max              2.000000     0.301856    51.691585
```

1.5.3 4.3 Data Preparation

For clustering and spatial analysis, it is essential to select the necessary data for merging by neighborhoods. The spatial clustering analysis will concentrate on the four numerical variables detailed in the table below.

Variables	Definition
`Severity of Accidents`	The average severity of accidents within a specific neighborhood.
`Frequency of Collision Accidents`	Frequency of Collision Accidents happening in a specific neighbourbood.
`Average Number of Vehicles`	The average number of vehicles involved per traffic accident in a specific neighborhood.
`Average Number of Casualties`	The average number of casualties per traffic accident in a specific neighborhood.

[9]:
```python
# Group by MSOA_ID and return the mean/count in the group.
a = pd.DataFrame(accidents.
 ↪groupby('msoa_of_accident_location')['accident_severity'].mean())
b = pd.DataFrame(accidents.
 ↪groupby('msoa_of_accident_location')['accident_index'].count())
c = pd.DataFrame(accidents.
 ↪groupby('msoa_of_accident_location')['number_of_vehicles'].mean())
d = pd.DataFrame(accidents.
 ↪groupby('msoa_of_accident_location')['number_of_casualties'].mean())

# change the columns' name
b.rename(columns={'accident_index':'Freq'},inplace=True)
c.rename(columns={'number_of_vehicles':
 ↪'average_number_of_vehicles'},inplace=True)
d.rename(columns={'number_of_casualties':
 ↪'average_number_of_casualties'},inplace=True)

# Then we should merge these four attributes' values
temp_1 = a.join(b)
temp_2 = temp_1.join(c)
temp_3 = temp_2.join(d)
```

```
dfm = temp_3.copy()
dfm.head()
```

[9]:
```
                             accident_severity  Freq  \
msoa_of_accident_location
E02000001                             2.659091   176
E02000002                             2.888889    18
E02000003                             2.884615    52
E02000004                             2.750000     4
E02000005                             2.700000    10

                             average_number_of_vehicles  \
msoa_of_accident_location
E02000001                                      1.693182
E02000002                                      2.277778
E02000003                                      1.673077
E02000004                                      2.000000
E02000005                                      1.900000

                             average_number_of_casualties
msoa_of_accident_location
E02000001                                        1.068182
E02000002                                        1.388889
E02000003                                        1.307692
E02000004                                        1.500000
E02000005                                        1.400000
```

The following descriptive statistics will help us understand how to scale these four values. For instance, the average number of accidents per neighborhood is approximately 23.85, with a substantial variability indicated by a standard deviation of 17.59. This suggests that while some neighborhoods might experience few accidents, others could have significantly more, with the maximum reaching 224 accidents.

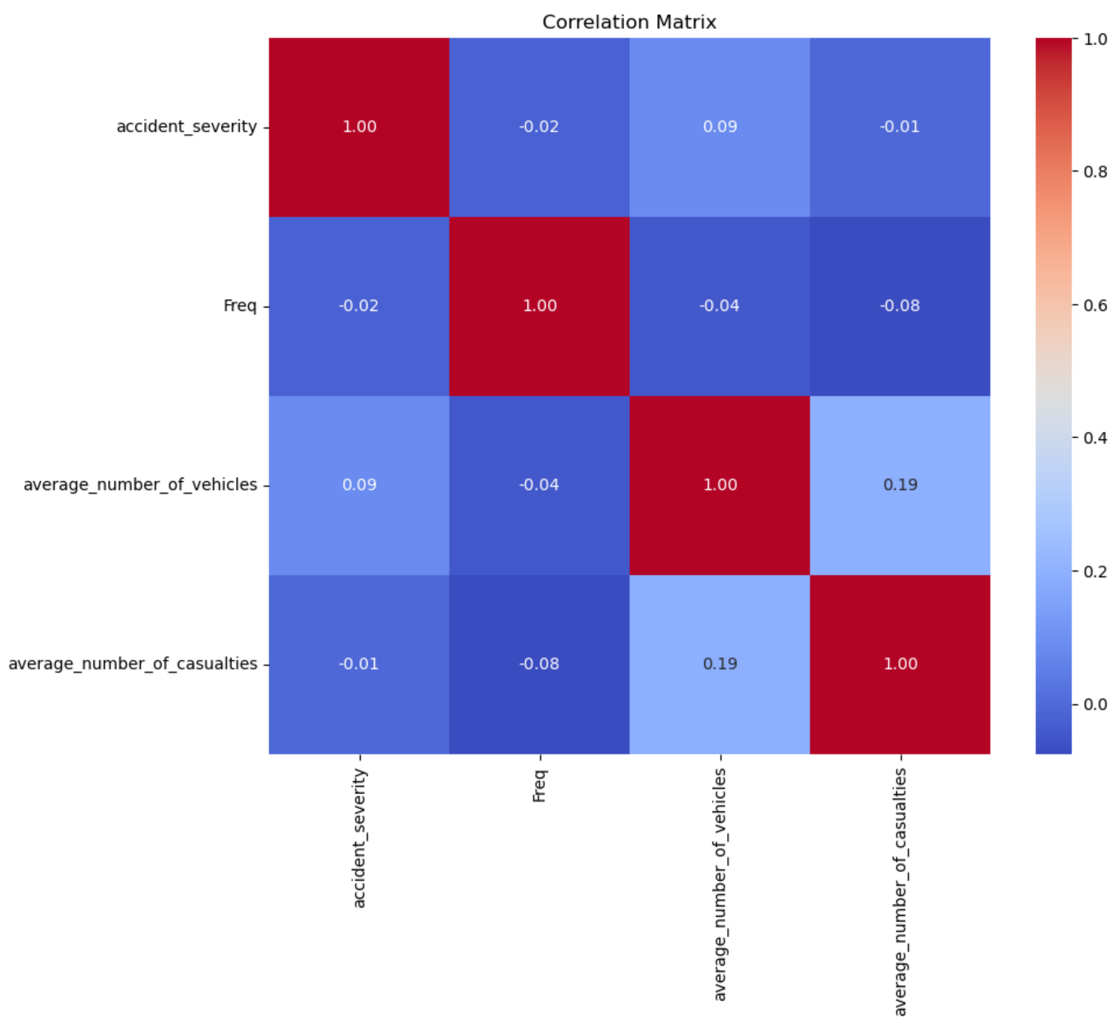
[10]:
```
dfm.describe()
```

[10]:
```
       accident_severity        Freq  average_number_of_vehicles  \
count         983.000000  983.000000                  983.000000
mean            2.834708   23.874873                    1.826043
std             0.102265   17.663799                    0.182672
min             2.250000    1.000000                    1.000000
25%             2.777778   13.000000                    1.724747
50%             2.842105   20.000000                    1.814815
75%             2.904762   30.000000                    1.923077
max             3.000000  224.000000                    3.000000

       average_number_of_casualties
count                    983.000000
```

```
mean                           1.168800
std                            0.158103
min                            1.000000
25%                            1.060606
50%                            1.142857
75%                            1.235294
max                            2.461538
```

[11]:

```python
# Calculate the correlation matrix
corr = dfm.corr()

# Heat mapping with Seaborn
plt.figure(figsize=(10,8))
sns.heatmap(corr, annot=True, fmt=".2f", cmap="coolwarm", cbar=True)
plt.title("Correlation Matrix")
plt.show()
```

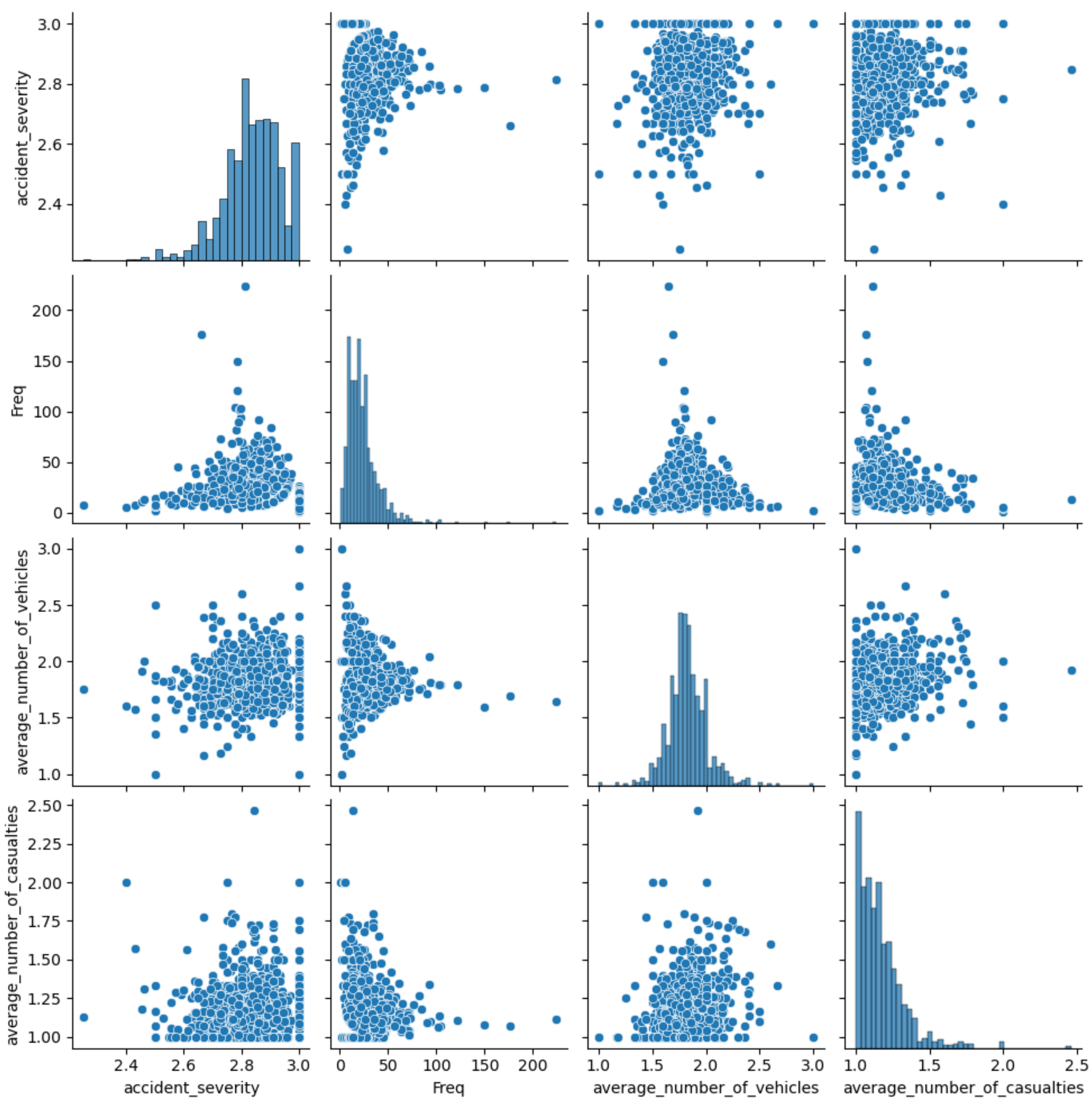


After examining the merged DataFrame containing the four variables, we find that the likelihood of significant relationships between these attributes is relatively low based on the correlation matrix. Additionally, by employing exploratory data analysis methods, we can clearly observe the distribution patterns within the merged DataFrame.

1.5.4 4.4 Descriptive Statistics

[12]:
```
sns.pairplot(dfm)
```

[12]: <seaborn.axisgrid.PairGrid at 0x7efc2ecf4c50>

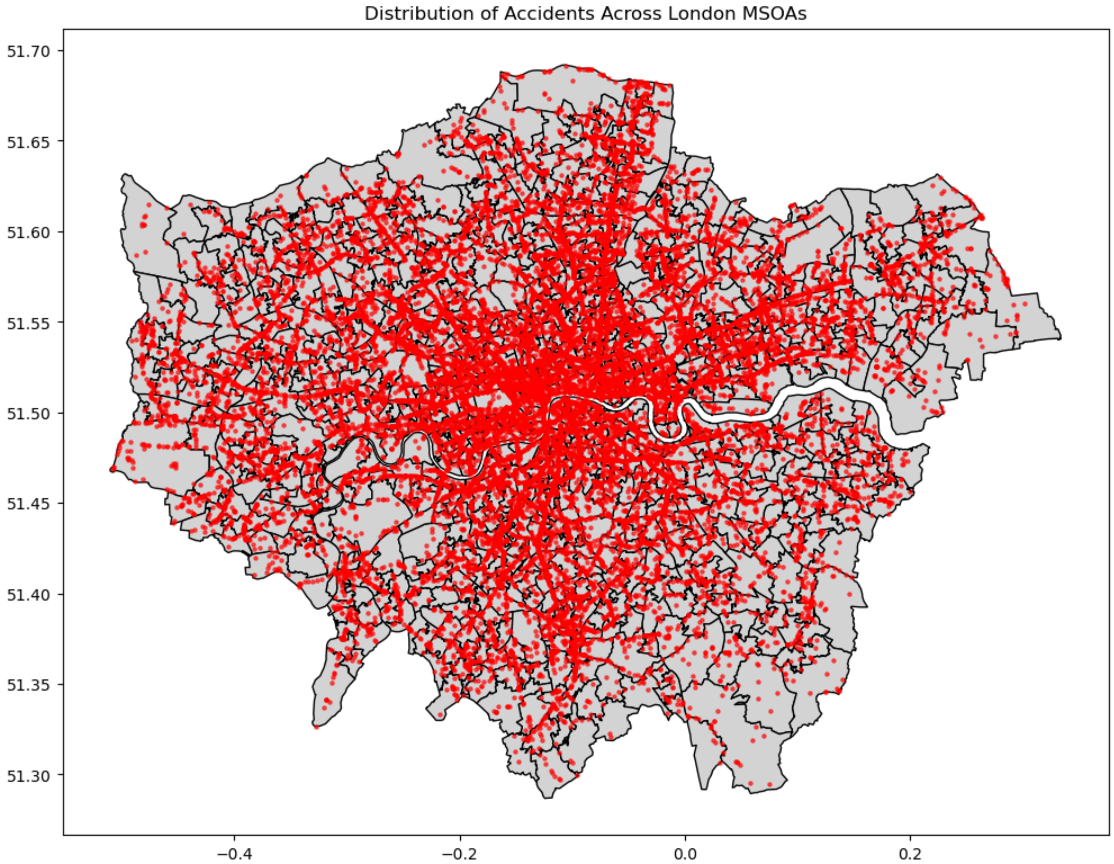


The map below illustrates the distribution of traffic accidents across London's Middle Super Output

Areas (MSOAs), with accidents highlighted in red. Key observations from this map include:

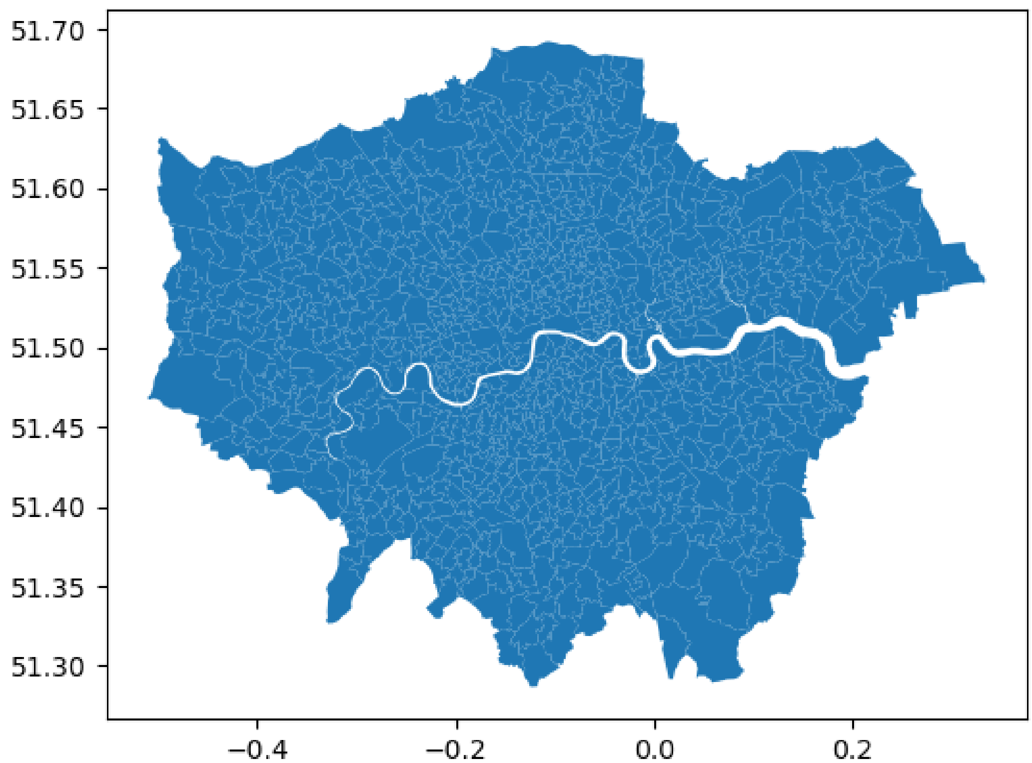
- Concentration of Accidents: A significant cluster of accidents is evident in central London, correlating with the region's high traffic density and increased accident rates.
- Accidents on Major Roads: Accidents are particularly concentrated on A-roads, which are known to handle heavier traffic and, consequently, experience higher accident rates.
- Consistency with Literature: Supporting findings by Balawi & Goktug Tenekeci (2024), about 60% of traffic accidents in London happen on A-roads, consistent with the patterns shown on the map.

[13]:
```python
# Creating a GeoDataFrame of the accidents location
accidents_gdf = gpd.GeoDataFrame(
    accidents,
    geometry=gpd.points_from_xy(accidents.longitude, accidents.latitude),
    crs="EPSG:4326"
)

# Ensure that the CRS of the GeoDataFrame for the London region is consistent␣
 ↪with the accidents data
gdf_london_msoa = gdf_london_msoa.to_crs("EPSG:4326")  # CRS WGS 84

fig, ax = plt.subplots(1, 1, figsize=(12, 12))

# Mapping the London boroughs
gdf_london_msoa.plot(ax=ax, color='lightgrey', edgecolor='black')

# Plotting accident points on the base map
accidents_gdf.plot(ax=ax, markersize=5, color='red', alpha=0.6)

# Add legend and title
ax.set_title('Distribution of Accidents Across London MSOAs')
plt.show()
```



[14]:
```
# merge with original case
# dfmg = dfm with geo info
dfmg = pd.merge(gdf_london_msoa, dfm, left_on='MSOA11CD',␣
↪right_on='msoa_of_accident_location', how='inner').reset_index()
dfmg.plot()
```

[14]: <Axes: >



The map presented above is the result of successfully merging two DataFrames, with the completeness of the map suggesting that the merge was executed without any data loss.

[15]:
```
# We can inspect the min and max in these three value in London
print('Frequency of Colision Accidents')
print(dfmg[dfmg.Freq.isin([dfmg.Freq.max(), dfmg.Freq.min()])][['Freq']])
```

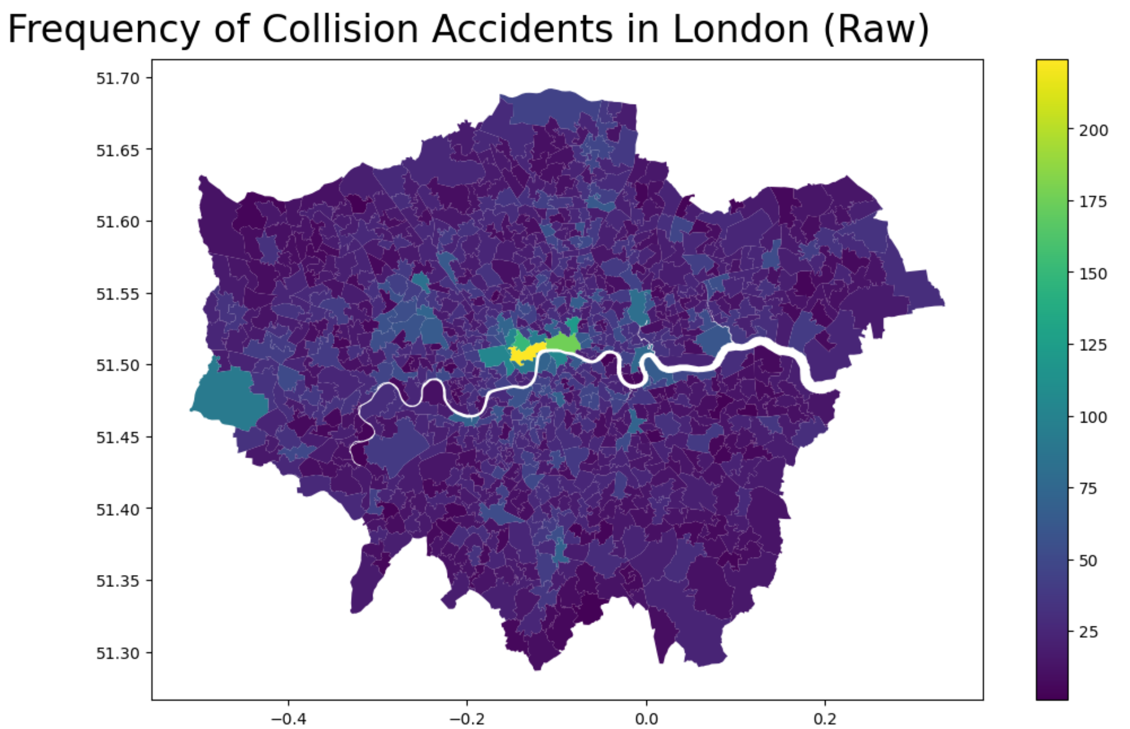
```
Frequency of Colision Accidents
     Freq
419     1
944   224
```

The key findings from this merged spatial and numeric dataset:

- The highest frequency of collision accidents is found in Westminster 018, with 224 incidents, located in the city center of London.
- The lowest frequency of collision accidents occurs in Harrow 003, with only one recorded incident, situated on the northwestern edge of London.

[16]:
```
tfont = {'fontname':'DejaVu Sans', 'horizontalalignment':'left'}
f = plt.figure()
f.set_size_inches(12,7)

ax1 = f.add_subplot()
```

```
dfmg.plot(column='Freq', legend=True, cmap='viridis', figsize=(12,6), ax=ax1)

f.subplots_adjust(top=0.92)
f.suptitle(f"Frequency of Collision Accidents in London (Raw)", x=0.025, size=24, **tfont);
plt.show()
```



The above frequency distribution shows that the closer you are to the city centre, the more frequent the accident "black spots" are, but there are some exceptions.

1.6 5 Methodology

This study focuses on identifying 'black spots' through spatial clustering and supervised learning techniques. It involves data normalization, the use of clustering algorithms such as DBSCAN and K-means, and the application of RF for advanced data classification.

1.6.1 5.1 Standardisation

This study concentrates on four variables detailed in the table below. The range of `accident_severity` is from 2.25 to 3.00; `average_number_of_vehicles` varies from 1 to 3; `average_number_of_casualties` spans from 1.00 to 2.46; and `Freq` extends from 1 to 224. All numerical data will undergo rescaling to fit within a (-1,1) range, a process known as standardization.

Variables	Definition	Value Scale
`Severity of Accident`	The average severity of accidents within a specific neighborhood.	(2.25, 3.00)
`Average Number of Vehicles`	The average number of vehicles involved per traffic accident in a specific neighborhood.	(1, 3)
`Average Number of Casualties`	The average number of casualties per traffic accident in a specific neighborhood.	(1.00, 2.46)
`Frequency of Collision Accidents`	Frequency of Collision Accidents happening in a specific neighbourbood.	(1, 224)

[17]:
```
rs = RobustScaler(quantile_range=(10.0, 90.0))

to_cluster = dfmg[['MSOA11CD','accident_severity','Freq',␣
 ↪'average_number_of_vehicles', 'average_number_of_casualties']].
 ↪set_index('MSOA11CD').copy()
to_cluster.head()

# normnise the value
normed = to_cluster.copy()
for c in to_cluster.columns.values:
    normed[c] = rs.fit_transform(to_cluster[c].values.reshape(-1,1))
normed.head()
```

[17]:
```
           accident_severity      Freq  average_number_of_vehicles  \
MSOA11CD
E02000001          -0.772428  4.457143                   -0.324355
E02000002           0.197454 -0.057143                    1.234568
E02000003           0.179418  0.914286                   -0.377968
E02000004          -0.388738 -0.457143                    0.493827
E02000005          -0.599768 -0.285714                    0.227160

           average_number_of_casualties
MSOA11CD
E02000001                     -0.211933
E02000002                      0.698254
E02000003                      0.467813
E02000004                      1.013594
E02000005                      0.729788
```

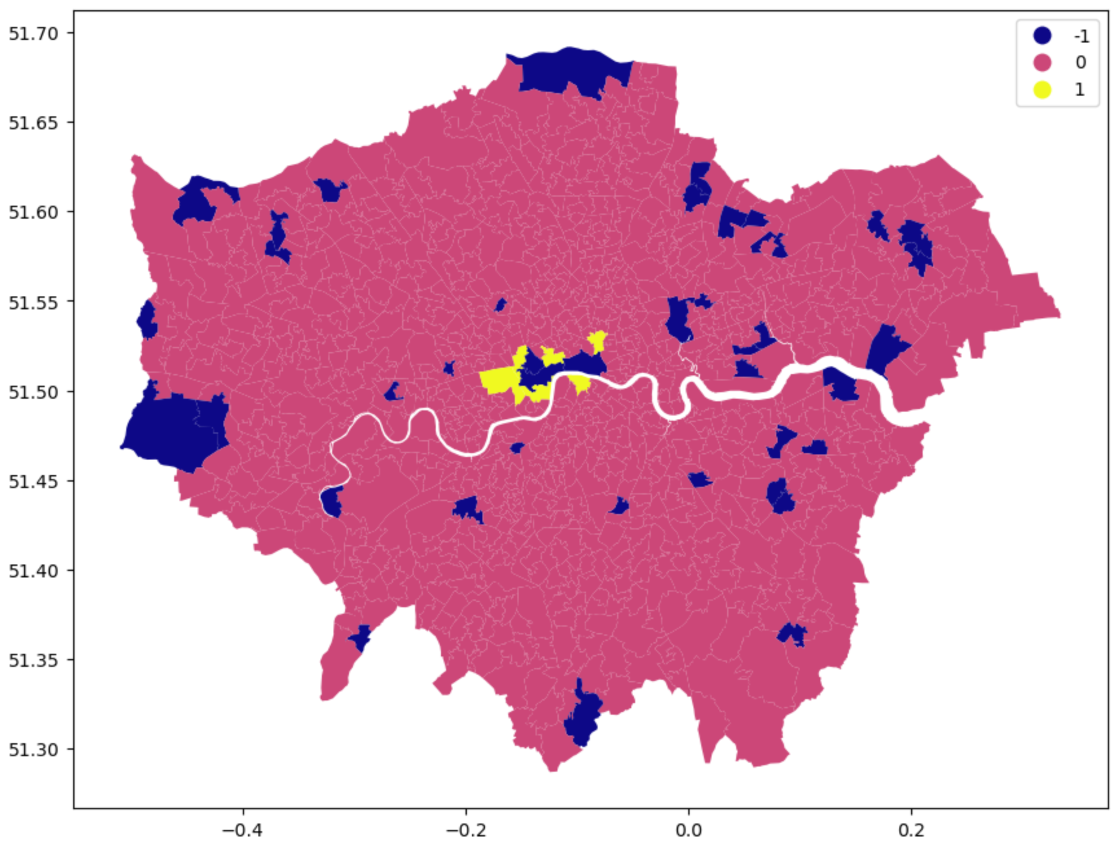
1.6.2 5.2 DBSCAN Clustering Analysis

After normalizing the four variables, DBSCAN clustering analysis can be performed to fit them. The parameters set for this DBSCAN cluster are as follows:

Epsilon (eps) is the maximum distance between two samples for one to be considered within the neighborhood of the other. The correct selection of DBSCAN parameters is crucial and depends on the dataset and the distance function used. Epsilon is set at 0.5, and the minimum number of points required in a neighborhood for a point to be considered a core point (minPts) is set at 3. Finally, the silhouette score method will be applied to evaluate the quality of the clustering results.

[18]:
```
minPts = 3
epsilon = 0.5
dbsc = DBSCAN(eps=epsilon, min_samples=minPts)
dbsc.fit(normed)

clusters = dbsc.labels_

dfmg['DBSCAN_Cluster'] = clusters
dfmg.plot(column='DBSCAN_Cluster', categorical=True, legend=True,␣
 ↪figsize=(12,8), cmap = 'plasma');

from sklearn import metrics
print("Silhouette Score is :")
print(metrics.silhouette_score(normed, clusters))
```

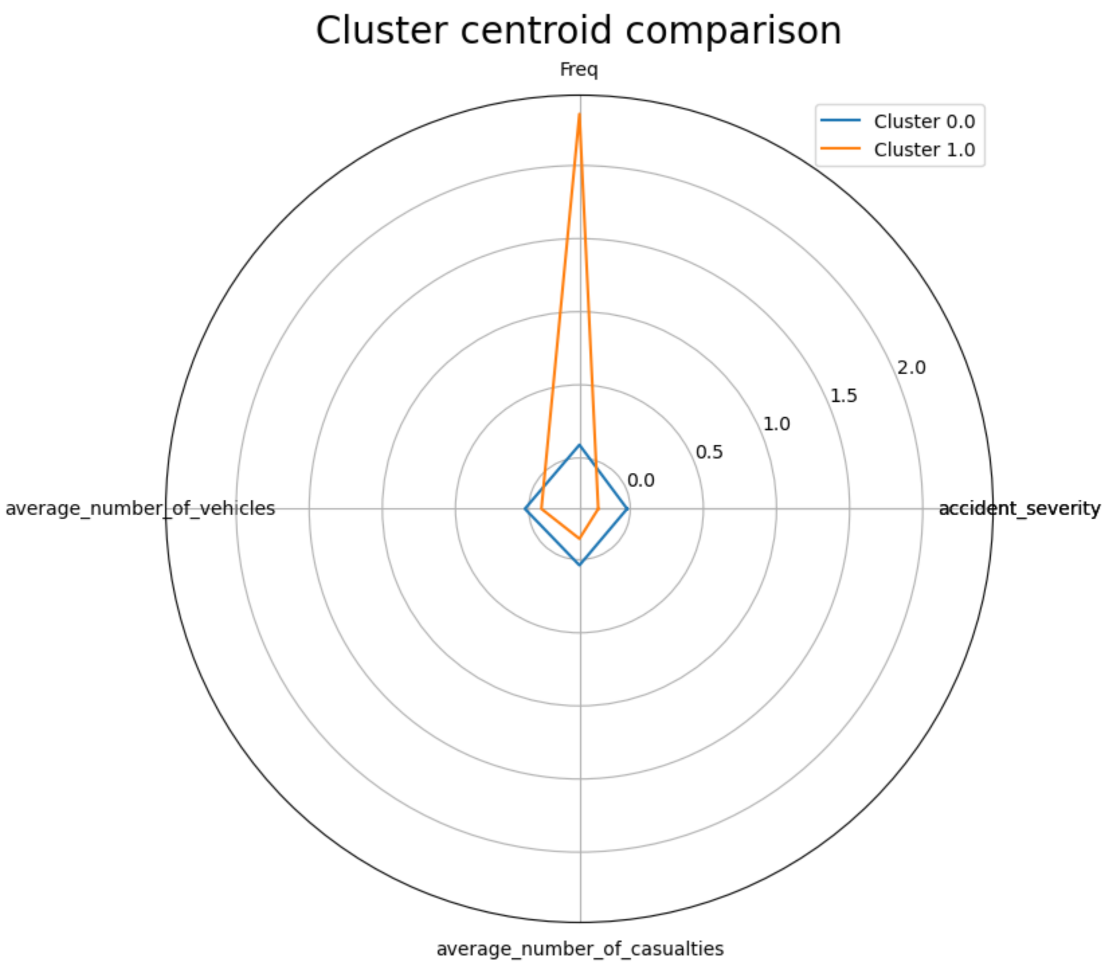
```
Silhouette Score is :
0.481413555180166
```



The silhouette score indicates that the DBSCAN clustering provides a reasonable structure. According to the tutorial, it is important to note that the areas marked in blue on the map represent noise samples for the DBSCAN method. Next, we will visualize the cluster centroids using a radar plot.

[19]:
```
def radar_plot_cluster_centroids1(df_cluster_centroid):
    # parameters
    # df_cluster_centroid: a dataframe with rows representing a cluster␣
 ↪centroid and columns representing variables

    # add an additional element to both categories and restaurants that's␣
 ↪identical to the first item
    # manually 'close' the line
    categories = df_cluster_centroid.columns.values.tolist()[1:]
    categories += [categories[0]]

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

    plt.figure(figsize=(12, 8))
    ax = plt.subplot(polar=True)
    for index, row in df_cluster_centroid.iterrows():
        centroid = row.tolist()[1:]
        centroid += [centroid[0]]
        label = "Cluster {}".format(row['cluster'])
        ax.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()
```

[20]:
```
df_dbscan = normed.copy()
df_dbscan['cluster'] = dbsc.labels_

# Calculate cluster centres and remove noisy points
df_dbscan_centroid1 = df_dbscan.groupby('cluster').mean()
df_dbscan_centroid1.drop(-1, inplace=True)
df_dbscan_centroid1.reset_index(inplace=True)

radar_plot_cluster_centroids1(df_dbscan_centroid1)
```



However, it is essential to append the new labels from the DBSCAN analysis to the original dataset, as all real data has been rescaled in the normalization process.

[21]:
```
summary_df_dbscan = \
    dfmg[['accident_severity', 'Freq', 'average_number_of_vehicles',␣
 ↪'average_number_of_casualties', 'DBSCAN_Cluster']].groupby('DBSCAN_Cluster')\
    .mean() \
    .round(2)

summary_df_dbscan
```

[21]:
```
                accident_severity    Freq  average_number_of_vehicles  \
DBSCAN_Cluster
-1                           2.79   28.86                        1.87
 0                           2.84   23.15                        1.82
 1                           2.79  102.33                        1.78
```

```
                     average_number_of_casualties
DBSCAN_Cluster
-1                                           1.46
 0                                           1.16
 1                                           1.09
```

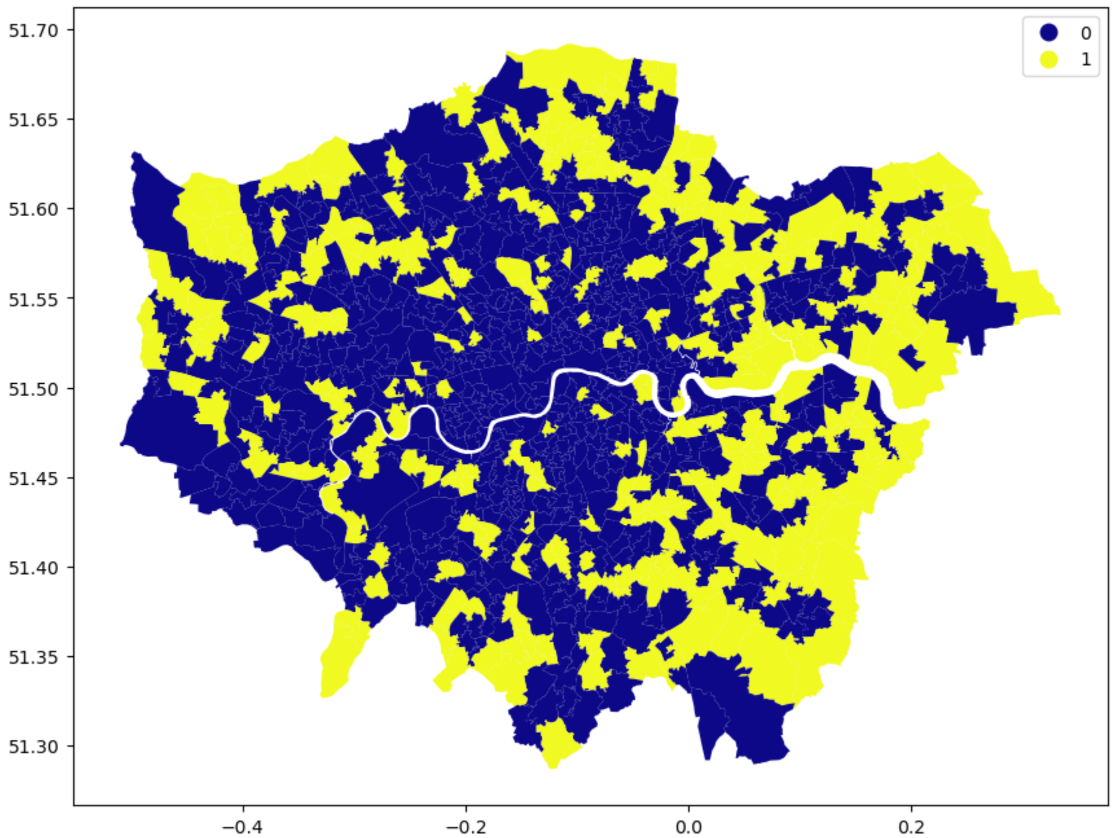
1.6.3 5.3 K-means Clustering Analysis

Simultaneously, the K-means clustering analysis is applied to the London Road Safety Dataset to explore its latent features. K-means clustering involves forming clusters and determining the number of centroids, as illustrated below. Similar to the aforementioned DBSCAN analysis, it is advisable to append the K-means clustering labels to the London Road Safety Dataset.

[22]:

```python
from sklearn.cluster import KMeans
k_cluster = 2
random_seed = 1
kmeans_method = KMeans(n_clusters=k_cluster,random_state=random_seed)
kmeans_method.fit(normed)

dfmg['Kmeans_Cluster'] = kmeans_method.labels_
dfmg.plot(column='Kmeans_Cluster', categorical=True, legend=True,␣
 ↪figsize=(12,8),cmap = 'plasma');
from sklearn import metrics
print("Silhouette Score is :")
print(metrics.silhouette_score(normed, dfmg['Kmeans_Cluster']))
```

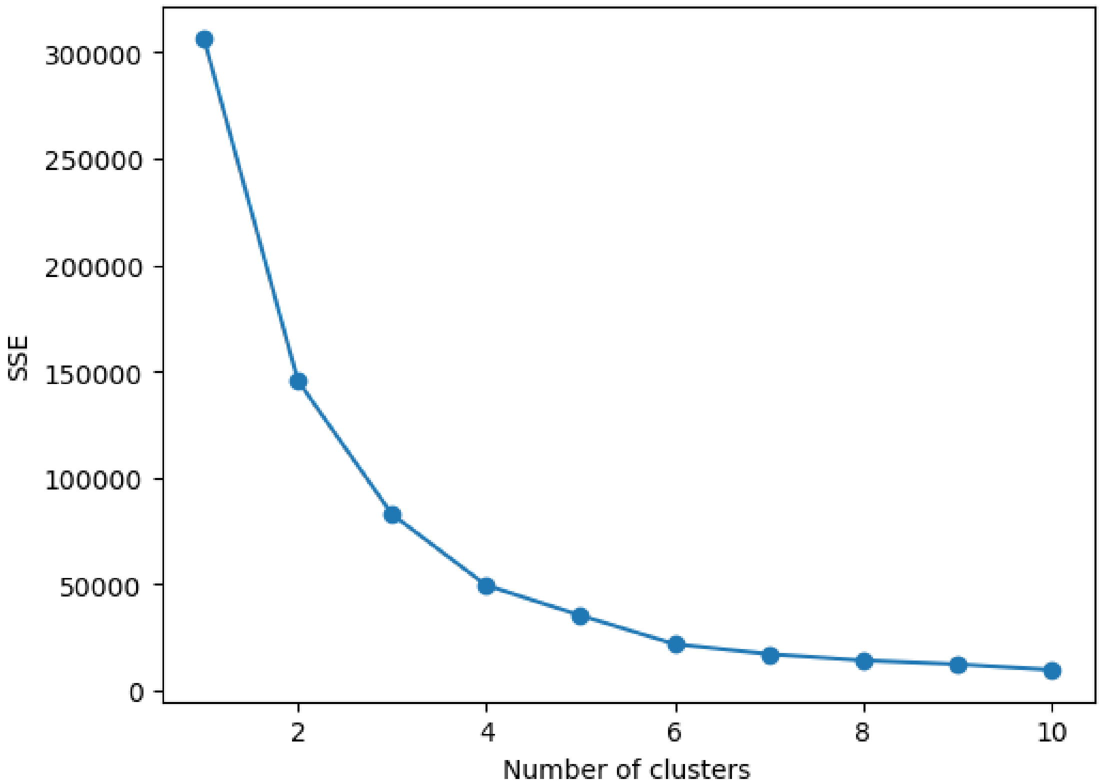
```
Silhouette Score is :
0.21749564861115095
```



The optimization solution involves using the SSE (Sum of Squared Errors) plot to determine the optimal number of clusters. The SSE plot below identifies the best k value for clustering. Subsequently, this study will re-run the K-Means function with k set to 3.

[23]:
```
# 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(to_cluster)
    # inertia is a concept in physics. Roughly it means SSE of clustering.
    list_SSE.append(km.inertia_)

# plot
plt.plot(range_k, list_SSE, marker='o')
```

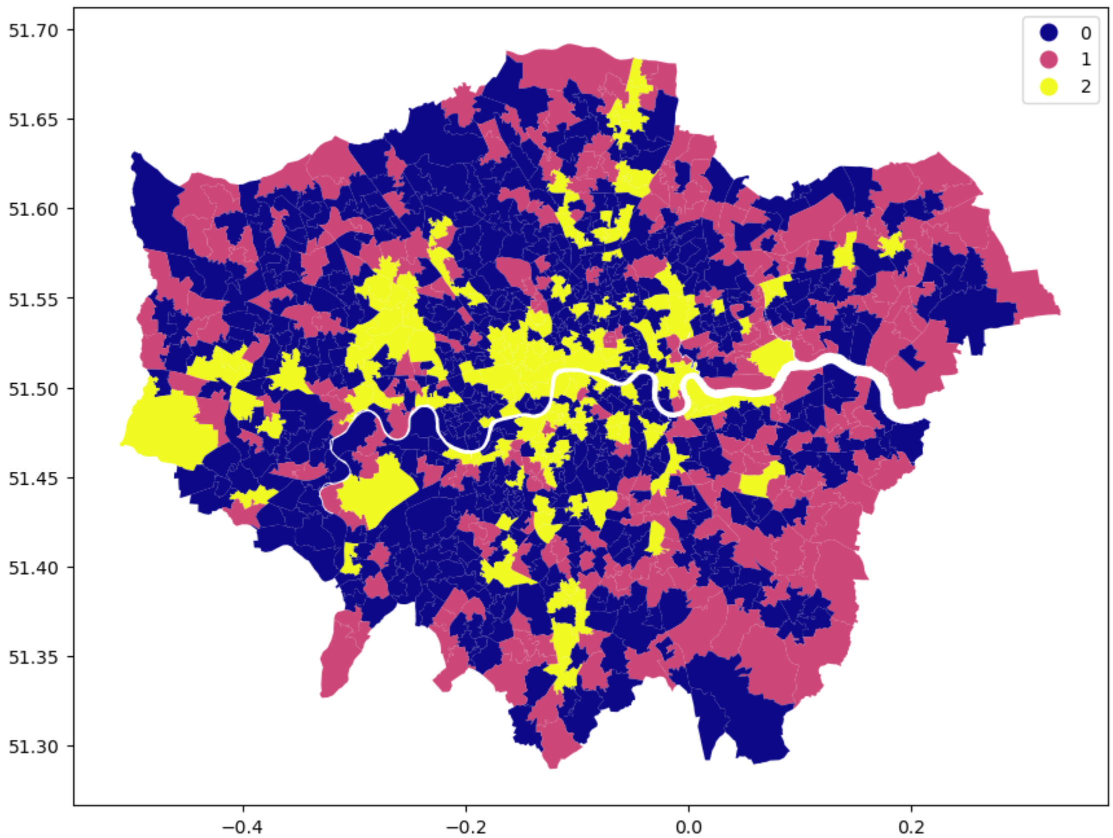
```
plt.xlabel('Number of clusters')
plt.ylabel('SSE')
plt.show()
```



[24]:

```
from sklearn.cluster import KMeans
k_cluster = 3
random_seed = 1
kmeans_method = KMeans(n_clusters=k_cluster,random_state=random_seed)
kmeans_method.fit(normed)

dfmg['Kmeans_Cluster'] = kmeans_method.labels_
dfmg.plot(column='Kmeans_Cluster', categorical=True, legend=True,␣
 ↪figsize=(12,8),cmap = 'plasma');
from sklearn import metrics
print("Silhouette Score is :")
print(metrics.silhouette_score(normed, dfmg['Kmeans_Cluster']))
```

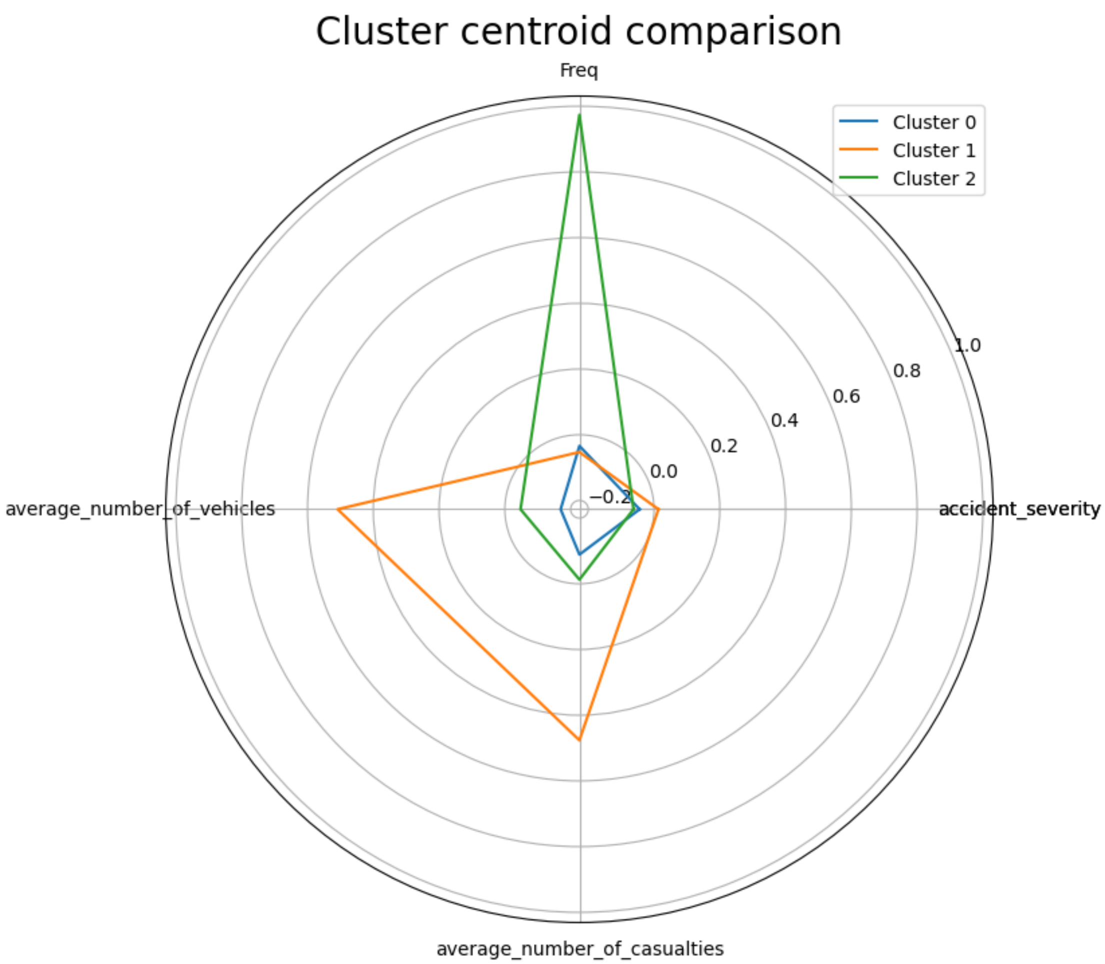
```
Silhouette Score is :
0.22306314322316226
```



[25]:
```
# adapted from this tutorial: https://towardsdatascience.com/
↪how-to-make-stunning-radar-charts-with-python-implemented-in-matplotlib-and-plotly-91e21801
def radar_plot_cluster_centroids2(df_cluster_centroid):
    # parameters
    # df_cluster_centroid: a dataframe with rows representing a cluster␣
↪centroid and columns representing variables

    # add an additional element to both categories and restaurants that's␣
↪identical to the first item
    # 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()
```

[26]:
```
df_cluster_centroid2 = pd.DataFrame(kmeans_method.cluster_centers_,␣
 ↪columns=normed.columns)
radar_plot_cluster_centroids2(df_cluster_centroid2)
```



[27]:
```
summary_df_Kmeans = \
    dfmg[['accident_severity', 'Freq', 'average_number_of_vehicles',␣
 ↪'average_number_of_casualties', 'Kmeans_Cluster']].groupby('Kmeans_Cluster')\
    .mean() \
    .round(2)
```

```
summary_df_Kmeans
```

[27]:

```
                accident_severity   Freq  average_number_of_vehicles  \
Kmeans_Cluster
0                            2.83  18.81                        1.75
1                            2.85  18.16                        2.01
2                            2.83  54.08                        1.80

                average_number_of_casualties
Kmeans_Cluster
0                                       1.11
1                                       1.31
2                                       1.14
```

1.6.4 5.4 RF Classification Analysis

The Random Forest (RF) model is crucial for classifying traffic accidents using characteristics identified during the initial clustering phase. It employs labels derived from K-means clustering as target categories, effectively categorizing accidents within predetermined clusters. Key attributes such as day of the week, road type, and speed limit are standardized and leveraged to train the RF model, which then predicts the severity or type of accidents. The model's accuracy is evaluated using performance metrics to ensure precise and reliable classification within the established clusters.

[28]:

```
accidents= pd.merge(accidents, dfmg[['MSOA11CD', 'Kmeans_Cluster']],
                      left_on='msoa_of_accident_location',
                      right_on='MSOA11CD', how='left')

accidents.rename(columns={'Kmeans_Cluster': 'cluster_label'}, inplace=True)

accidents.drop(columns=['MSOA11CD'], inplace=True)

accidents.head()
```

[28]:

```
  accident_index  accident_severity  number_of_vehicles  number_of_casualties  \
0   2022010352073                  3                   2                     1
1   2022010352573                  3                   2                     1
2   2022010352575                  3                   2                     1
3   2022010352578                  3                   2                     2
4   2022010352580                  3                   4                     3

   day_of_week msoa_of_accident_location  first_road_class  road_type  \
0            4                 E02000388                 3          6
1            7                 E02000314                 3          3
2            7                 E02000083                 3          6
3            7                 E02000308                 6          6
4            7                 E02000757                 3          3
```

```
   speed_limit  second_road_class  pedestrian_crossing_physical_facilities  \
0           30                  6                                        0
1           50                  6                                        4
2           30                  6                                        0
3           30                  0                                        0
4           50                  6                                        0

   light_conditions  weather_conditions  road_surface_conditions  \
0                 4                   1                        1
1                 4                   1                        1
2                 4                   1                        1
3                 4                   1                        1
4                 4                   1                        1

   urban_or_rural_area  longitude   latitude  cluster_label
0                    1  -0.198224  51.486454              0
1                    2   0.105042  51.498830              1
2                    1   0.173482  51.451924              1
3                    1  -0.139873  51.614153              0
4                    1   0.016495  51.595151              1
```

[29]:

```
# Selection of features and labels
features = accidents[['day_of_week', 'first_road_class', 'road_type',␣
 ↪'speed_limit', 'second_road_class',
                      'pedestrian_crossing_physical_facilities',␣
 ↪'light_conditions', 'weather_conditions',
                      'road_surface_conditions', 'urban_or_rural_area']]
labels = accidents['cluster_label']

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(features, labels,␣
 ↪test_size=0.2, random_state=42)

# Creating a Random Forest Classifier Example
random_forest = RandomForestClassifier(n_estimators=100, random_state=42)

# Training Models
random_forest.fit(X_train, y_train)

# Prediction using test sets
y_pred = random_forest.predict(X_test)

# Output classification reports and accuracy
print(classification_report(y_test, y_pred))
print(f"Accuracy: {accuracy_score(y_test, y_pred):.2f}")
```

```
              precision    recall  f1-score   support

           0       0.48      0.60      0.54      2141
           1       0.32      0.20      0.25       963
           2       0.44      0.39      0.41      1590

    accuracy                           0.45      4694
   macro avg       0.41      0.40      0.40      4694
weighted avg       0.43      0.45      0.44      4694

Accuracy: 0.45
```

The Random Forest model exhibits moderate overall performance with an accuracy of 45%. It predicts Class 0 most effectively, evidenced by reasonable precision and recall. Class 2 displays moderate predictive accuracy with balanced metrics. However, Class 1 demonstrates poor predictive capabilities.

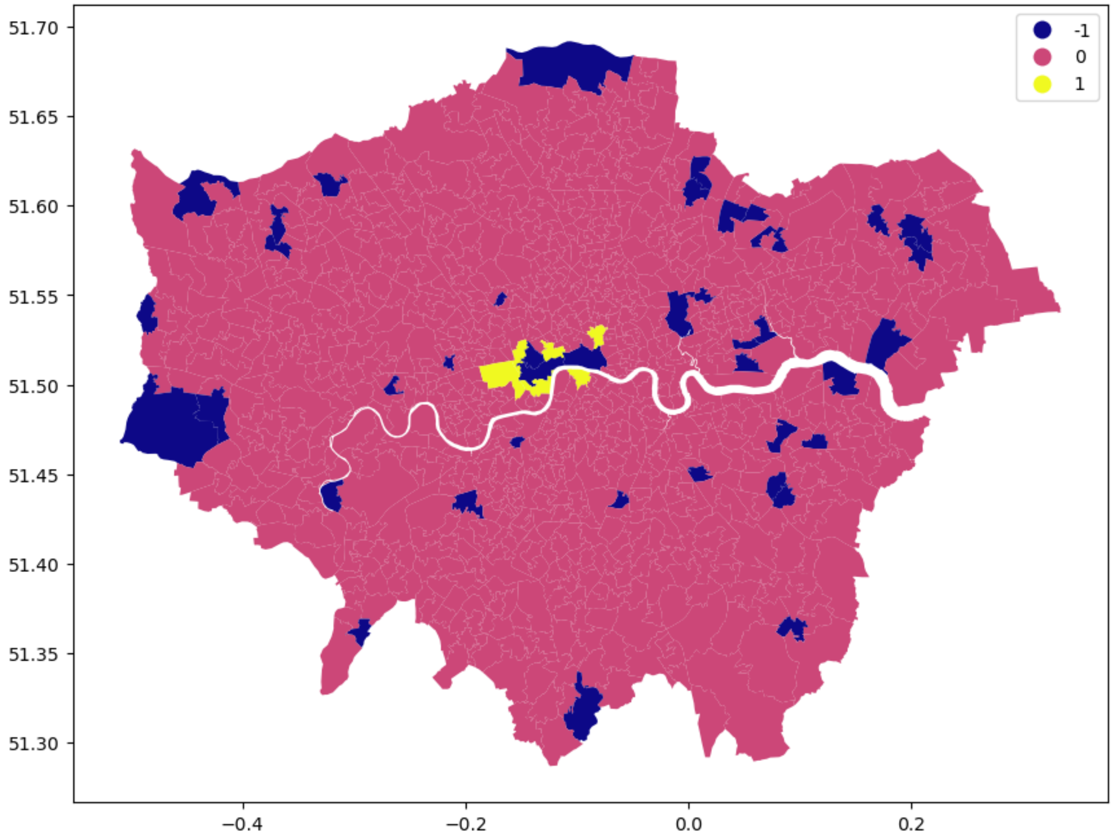
1.7 6 Results and Discussion

1.7.1 6.1 DBSCAN

The DBSCAN clustering results, as depicted on the geographic distribution map, show cluster '0' extensively throughout London, indicating a ubiquitous occurrence of traffic accidents across various districts. In stark contrast, the sparse instances of cluster '1' deviate significantly from the urban norm, potentially pointing to areas with unique traffic dynamics or a higher propensity for severe incidents. Noise points, labeled as '-1', represent outliers that diverge from common accident patterns, calling for individualized analysis to understand their unique contexts.

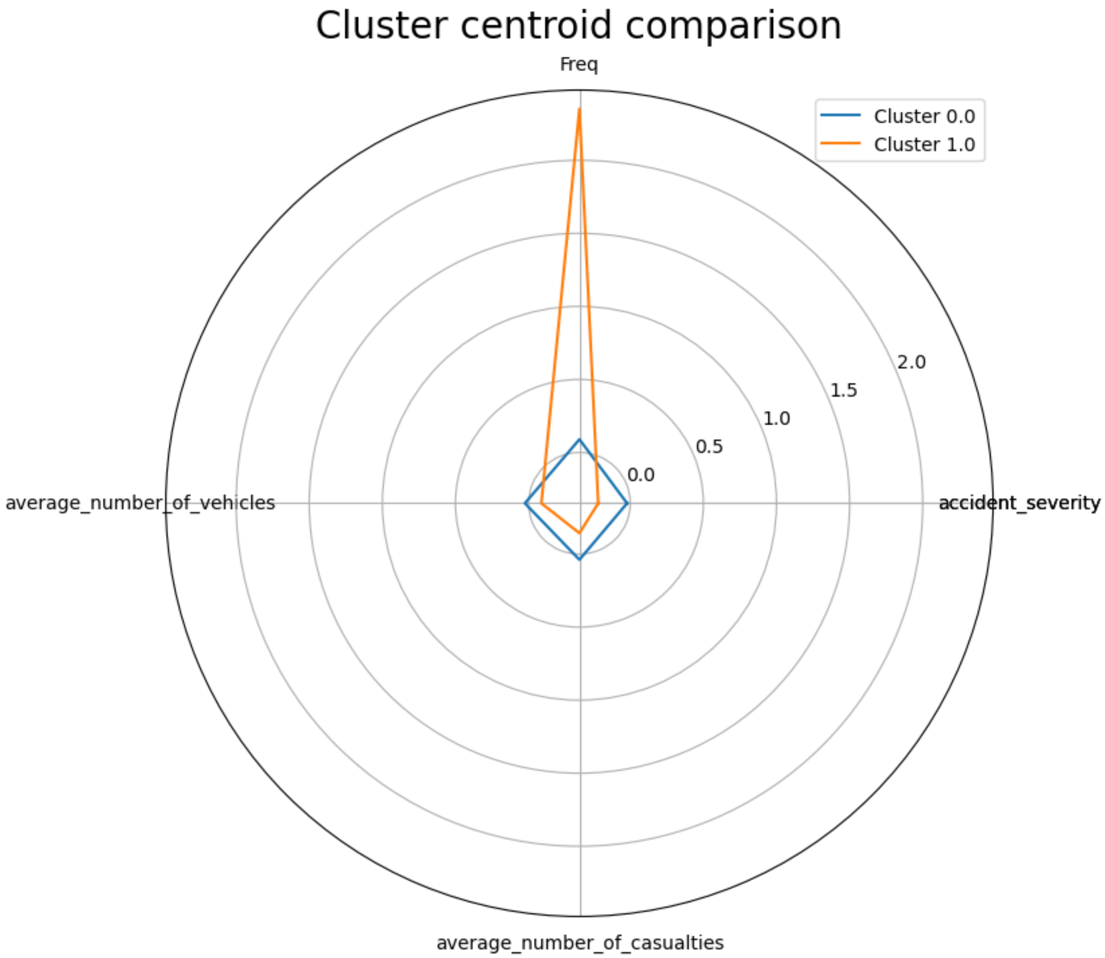
[30]:
```
dfmg.plot(column='DBSCAN_Cluster', categorical=True, legend=True,␣
 ↪figsize=(12,8), cmap = 'plasma')
```

[30]:
```
<Axes: >
```



The radar chart offers a comparative visualization of the centroids for clusters '0' and '1', accentuating the pronounced profile of cluster '1' with its exceptionally high frequency of occurrences—essentially, the aforementioned 'black spots' that markedly exceed the usual incident rate of cluster '0'. These areas emerge as pivotal targets for the enhancement of traffic safety measures and infrastructural planning.

[31]:
```
radar_plot_cluster_centroids1(df_dbscan_centroid1)
```



Upon meticulous scrutiny of the cluster table, the noise cluster '-1' is discerned to comprise accidents with escalated severity, an increased number of involved vehicles, and higher casualties. This anomaly could signify scenarios of augmented hazard, albeit less frequent yet bearing substantial risk and impact, which mandates the formulation of specific remedial measures.

[32]:
```
summary_df_dbscan
```

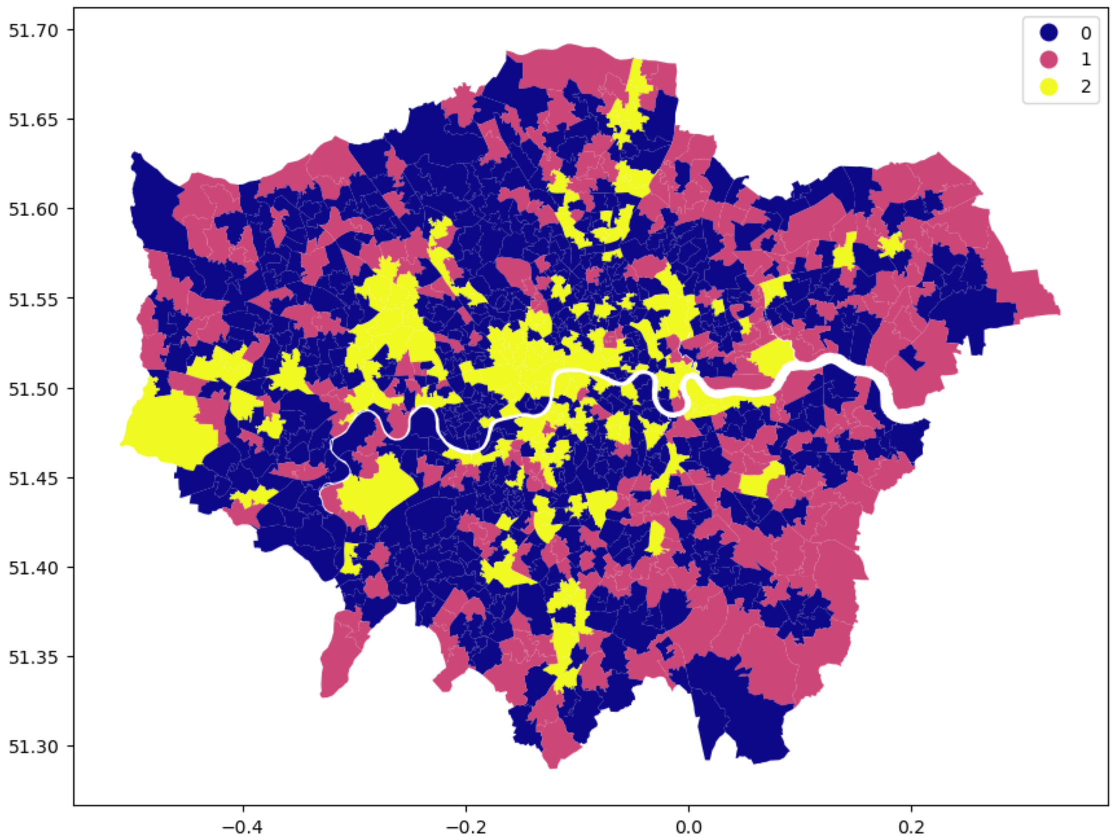
[32]:
```
                accident_severity    Freq  average_number_of_vehicles  \
DBSCAN_Cluster
-1                           2.79   28.86                        1.87
 0                           2.84   23.15                        1.82
 1                           2.79  102.33                        1.78

                average_number_of_casualties
DBSCAN_Cluster
-1                                      1.46
 0                                      1.16
```

```
1                                          1.09
```

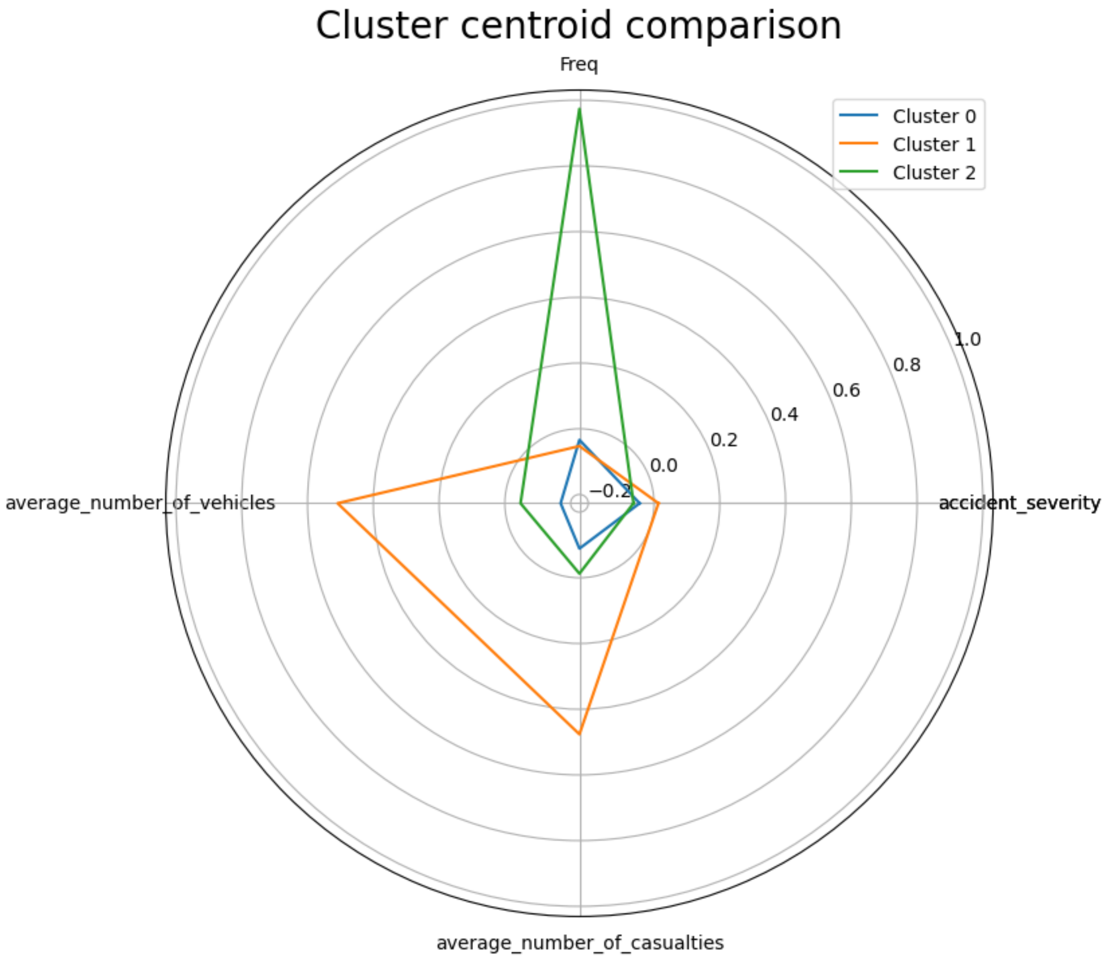
1.7.2 6.2 Kmeans

The K-means clustering map offers an intricate portrayal of traffic accidents across London, dividing them into three principal clusters. Cluster '0', pervasive across the metropolis, signifies a consistent pattern of urban traffic incidents. Alternatively, the infrequent yet pronounced Cluster '1' may reflect specific anomalous conditions, with its increased vehicle and casualty averages pointing to more grave incidents. Cluster '2' stands out with its heightened accident rate, signaling areas in dire need of targeted safety measures.

[33]:
```
fig_kmeans_4 = dfmg.plot(column='Kmeans_Cluster', categorical=True,␣
 ↪legend=True, figsize=(12,8),cmap = 'plasma')
```



[34]:
```
radar_plot_cluster_centroids2(df_cluster_centroid2)
```



Both the radar chart and the tabular data enhance our understanding by quantifying each cluster's characteristics. They suggest that Cluster '0', despite its lower frequency of accidents, does not necessarily correlate with lower severity or fewer casualties. This observation contradicts common perceptions about the frequency and impact of accidents. Cluster '1', with its heightened vehicle and casualty rates, suggests a propensity for more severe accidents, necessitating tailored traffic management strategies. Cluster '2', with its acute frequency of incidents, highlights "black spots" that urgently require road safety advancements and infrastructural changes to avert future incidents.

[35]:
```
summary_df_Kmeans
```

[35]:
```
                accident_severity   Freq  average_number_of_vehicles  \
Kmeans_Cluster
0                            2.83  18.81                        1.75
1                            2.85  18.16                        2.01
2                            2.83  54.08                        1.80

                average_number_of_casualties
```

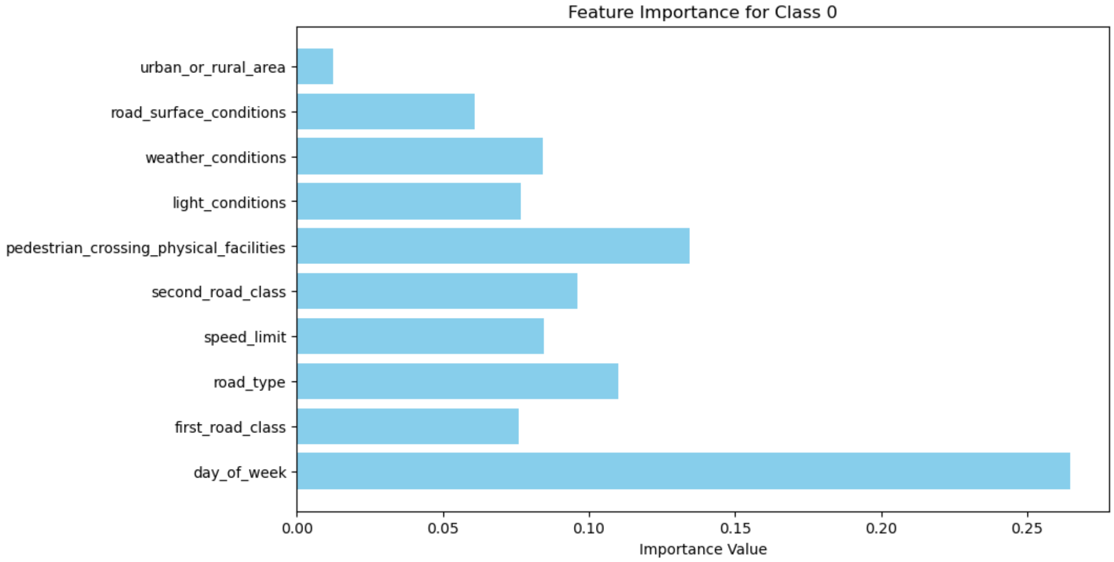
```
Kmeans_Cluster
0                                        1.11
1                                        1.31
2                                        1.14
```

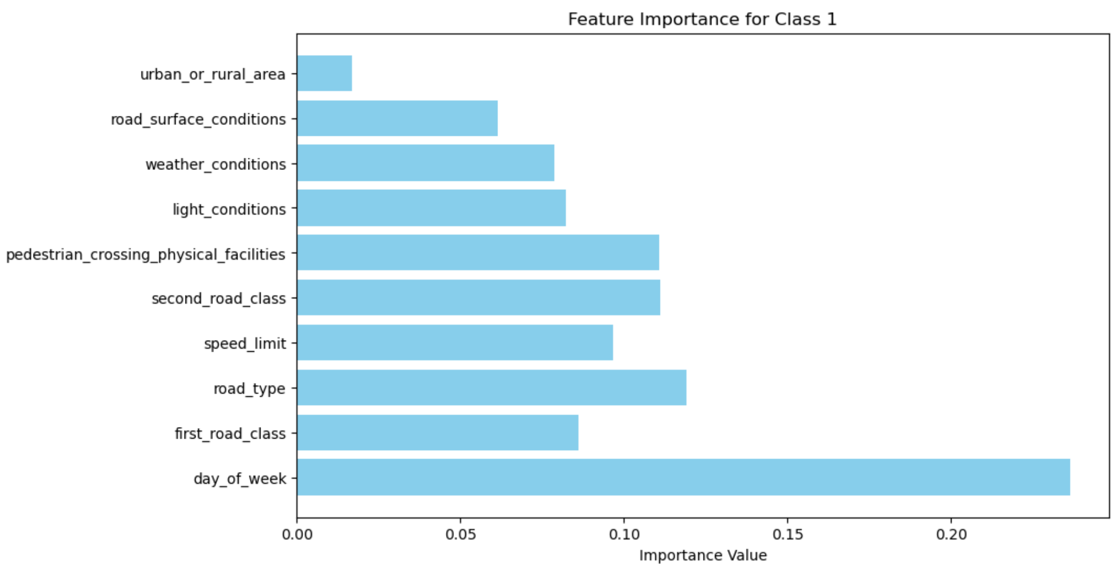
1.7.3 6.3 RF

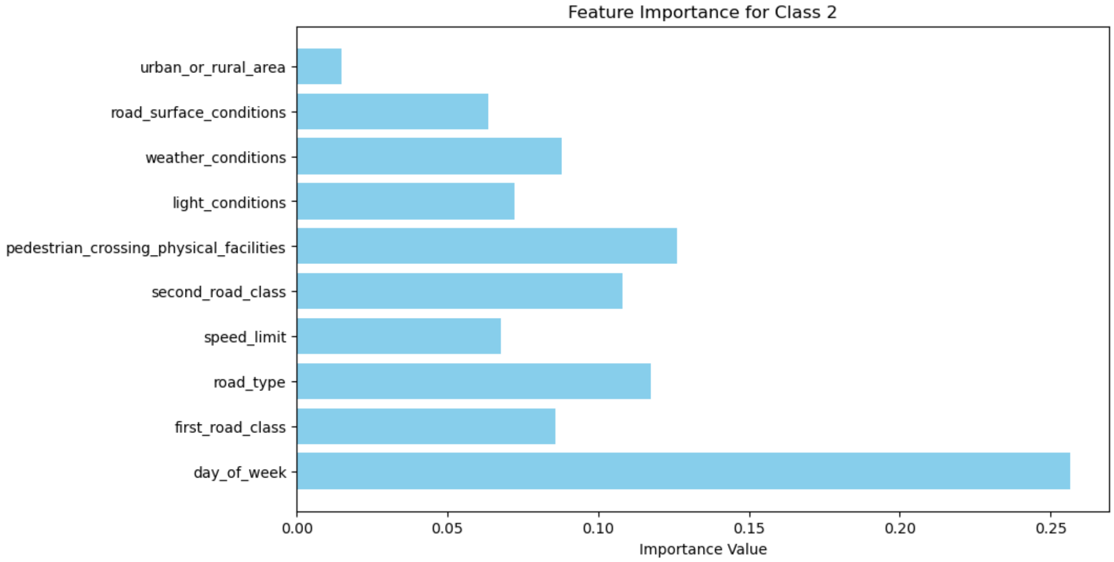
The feature importance charts for a Random Forest model reveal key factors influencing the classification of traffic accidents into Kmeans clusters:

- For Class 0, the timing of accidents ('day of the week') is paramount, while the type of road plays a secondary role. The setting ('urban or rural area') seems negligible, suggesting a uniform risk across environments.
- Class 1 is dominated by the type of road and the speed limit, hinting that structural road features and traffic regulation significantly affect accident classification.
- In Class 2, visibility ('light conditions') is the most critical factor, with environmental conditions ('weather' and 'road surface') also being influential.

[36]:
```
# Further analysis of which features are most critical to the prediction of␣
 ↪cluster labels

# Initialize OVR classifier using Random Forest
ovr_classifier = OneVsRestClassifier(random_forest)
ovr_classifier.fit(X_train, y_train)

# Plot feature importances for each class
for i, estimator in enumerate(ovr_classifier.estimators_):
    plt.figure(figsize=(10, 6))
    plt.barh(features.columns, estimator.feature_importances_, color='skyblue')
    plt.xlabel('Importance Value')
    plt.title(f'Feature Importance for Class {ovr_classifier.classes_[i]}')
    plt.show()
```


Feature Importance for Class 0
urban_or_rural_area
road_surface_conditions
weather_conditions
light_conditions
pedestrian_crossing_physical_facilities
second_road_class
speed_limit
road_type
first_road_class
day_of_week
0.00
0.05
0.10
0.15
0.20
0.25
Importance Value


Feature Importance for Class 1
urban_or_rural_area
road_surface_conditions
weather_conditions
light_conditions
pedestrian_crossing_physical_facilities
second_road_class
speed_limit
road_type
first_road_class
day_of_week
0.00
0.05
0.10
0.15
0.20
Importance Value



The feature importance from the Random Forest model highlights the primacy of temporal and environmental conditions over geographical settings in predicting traffic accident clusters. 'Day of the week' and 'light conditions' emerge as critical indicators, overshadowing 'urban or rural' distinctions. This suggests that policy interventions may benefit from focusing on when and under what conditions accidents occur, rather than solely on where they take place. The data advocates for targeted safety measures at times and in conditions most conducive to accidents, potentially offering a more nuanced approach to traffic safety strategies.

1.8 7 Conclusion

In conclusion, this study's comprehensive analysis has identified and characterized traffic collision 'black spots' in London. Through the integration of spatial clustering and supervised learning methods, we have discerned the influence of temporal, environmental, and infrastructural factors on accident occurrences. The research underscores the complexity of urban traffic accidents and reinforces the critical need for targeted safety interventions, transcending traditional urban-rural distinctions. This study contributes to the overarching goal of enhancing road safety and mitigating the socio-economic impacts of traffic-related injuries and fatalities.

1.9 References

Anderson, T. (2005) 'Spatial variations in road collision propensities in London' [Online]. Available at: https://discovery.ucl.ac.uk/id/eprint/1266 (Accessed: 12 April 2024).

Aziz, S. and Ram, S. (2022) 'A Meta-analysis of the methodologies practiced worldwide for the identification of Road Accident Black Spots', Transportation Research Procedia, 62, pp. 790-797.

Balawi, M. & Tenekeci, G. (2024) Time series traffic collision analysis of London hotspots: Patterns, predictions and prevention strategies. Heliyon, Elsevier BV, pp.e25710-e25710.

Karamanlis, I., Nikiforiadis, A., Botzoris, G., Kokkalis, A. and Basbas, S., 2023. Towards sustainable transportation: The role of black spot analysis in improving road safety. Sustainability, 15(19), p.14478.

Siamidoudaran, M. and Iscioglu, E., 2019. Injury severity prediction of traffic collision by applying a series of neural networks: The City of London case study. Promet – Traffic & Transportation, 31(6), pp.643-654.

Szénási, S. and Csiba, P., 2014. Clustering algorithm in order to find accident black spots identified by GPS coordinates. In: 14th SGEM GeoConference on Informatics, Geoinformatics and Remote Sensing, 17-26 June 2014, Bulgaria. Available at: <https://www.researchgate.net/publication/264635564

UNEP (2003) Technical Guidelines for the Environmentally Sound Management of Waste Lead-Acid Batteries. Basel Convention Series/SBC No. 2003/9. Geneva, Switzerland: Basel Convention Secretariat. Available at: http://www.basel.int/Portals/4/Basel%20Convention/docs/pub/techguid/techwasteacid.pdf (Accessed: 12 April 2024).