COMP5721M: Programming for Data Science

Coursework 3: Data Analysis Project

UK Accidents and Casualties

- Kriti Garg, mm22kg@leeds.ac.uk (mailto:mm22kg@leeds.ac.uk)
- Raghav Arora, mm22ra@leeds.ac.uk (mailto:mm22ra@leeds.ac.uk)
- Baha Abdul Vahab, mm22bura@leeds.ac.uk (mailto:mm22bura@leeds.ac.uk)
- Kartikeya Malimath, sc22kpm@leeds.ac.uk (mailto:sc22kpm@leeds.ac.uk)

Project Plan

The Data

Anything which has a merit, its not necessary that there can't be any demerit of it. For eg: In case of Vehicles, Pedestrian Crossing, Road Safety seems like shouldn't have any problems because they have there existence to save one's time, provide services and safety. That is why we have selected UK Accidents and Casualties 2011 and 2012 data to analyse what problems are faced at the massive level and at what regions. We are analysing 2 datasets, First one is the main set which contains all the information about the location and number of accidents, casualties and the surrounding during the accidents, Second One is the mapping Excel which comprises of the keys and value for the later columns, For eg: for Accident Severity we have keys 1, 2 and 3 and in the mapping Excel we have values for it i.e. Fatal, Serious and Slight.

We have collected the data from Kaggle (https://www.kaggle.com/datasets/devansodariya/road-accident-united-kingdom-uk-dataset?select=UK_Accident.csv) and data.gov.uk (https://www.data.gov.uk/dataset/cb7ae6f0-4be6-4935-9277-47e5ce24a11f/road-safety-data) which is accumulated by the UK government from the traffic data. Unexpectedly, the data is almost clean and looks good to work upon directly.

The below table describes the key columns of our first dataset and their meaning:

Column Name	Description	Data Type
Accident_Index	unique identifier of an accident(combination of year of accident and accident reference)	object
Longitude	Longitude of the place where accident take place	float64
Latitude	Latitude of the place where accident take place	float64
Police_Force	The Police on whose shoulders the responsibility for investigating the accident is	int64(1-98)
Accident_Severity	The Severity of the accident (Fatal, Serious, Slight)	int64(1,2,3)
Number_of_Vehicles	Total Number of Vehicles involved in the accident	int64
Number_of_Casualties	Total Number of casualties due to the accident	int64
Date	The date of the accident	object
Day_of_Week	the day of the week(1-7{Monday-Sunday})	int64
Local_Authority_(District)	Under which district the accident take place	int64
Local_Authority_(Highway)	Under which Highway the accident take place	object

Column Name	Description	Data Type
Road_Type	Type of the road i.e. singleway or dualway etc	object
Pedestrian_Crossing-Human_Control	By whom the Pedestrian crossing was controlled or by none	object
Pedestrian_Crossing-Physical_Facilities	By which the Crossing was controlled such as has zebra crossing, had pedestrian junction at traffic signal etc	object
Light_Conditions	Was it dark or was during day light	object
Weather_Conditions	Conditions such as if there was fog, or was fine, was raining or snowing etc	object
Road_Surface_Conditions	Condition og the road such as dry, flooded, wet, Normal, Frost/Ice, Snow etc	object
Special_Conditions_at_Site	any Special conditions such as Mud, oil or diesel, Construction going on etc	object
Carriageway_Hazards	Anything or person that that can be a reason for the accident i.e. pedestrian or an animal etc	object
Urban_or_Rural_Area	Contains 1,2,3 - will be mapped with mapping Excel to show if it is Urban, Rural or Unclassified	object
Did_Police_Officer_Attend_Scene_of_Accident	Contains Yes or No	object
Year	Year in which accident happen, in our case it will be 2011 or 2012	int64

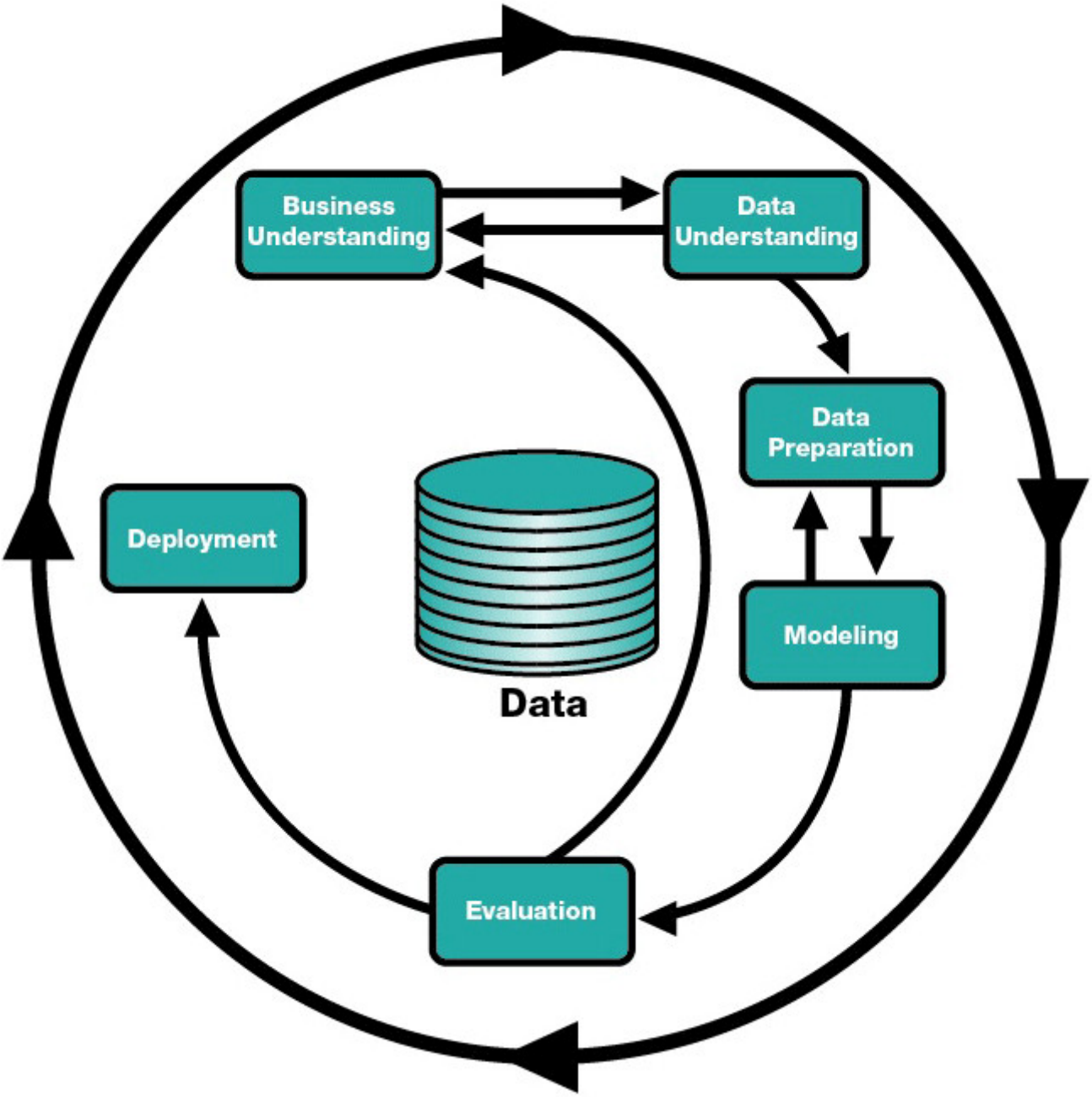
Below are the key columns of mapping Excel:

Column Name	Description	Format	Label
Field Name	Name of the column present in First Dataset	Key	Value

Project Aim and Objectives

The major aim of our project is to identify what changes should be brought by an individual and by the govt so that the number of accidents can be decreased. The highways, carriageways, police, local authorities, Pedestrian Crossing are all made for the safety of the people and if in case of failure in 1 such thing there are accidents happening, those reasons should be addressed so that the proper measures can be taken. By this Road Accidents data we are trying to address the following:

- major reasons behind the fatal/severe accidents i.e. if they are due to human error or lack of physical facilities.
- the top cities which have the maximum number of accidents and the top cities that are safest to live i.e. which have lowest number of accidents happened.



- *Classification: separate data items into classes according to their charactersitics (can be either a definite or a statistical kind of classification)*
- *Corellation: find correspondences between different attributes within a dataset*
- *Search: find solutions matching some criteria*
- *Visualisation: find informative ways to display the structure of a large and/or complex dataset*
- *Query Answering: create a system that enables one to retrieve information by evaluating some form of query representation*
- *Simulation: model the evolution of a complex process Here you should describe the general aim of your project in around 200-300 words.*

This can can be anything from classifying items according to their characteristic features (which mushrooms are poisonous?) to simulating an evolving process (will the rabbits eat all the carrots or get eaten by the foxes?)

Here some ideas of general types of processing functionality that you could implement:

- *Classification: separate data items into classes according to their charactersitics (can be either a definite or a statistical kind of classification)*
- *Corellation: find correspondences between different attributes within a dataset*
- *Search: find solutions matching some criteria*

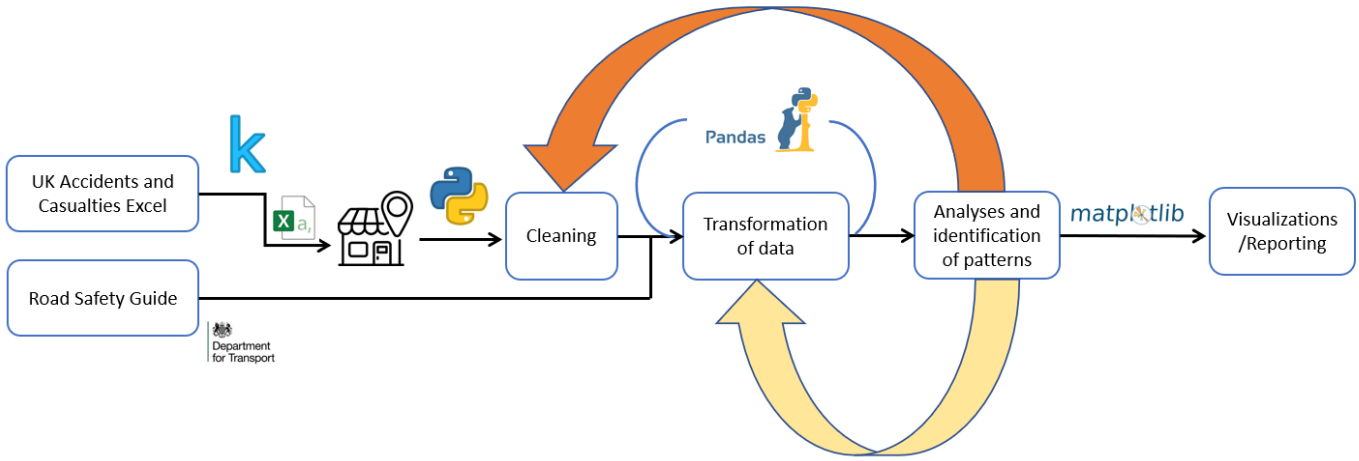
- *Visualisation: find informative ways to display the structure of a large and/or complex dataset*
- *Query Answering: create a system that enables one to retrieve information by evaluating some form of query representation*
- *Simulation: model the evolution of a complex process*

Specific Objective(s)

- **Objective 1:**To identify the highest number of casualties in each year and their potential reasons and how the police responds at the scene of accidents?
- **Objective 2:**What are the top 10 cities that had the maximum number of casualties?
- **Objective 3:**To identify the relationship between the reasons that are affecting the accidents and the number of casualties over the year and their trend. Additionally we are trying to understand the police response as per the reason
- **Objective 4:**What is the percentage change of the number of accidents coressponding to the cities and the major physical reasons such as Road Type, urban or rural?

System Design

Architecture



At first comes the datasets which are required for analysis i.e. the UK Accidents/Casualties and Road Safety Data Guide(Mapping Excel) which then are combined on the basis of the FieldName and columns to populate the true value of the keys provided. Once the data is merged using left join from Accidents to Mapping Excel the data is cleaned i.e. the unimportant columns are dropped. Transformation of data is done by updating the datatypes or, filling the missing values or, filtering the 2 years of data. After this Analysis is done to identify the correlation between the features and if there is any intersting patterns among them. As we move forward with the analysis there are times when we have to transform the data again or even perform the cleaning task again. Visualizations are created once we feel the analysis seems appropriate.

Processing Modules and Algorithms

- **Cleaning:**
  - Removed the following columns as were not required because there was lack of explanation.
    - `Location_Easting_OSGR` , `Location_Northing_OSGR` , `1st_Road_Class` , `1st_Road_Number` , `Junction_Control` , `2nd_Road_Class` , `2nd_Road_Number` , `LSOA_of_Accident_Location` , `Time` , `Unnamed(0)` (This was coming because of the first row as a rownumber in Excel)

In [80]:
```
data.head()
```

Out[80]:

	Unnamed: 0	Accident_Index	Location_Easting_OSGR	Location_Northing_OSGR	Longitude	Latitude	Police_Force	Accident_Severity	Number_of_Vehicles	N
0	0	200501BS00001	525680.0	178240.0	-0.191170	51.489096	1	2	1	
1	1	200501BS00002	524170.0	181650.0	-0.211708	51.520075	1	3	1	

- **Joining of Datasets:**
  - We have only 1 main dataset i.e. UK Road Accidents but for some columns such as **Accident Severity**, **Urban_or_Rural_Area**, **Local_Authority_(District)**, we don't have the mapping of the values for eg: what does 1 as severity means or 2 as Urban_or_Rural Means. So we Downloaded a mapping Excel from data.gov.uk site and added the mapping for these columns.
- **Transformation:**
  - Updated the data types of Date column from object to Datetime64
- **Analysis and Visualization:**
  - Perform Analysis to identify the regions with massive number of casulties, what combinations of reasons are the main cause of accidents taking place, Accidents of what severity happened at what regions, Percentage of Accidents happened over the Fatal Accidents and their plot their Visualizations.

Program Code

Importing Libraries which we will be using in this project

In [1]:

```
## Libraries
import pandas as pd
import matplotlib.pyplot as plt
import datetime
import numpy as np
import seaborn as sns
import decimal
from matplotlib.patches import ConnectionPatch
from matplotlib.gridspec import GridSpec
import calendar
import matplotlib.patches as mpatches
```

Reading first dataset 'UK_Accident.csv' in a `DataFrame` using `pandas`

In [2]:

```
uk_acc_all_data = pd.read_csv("UK_Accident.csv")
```

The following cell performs cleaning of the dataset by removing the unwanted columns

In [3]:

```
uk_acc_all_data.drop(columns=['LSOA_of_Accident_Location',
                              'Location_Easting_OSGR',
                              'Location_Northing_OSGR',
                              '1st_Road_Class',
                              '1st_Road_Number',
                              'Junction_Control',
                              '2nd_Road_Class',
                              '2nd_Road_Number',
                              'Time',
                              'Unnamed: 0'], inplace = True)
```

Filtering out the data for 2011 and 2012 in which we want to do analysis

In [4]:

```
uk_filter_data = uk_acc_all_data[uk_acc_all_data["Year"].isin([2011,2012])]
```

For our Analysis and Visualizations we require 3 new columns and these will be added to the uk_filter_data `DataFrame`

- Month - Extracting from Date column and converting the argument to numeric type
- Reason - Combining the values of `Road_Surface_Conditions` , `Light_Conditions` and `Weather_Conditions`
- First_day_of_month - Extracting the month/year and creating generating the first day of every month and year

In [5]:

```
uk_filter_data["month"] = pd.to_numeric(uk_filter_data["Date"].str[3:5])
uk_filter_data["reason"] = (uk_filter_data["Road_Surface_Conditions"]+"+"+uk_filter_dat
uk_filter_data["First_day_of_month"]=pd.to_datetime(uk_filter_data["month"].astype(str)
```

...

The below command shows the sum of all the null values if any

In [6]:

```
1 uk_filter_data.isnull().sum()
```

Out[6]:

```
Accident_Index                                0
Longitude                                     0
Latitude                                      0
Police_Force                                  0
Accident_Severity                             0
Number_of_Vehicles                            0
Number_of_Casualties                          0
Date                                          0
Day_of_Week                                   0
Local_Authority_(District)                    0
Local_Authority_(Highway)                     0
Road_Type                                     0
Speed_limit                                   0
Pedestrian_Crossing-Human_Control             0
Pedestrian_Crossing-Physical_Facilities       0
Light_Conditions                              0
Weather_Conditions                            0
Road_Surface_Conditions                       0
Special_Conditions_at_Site                    0
Carriageway_Hazards                           0
Urban_or_Rural_Area                           0
Did_Police_Officer_Attend_Scene_of_Accident    0
Year                                          0
month                                         0
reason                                        0
First_day_of_month                            0
dtype: int64
```

From the above command's result we can see that there is no null value coming in data now.

Reading second dataset into a `DataFrame` using `pandas` read_excel function

In [7]:

```
1 #importing a new dataset- dim table
2 data_guide = pd.read_excel("Road-Safety-Open-Dataset-Data-Guide.xlsx")
```

Creating 3 different Dataframes for 3 seperate columns for whose key's, the values needs to be populated. The below cell is mapping values for `local_authority_district` column

In [8]:

```
1 df_lad_mapping = data_guide[data_guide["field name"] == 'local_authority_district'][["
2                 .rename(columns = {"code/format":"Local_Authority_(District)","label":"
```

The below cell is mapping values for `urban_or_rural_area` column

In [9]:

```
df_uor_mapping = data_guide[data_guide["field name"] == 'urban_or_rural_area'][["code/f
                 .rename(columns = {"code/format":"Urban_or_Rural_Area","label":"Urban_c
```

The below cell is mapping values for `casualty_severity` column

In [10]:

```
df_sev_mapping = data_guide[data_guide["field name"] == 'casualty_severity'][["code/for
                 .rename(columns = {"code/format":"Accident_Severity","label":"Accident_
```

Left Joining the above 3 `DataFrame's` to the uk_filter_data insert the values of the column in the DataFrame

In [11]:

```
left1 = pd.merge(uk_filter_data, df_lad_mapping, on = 'Local_Authority_(District)', how
left2 = pd.merge(left1,df_uor_mapping,on = 'Urban_or_Rural_Area', how = 'left')
left3 = pd.merge(left2,df_sev_mapping,on = 'Accident_Severity', how = 'left')
uk_filter_data = left3
```

Checking is the data types of all the columns are as expected.

In [12]:

```
1 uk_filter_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 331189 entries, 0 to 331188
Data columns (total 29 columns):
 #   Column                                       Non-Null Count   Dtype
---  ------                                       --------------   -----
 0   Accident_Index                               331189 non-null  object
 1   Longitude                                    331189 non-null  float64
 2   Latitude                                     331189 non-null  float64
 3   Police_Force                                 331189 non-null  int64
 4   Accident_Severity                            331189 non-null  object
 5   Number_of_Vehicles                           331189 non-null  int64
 6   Number_of_Casualties                         331189 non-null  int64
 7   Date                                         331189 non-null  object
 8   Day_of_Week                                  331189 non-null  int64
 9   Local_Authority_(District)                   331189 non-null  object
 10  Local_Authority_(Highway)                    331189 non-null  object
 11  Road_Type                                    331189 non-null  object
 12  Speed_limit                                  331189 non-null  int64
 13  Pedestrian_Crossing-Human_Control            331189 non-null  object
 14  Pedestrian_Crossing-Physical_Facilities      331189 non-null  object
 15  Light_Conditions                             331189 non-null  object
 16  Weather_Conditions                           331189 non-null  object
 17  Road_Surface_Conditions                      331189 non-null  object
 18  Special_Conditions_at_Site                   331189 non-null  object
 19  Carriageway_Hazards                          331189 non-null  object
 20  Urban_or_Rural_Area                          331189 non-null  object
 21  Did_Police_Officer_Attend_Scene_of_Accident  331189 non-null  object
 22  Year                                         331189 non-null  int64
 23  month                                        331189 non-null  int64
 24  reason                                       331189 non-null  object
 25  First_day_of_month                           331189 non-null  datetime6
4[ns]
 26  Local_Authority_District_value               331189 non-null  object
 27  Urban_or_Rural_Area_value                    331189 non-null  object
 28  Accident_Severity_value                      331189 non-null  object
dtypes: datetime64[ns](1), float64(2), int64(7), object(19)
memory usage: 75.8+ MB
```

As can be seen that `Date` and `Severity` is of `Object` dtype. Updating the data type of `Date` to datetime and `Accident Severity` to int

In [13]:

```
1 uk_filter_data["Date"] = pd.to_datetime(uk_filter_data["Date"])
2 uk_filter_data["Accident_Severity"] = pd.to_numeric(uk_filter_data["Accident_Severity"]
```

...

Objective 1:

Creating a `DataFrame` df, grouping up on the basis of the new column `First_day_of_month` created and aggregating on `Number_of_Casualties`

In [15]:

```
df = uk_filter_data.groupby(["First_day_of_month"])["Number_of_Casualties"].agg(['sum'
```

```
Below cell creates another Dataframe defining the `First_day_of_month` along with
the casualties corresponding to the `Reason`
```

In [16]:

```
uk_bar_graph_data_natural = uk_filter_data\
                            .groupby(["First_day_of_month","reason"])["Number_of_Casual
                            .sum().reset_index()
```

Fetching the top 5 reasons resulting in highest number of casualties for `2011` and `2012` years separately

In [17]:

```
# taking tops resons to plot the data in pie chart of 2011
df_2011 = uk_bar_graph_data_natural[uk_bar_graph_data_natural["First_day_of_month"]==da
                                    .sort_values("Number_of_Casualties",ascending=False).he

# taking tops resons to plot the data in pie chart of 2012
df_2012 = uk_bar_graph_data_natural[uk_bar_graph_data_natural["First_day_of_month"]==da
                                    .sort_values("Number_of_Casualties",ascending=False).he
```

Below command defines a `DataFrame` df_bar_police that tell us did the poilce attended the accident or not.

In [18]:

```
df_bar_police = uk_filter_data.groupby(["First_day_of_month","Did_Police_Officer_Attend
```

Below function converts the number to percentages

In [19]:

```
def func(pct, allvals):
    absolute = int(pct/100.*np.sum(allvals))
    return "{:.1f}%\n({:d})".format(pct, absolute)
```

The below cell defines the function which will be called for plotting visuals for first objective i.e. trend of accidents and casualties with the year and month using Gridspec. The trends will be shown with line chart and the percentage of reasons

In [20]:

```
def plot_obj1():
    fig = plt.figure(figsize=(15,15))
    #using gridspec to create the visual in the single space
    gs = GridSpec(2, 2, figure=fig)
    #plotting for the first visual starting
    ax1 = fig.add_subplot(gs[0, :])
    ax1.set_title('Accident & Causality VS yearmonth')
    l1 = ax1.plot("First_day_of_month","sum",data=df,marker='o',markersize = 4,label="#
    ax_new = ax1.twinx()
    l2 = ax_new.plot("First_day_of_month","count",data=df,marker='o',markersize = 4,col
    #adding legends and label for 1st graph
    lns = l1+l2
    labs = [l.get_label() for l in lns]
    ax1.legend(lns,labs)
    ax1.set_xlabel("YearMonth")
    ax1.set_ylabel("Causality")
    ax_new.set_ylabel("Accidents")
    #second graph plot start
    ax2 = fig.add_subplot(gs[1, :-1])
    ax2.set_title('Top 5 Reasons for Oct-2011')
    reason1 = df_2011["reason"].to_list()
    data1 = df_2011["Number_of_Casualties"].to_list()
    wedges, texts, autotexts  = ax2.pie(data1,autopct=lambda pct: func(pct, data1))
    ax2.legend(wedges, reason1,
              title="Reasons",
              loc="center left",
              bbox_to_anchor=(0,-0.1,1,0)
              )
    #using connection patch to create the arrow from 1st graph to second
    con = ConnectionPatch(  xyA=(datetime.date(2011, 10, 1), 18479),
                            coordsA="data",
                            xyB=(0,1),
                            coordsB=ax2.transData,
                            axesA=ax1,
                            axesB=ax2,
                            arrowstyle="-|>", shrinkA=5, shrinkB=5,
                            mutation_scale=20, fc="w")
    #displaying the arrow
    ax2.add_artist(con)
    #3rd plot start in gridspec
    ax5 = fig.add_subplot(gs[1:2,1])
    ax5.set_title('Top 5 Reasons for Oct-2012')
    reason4 = df_2012["reason"].to_list()
    data4 = df_2012["Number_of_Casualties"].to_list()
    wedges, texts, autotexts  = ax5.pie(data4,autopct=lambda pct: func(pct, data4))
    ax5.legend(wedges, reason4,
              title="Reasons",
              loc="center left",
              bbox_to_anchor=(0, -0.1, 0, 0)
             )
    #using connection patch to get the arrow from 1st visual to 3rd visual
    con = ConnectionPatch(  xyA=(datetime.date(2012, 10, 1), 21472),
                            coordsA="data",
                            xyB=(0,1),
                            coordsB=ax5.transData,
                            axesA=ax1,
                            axesB=ax5,
                            arrowstyle="-|>", shrinkA=5, shrinkB=5,
                            mutation_scale=20, fc="w")
```

```
    #displaying arrow
    ax5.add_artist(con)
```

In the below command we are ploting the data bettwen police resopose over the year

In [79]:

```
def plot_police():
    fig, ax = plt.subplots(figsize=(10,5))
    index = np.arange(24)
    labels = df_bar_police[df_bar_police["Did_Police_Officer_Attend_Scene_of_Accident"]
    bar_width = 0.4
    ax.bar(index-0.2, df_bar_police[df_bar_police["Did_Police_Officer_Attend_Scene_of_A
    ax.bar(index+0.2, df_bar_police[df_bar_police["Did_Police_Officer_Attend_Scene_of_A
    ax.plot(index,df["count"],bar_width)
    ax.legend(title = "police arrived on location",fontsize=7, title_fontsize=7)
    ax.set_xticks(index, labels, rotation=45, ha='right');
```

Objective 2:

Below command creates new Dataframe by grouping District, Reason, Severity and summing up by the number of casualties.

In [22]:

```
df_uk_region_reason = uk_filter_data.groupby(["Local_Authority_District_value","reason"
```

- The first dataframe contains the information of Number of Casualties per every region and is done by group by and sum python functions
- The second contains the top 10 regions which reported the highest number of casualties
- And the third contains the bottom 10 regions which reported the least number of casualties

In [23]:

```
#creating a region specific dataframe
df_uk_only_region = df_uk_region_reason.groupby(["Local_Authority_District_value"])["Nu
#identifiying top 10 and bottom 10 locations in uk
df_uk_top10_region = df_uk_only_region.sort_values(by=["Number_of_Casualties"],ascendir
df_uk_bottom10_region = df_uk_only_region[df_uk_only_region["Local_Authority_District_v
```

The below command takes the above grouped Dataframe and shows only the reason with the number of Casualties which then is used to identify the top 10 reasons that are resulting in the maximum number of Casualties.

In [24]:

```
# creating reason specific data frame
df_uk_only_reason = df_uk_region_reason.groupby(["reason"])["Number_of_Casualties"].sum
#identifiying top 10 reasons for the accidents
df_uk_top10_reason = df_uk_only_reason.sort_values(by=["Number_of_Casualties"],ascendin
```

In the below command we are fetching those records in the dataframe where those regions had the maximum number of casualities and then using this dataframe we are simultaneously fetching the records with the top 10 reasons of the mishappenings.

In [25]:

```
# joining reason and region data uk main data to get the final result
top_10_final_df=  pd.merge(df_uk_region_reason, df_uk_top10_region["Local_Authority_Dis
top_10_final_df = pd.merge(top_10_final_df,df_uk_top10_reason["reason"],on="reason",how
```

In the Above command we fetched the top regions and reasons and in the below we are getching the bottom 10 regions and reasons where the casualties were highest

In [26]:

```
df_uk_only_reason_bottom = df_uk_region_reason.groupby(["reason","Local_Authority_Distr
df_uk_bottom_reason = pd.merge(df_uk_only_reason_bottom,df_uk_bottom10_region["Local_Au
df_uk_bottom10_reason = df_uk_bottom_reason.groupby("reason",dropna=False)["Number_of_C
```

In the below command we are joining our all data of region and reason with the top 10 regions we are doing inner join this will filter our data and only keep the top 10 region data with all their corresponding reason

In the second command in the same df, now we are joining the data with the top 10 reasons for accidents with the previously filtered data, now finally we will only have the top 10 regions and corresponding to those region top 10 reasons for each region in the dataframe

In [27]:

```
bottom_10_final_df=  pd.merge(df_uk_region_reason, df_uk_bottom10_region["Local_Authori
bottom_10_final_df = pd.merge(bottom_10_final_df,df_uk_bottom10_reason["reason"],on="re
```

In the command below we are grouping, summing the data with the region and severity and getting a dataframe that has all the severity with the region

In the next statment we are joining it with the top 10 region that we identified earlier to only keep those regions

In [28]:

```
## top severity
df_uk_region_severity = df_uk_region_reason.groupby(["Local_Authority_District_value",'
#joining with top 10 regions to get the information
df_uk_region_severity_top10 = pd.merge(df_uk_region_severity,df_uk_top10_region["Local_
```

In the command below we are doing the same thing but with the bottom 10 of our list.

In [29]:

```
## bottom severirty
df_uk_region_severity_bottom10 = pd.merge(df_uk_region_severity,df_uk_bottom10_region['
df_uk_region_severity_bottom10.loc[-1] = ["Shetland Islands", 1,"Fatal",0]
```

In the command below, we are using the seaborn library to create a heatmap. We are explicitly providing the label for Xaxsis. then we are creating a legend table without any border that will give the full form of the abbreviation

In [30]:

```
def heatmap():
    heat= top_10_final_df.groupby(["Local_Authority_District_value","reason"])["Number_
    #creating label as the reason are too big to disply in the visual
    labelx = ["R1","R2","R3","R4","R5","R6","R7","R8","R9","R10"]
    # labely = heat.reset_index()["Local_Authority_District_value"].to_list()
    sns.heatmap(heat,cmap="RdBu",square = True,linewidth =1,xticklabels=labelx)
    cell_text= [[i] for i in top_10_final_df["reason"].drop_duplicates().to_list()]
    the_table = plt.table(cellText =cell_text ,rowLabels=labelx,colLabels=["Fullform"],
    the_table.auto_set_font_size(False)
    the_table.set_fontsize(8)
```

In this command we are creating the data set that will help us in creating the bar chart with table.

data - here we are creating the list of list that will be passed into the bar graph plot, and because we are creating a stack plot we need to creat a list of list.

columns - this is the xaxis/ the table header that will be displayed in the visual

rows - this the row heading for the table

In [77]:

```
data =  [df_uk_region_severity_top10[df_uk_region_severity_top10["Accident_Severity"]==
        df_uk_region_severity_top10[df_uk_region_severity_top10["Accident_Severity"]==2
        df_uk_region_severity_top10[df_uk_region_severity_top10["Accident_Severity"]==1
       ]
cell_text = [df_uk_region_severity_top10[df_uk_region_severity_top10["Accident_Severity
        df_uk_region_severity_top10[df_uk_region_severity_top10["Accident_Severity"]==2
        df_uk_region_severity_top10[df_uk_region_severity_top10["Accident_Severity"]==1
       ]
# col = df_uk_region_severity_top10[["Local_Authority_District_value"]].drop_duplicates
columns=('Barnet','Birmingham','Bradford','Cheshire East','Cheshire West','Cornwall','C
rows = df_sev_mapping["Accident_Severity_value"].to_list()
```

In the command below we are plotting a bar graph with the table at its bottom that will act as the legned for the x-axis of the bar graph and table header for the table graph

In [32]:

```
def CbyR():
    # Get some pastel shades for the colors -BuPu
    colors = plt.cm.Accent(np.linspace(0, 0.5, len(rows)))
    n_rows = len(data)
    index = np.arange(len(columns)) + 0.3
    bar_width = 0.4
    # Initialize the vertical-offset for the stacked bar chart.
    y_offset = np.zeros(len(columns))
    fig,ax = plt.subplots(figsize = (15,7))
    # Plot bars and create text labels for the table
    for row in range(n_rows):
        ax.bar(index, data[row], bar_width, bottom=y_offset, color=colors[row])
        y_offset = y_offset + data[row]
    # Reverse colors and text labels to display the last value at the top.
    colors = colors[::-1]
    cell_text.reverse()
    # Add a table at the bottom of the axes
    the_table = ax.table(cellText=cell_text,
                         rowLabels=rows,
                         rowColours=colors,
                         colLabels=columns,
                         loc='bottom')
    the_table.auto_set_font_size(False)
    the_table.set_fontsize(10)
    # Adjust layout to make room for the table:
    # plt.subplots_adjust(left=0.2, bottom=0.2)
    plt.ylabel("Number of Casulities")
    plt.xticks([])
    plt.title('Casulities by Region')
    plt.show()
```

same as heatmap1 but passing differnt data points

In [33]:

```
def heatmap2():
    heat_bottom= bottom_10_final_df.groupby(["Local_Authority_District_value","reason"]
    labelx = ["R1","R2","R3","R4","R5","R6","R7","R8","R9","R10"]
    # labely = heat.reset_index()["Local_Authority_District_value"].to_list()
    sns.heatmap(heat_bottom,cmap="seismic",square = True,linewidth =1,xticklabels=label

    cell_text= [[i] for i in bottom_10_final_df["reason"].drop_duplicates().to_list()]
    the_table = plt.table(cellText =cell_text ,rowLabels=labelx,colLabels=["Fullform"],
    the_table.auto_set_font_size(False)
    the_table.set_fontsize(8)
```

same as the above exlanation for bar chart with table data prepration

In [68]:

```
data1 =  [df_uk_region_severity_bottom10[df_uk_region_severity_bottom10["Accident_Sever
        df_uk_region_severity_bottom10[df_uk_region_severity_bottom10["Accident_Severit
        df_uk_region_severity_bottom10[df_uk_region_severity_bottom10["Accident_Severit
       ]

cell_text1=  [df_uk_region_severity_bottom10[df_uk_region_severity_bottom10["Accident_S
        df_uk_region_severity_bottom10[df_uk_region_severity_bottom10["Accident_Severit
        df_uk_region_severity_bottom10[df_uk_region_severity_bottom10["Accident_Severit
       ]

col = df_uk_region_severity_bottom10[["Local_Authority_District_value"]].drop_duplicate
columns = tuple(col["Local_Authority_District_value"])
# ('Barnet','Birmingham','Bradford','Cheshire East','Cheshire West','Cornwall','County
rows = df_sev_mapping["Accident_Severity_value"].to_list()
```

code for 2nd table with bar chart- explantion same as the first code

In [69]:

```
def CbyR_best():
    # Get some pastel shades for the colors -BuPu
    # colors = plt.cm.Pastel2(np.linspace(0, 0.5, len(rows)))
    colors = plt.cm.tab20(np.linspace(0, 0.5, len(rows)))
    n_rows = len(data1)
    index = np.arange(len(columns)) + 0.3
    bar_width = 0.4
    # Initialize the vertical-offset for the stacked bar chart.
    y_offset = np.zeros(len(columns))
    fig,ax = plt.subplots(figsize = (15,7))
    # Plot bars and create text labels for the table
    for row in range(n_rows):
        ax.bar(index, data1[row], bar_width, bottom=y_offset, color=colors[row])
        y_offset = y_offset + data1[row]
    # Reverse colors and text labels to display the last value at the top.
    colors = colors[::-1]
    cell_text1.reverse()
    # Add a table at the bottom of the axes
    the_table = ax.table(cellText=cell_text1,
                         rowLabels=rows,
                         rowColours=colors,
                         colLabels=columns,
                         loc='bottom')
    the_table.auto_set_font_size(False)
    the_table.set_fontsize(10)
    # Adjust layout to make room for the table:
    # plt.subplots_adjust(left=0.2, bottom=0.2)
    plt.ylabel("Number of Casulities")
    plt.xticks([])
    plt.title('Casulities by Region')
    plt.show()
```

Objective 3:

In the below command we are fetching the sum and count of Severities corresponding to the first day of every month in a year

In [36]:

```
df_severity = uk_filter_data.groupby(["First_day_of_month","Accident_Severity"])["Numbe
```

In the below command we are manupulating the data 1 - group by to get only the required data points 2 - pivoting the group by data to get in desired format that will be usable in the stackplot

In [37]:

```
#fetching only 3 columns
df_severity = df_severity[["First_day_of_month","Accident_Severity","sum"]]
#setting the index from default to first day of month
df_severity = df_severity.set_index("First_day_of_month")
#made the columns as severity
df_severity = df_severity.pivot(columns='Accident_Severity')
#reseting index
df_severity = df_severity["sum"].reset_index()
#renaming the columns
df_severity = df_severity.rename(columns = {1:"y1",2:"y2",3:"y3"})
```

In the below command we are creating the plot we are simply using the stackplot function provided by the matplotlib and passing the data points that we prepared above

In [38]:

```
def scvsYear():
    fig,ax1 = plt.subplots(figsize=(10,6))

    ax1.set_title('Severity & Causality VS yearmonth')
    ax1.stackplot(
        'First_day_of_month',
        'y3',
        'y2',
        'y1',
        data=df_severity,
        colors=['pink', 'purple', 'red'],
        alpha=0.5
    )
    #adding legends and label for 1st graph
    labels = ["Slight", "Serious","Fatal" ]
    ax1.legend(labels)
    ax1.set_xlabel("YearMonth")
    ax1.set_ylabel("Causality")
    ax1.set_ylim(0, 26000,5000)
    plt.xticks(rotation = 45);
```

In the command below we are grouping the data on reason, accident severity to get only the required column. Then we are join with the top reasons dataframe created in objective 2 to only get the data for those reasons

In [39]:

```
df_uk_reason_sev = uk_filter_data.groupby(["reason","Accident_Severity"])["Number_of_Ca
df_uk_reason_sev = pd.merge(df_uk_reason_sev,df_uk_top10_reason["reason"],on="reason",h
```

In the command below we are pivoting the data to suffice our need to create a horizontal bar chart.from the below command, we will have 4 column reason, and remaining will be the sum of casualties by each severity in each column

In [40]:

```
df_uk_reason_sev = df_uk_reason_sev.set_index("reason").pivot(columns = ["Accident_Seve
```

Here we are converting the data in a dictionary and then passing that data into the visual so that we can run a loop over it and plot the stack horizontal bar chart and give the differnet colour for each severity

In [41]:

```
category_names = ["Slight","Serious","Fatal",]
results = {
    'R1': df_uk_reason_sev["Number_of_Casualties"].iloc[0].to_list()[::-1],
    'R2': df_uk_reason_sev["Number_of_Casualties"].iloc[1].to_list()[::-1],
    'R3': df_uk_reason_sev["Number_of_Casualties"].iloc[2].to_list()[::-1],
    'R4': df_uk_reason_sev["Number_of_Casualties"].iloc[3].to_list()[::-1],
    'R5': df_uk_reason_sev["Number_of_Casualties"].iloc[4].to_list()[::-1],
    'R6': df_uk_reason_sev["Number_of_Casualties"].iloc[5].to_list()[::-1],
    'R7': df_uk_reason_sev["Number_of_Casualties"].iloc[6].to_list()[::-1],
    'R8': df_uk_reason_sev["Number_of_Casualties"].iloc[7].to_list()[::-1],
    'R9': df_uk_reason_sev["Number_of_Casualties"].iloc[8].to_list()[::-1],
    'R10': df_uk_reason_sev["Number_of_Casualties"].iloc[9].to_list()[::-1]
}
def severity():
    labels = list(results.keys())
    data = np.array(list(results.values()))
    data_cum = data.cumsum(axis=1)
    category_colors = plt.colormaps['tab20b'](
        np.linspace(0.15, 0.85, data.shape[1]))

    fig, ax = plt.subplots(figsize=(12, 5))
    ax.invert_yaxis()
    ax.xaxis.set_visible(False)
    ax.set_xlim(0, np.sum(data, axis=1).max())

    for i, (colname, color) in enumerate(zip(category_names, category_colors)):
        widths = data[:, i]
        starts = data_cum[:, i] - widths
        rects = ax.barh(labels, widths, left=starts, height=0.5,
                        label=colname, color=color)

        r, g, b, _ = color
        text_color = 'white' if r * g * b < 0.5 else 'darkgrey'
    ax.legend(ncol=len(category_names), bbox_to_anchor=(0, 1),
              loc='lower left', fontsize='small')
    plt.show()
```

Objective 4:

Taking the sum of casualties for 2011 & 2012 by grouping the year and conditions: road, light and weather

In [42]:

```
df_combined = uk_acc_all_data[uk_acc_all_data["Year"].isin([2011, 2012])]
uk_bar_graph_data_natural = df_combined.groupby(
    ["Year", "Road_Surface_Conditions", "Light_Conditions", "Weather_Conditions"])["Nur
uk_bar_graph_data_natural = uk_bar_graph_data_natural.reset_index()
```

Creating two dataframes for 2011 & 2012 which is sorted based on the no of casualties, top 15 records is taken. New column is created in both the dataframes using the lambda function by concating the columns (Road_Surface_Conditions, Light_Conditions, Weather_Conditions) for all the rows

In [43]:

```
df_line_plot_2011 = uk_bar_graph_data_natural[uk_bar_graph_data_natural["Year"] == 2011
df_line_plot_2011['Reasons'] = df_line_plot_2011.apply(lambda row: (row['Road_Surface_C
                                row['Light_Conditions'] + '+' + row['Weather_Conditions
df_line_plot_2012 = uk_bar_graph_data_natural[(uk_bar_graph_data_natural["Year"] == 201
df_line_plot_2012['Reasons'] = df_line_plot_2012.apply(lambda row: (row['Road_Surface_C
                                 row['Light_Conditions'] + '+' + row['Weather_Condition
```

Merging both the years dataframes based on Reasons column, the top 10 records are taken. Storing the no of casualties as a list for 2011 & 2012 in two variables. Reasons is stored in a variable as list

In [44]:

```
merged_plot_df = pd.merge(df_line_plot_2011, df_line_plot_2012, on = 'Reasons', suffixe
merged_plot_df = merged_plot_df.sort_values("Reasons", ascending = False)

no_of_casualties_2011_list = merged_plot_df['Number_of_Casualties_2011'].tolist()
no_of_casualties_2012_list = merged_plot_df['Number_of_Casualties_2012'].tolist()
reasons_list_line = merged_plot_df['Reasons'].tolist()
```

Plotting the line chart based on the Reasons column for 2011 & 2012. Used matplotlib to specify the figure size and plot the x-y axis with the labels. x-axis has the top 10 reasons(R1-R10) and y-axis has the no of casualties

In [45]:

```
def reason_casulities():
    plt.rcParams["figure.figsize"] = [12, 5]
    plt.rcParams["figure.autolayout"] = True

    r1 = mpatches.Patch(color = 'white', label = 'R1 - ' + reasons_list_line[0])
    r2 = mpatches.Patch(color = 'white', label = 'R2 - ' + reasons_list_line[1])
    r3 = mpatches.Patch(color = 'white', label = 'R3 - ' + reasons_list_line[2])
    r4 = mpatches.Patch(color = 'white', label = 'R4 - ' + reasons_list_line[3])
    r5 = mpatches.Patch(color = 'white', label = 'R5 - ' + reasons_list_line[4])
    r6 = mpatches.Patch(color = 'white', label = 'R6 - ' + reasons_list_line[5])
    r7 = mpatches.Patch(color = 'white', label = 'R7 - ' + reasons_list_line[6])
    r8 = mpatches.Patch(color = 'white', label = 'R8 - ' + reasons_list_line[7])
    r9 = mpatches.Patch(color = 'white', label = 'R9 - ' + reasons_list_line[8])
    r10 = mpatches.Patch(color = 'white', label = 'R10 - ' + reasons_list_line[9])

    plt.legend(handles=[r1, r2, r3, r4, r5, r6, r7, r8, r9, r10])

    axes = plt.gca()

    merged_plot_df.plot(kind = 'line', x = 'Reasons', y = 'Number_of_Casualties_2011',
    merged_plot_df.plot(kind = 'line', x = 'Reasons', y = 'Number_of_Casualties_2012',

    axes.set_xticks([0,1,2,3,4,5,6,7,8,9])
    axes.set_ylabel('No of Casualites')
    axes.set_title('Casualties based on the Road, Light and Weather Conditions')
    axes.set_xticklabels(('R1', 'R2', 'R3', 'R4', 'R5', 'R6', 'R7', 'R8', 'R9', 'R10'))

    plt.show()
```

Taking the sum of casualties for 2011 & 2012 by grouping the column Did_Police_Officer_Attend_Scene_of_Accident(has two values Yes or No) and conditions: road, light and weather. New column is created in both the dataframes using the lambda function by concating the columns (Road_Surface_Conditions, Light_Conditions, Weather_Conditions) for all the rows.

In [46]:

```
df_bar_plot = df_combined.groupby(
    ["Did_Police_Officer_Attend_Scene_of_Accident", "Road_Surface_Conditions", "Light_C
df_bar_plot = df_bar_plot.reset_index()
df_bar_plot = df_bar_plot.sort_values(['Number_of_Casualties'], ascending = [False])
df_bar_plot['Reasons'] = df_bar_plot.apply(lambda row: (row['Road_Surface_Conditions']
                                row['Light_Conditions'] + '+' + row['Weather_Condition
```

Creating two dataframes depending if the police officer attened the scene of accident, sorted based on the no of casualties, top 20 records is taken. Merging both the years dataframes based on Reasons column, the top 10 records are taken.

In [47]:

```
df_bar_plot_police_attended_yes = df_bar_plot[(df_bar_plot["Did_Police_Officer_Attend_S
df_bar_plot_police_attended_no = df_bar_plot[(df_bar_plot["Did_Police_Officer_Attend_Sc
merged_bar_plot_df = pd.merge(df_bar_plot_police_attended_yes, df_bar_plot_police_atter
merged_bar_plot_df = merged_bar_plot_df.sort_values("Reasons", ascending = False)
```

Storing the no of casualties in two variables when the police attended the event or not. Reasons is stored in a variable as list.

In [48]:

```
no_of_casualties_police_attended_yes_list = merged_bar_plot_df['Number_of_Casualties_Ye
no_of_casualties_police_attended_no_list = merged_bar_plot_df['Number_of_Casualties_No'
reasons_list_bar = merged_bar_plot_df['Reasons'].tolist()
```

Plotting the bar chart based on the whether the police atteneded the scene or not. Used matplotlib to specify the figure size and plot the x-y axis with the labels. x-axis has the top 10 reasons(R1-R10) and y-axis has the no of casualties

In [49]:

```
def Police_severity():
    plt.rc('figure', figsize=(12, 5))

    x_axis = np.arange(len(merged_bar_plot_df))

    plt.bar(x_axis, no_of_casualties_police_attended_yes_list, width = 0.2, color = 'bl
    plt.bar(x_axis +0.20, no_of_casualties_police_attended_no_list, width = 0.2, color

    plt.xticks(x_axis, ('R1', 'R2', 'R3', 'R4', 'R5', 'R6', 'R7', 'R8', 'R9', 'R10'))

    y1 = mpatches.Patch(color = 'blue', label = 'Police attended the scene of accident'
    y2 = mpatches.Patch(color = 'red', label = 'Police did not attend the scene of acci
    r1 = mpatches.Patch(color = 'white', label = 'R1 - ' + reasons_list_bar[0])
    r2 = mpatches.Patch(color = 'white', label = 'R2 - ' + reasons_list_bar[1])
    r3 = mpatches.Patch(color = 'white', label = 'R3 - ' + reasons_list_bar[2])
    r4 = mpatches.Patch(color = 'white', label = 'R4 - ' + reasons_list_bar[3])
    r5 = mpatches.Patch(color = 'white', label = 'R5 - ' + reasons_list_bar[4])
    r6 = mpatches.Patch(color = 'white', label = 'R6 - ' + reasons_list_bar[5])
    r7 = mpatches.Patch(color = 'white', label = 'R7 - ' + reasons_list_bar[6])
    r8 = mpatches.Patch(color = 'white', label = 'R8 - ' + reasons_list_bar[7])
    r9 = mpatches.Patch(color = 'white', label = 'R9 - ' + reasons_list_bar[8])
    r10 = mpatches.Patch(color = 'white', label = 'R10 - ' + reasons_list_bar[9])

    plt.legend(handles=[y1, y2, r1, r2, r3, r4, r5, r6, r7, r8, r9, r10])

    axes = plt.gca()

    axes.set_xticks([0,1,2,3,4,5,6,7,8,9])
    axes.set_xlabel('Accident Reasons')
    axes.set_ylabel('No of Casualites')
    axes.set_title('Casualties based on the Road, Lightning and Weather Conditions')

    plt.show()
```

Project Outcome (10 + 10 marks)

This section should describe the outcome of the project by means of both explanation of the results and by graphical visualisation in the form of graphs, charts or or other kinds of diagram

The section should begin with a general overview of the results and then have a section for each of the project objectives. For each of these objectives an explanation of more specific results relating to that objective shoud be given, followed by a section presenting some visualisation of the results obtained. (In the case where the project had just one objective, you should still have a section describing the results from a general perspective followed by a section that focuses on the particular objective.)

The marks for this section will be divided into 10 marks for Explanation and 10 marks for Visualisation. These marks will be awarded for the Project Outcome section as a whole, not for each objective individually. Hence, you do not have to pay equal attention to each. However, you are expected to have a some explanation and visualisation for each. It is suggested you have 200-400 words explanation for each objective.

Overview of Results

Give a general overview of the results (around 200 words).

Objective 1

Explanation of Results

Causality trends over months and their top 5 reasons

Here we are trying to identify how the causality is changing over the months and how the factors like -

- Light conditions - *Day or Night*,
- Road conditions - *Wet or Dry*
- Weather Conditions - *Winds or no Winds*

impact the accident and the number of casualties.

The data suggests that there is a high relationship between the time of the day and road conditions. Along with this, we can see there is an increasing trend in a particular **time period of the year** where the accidents increase relatively and decrease after that particular period.

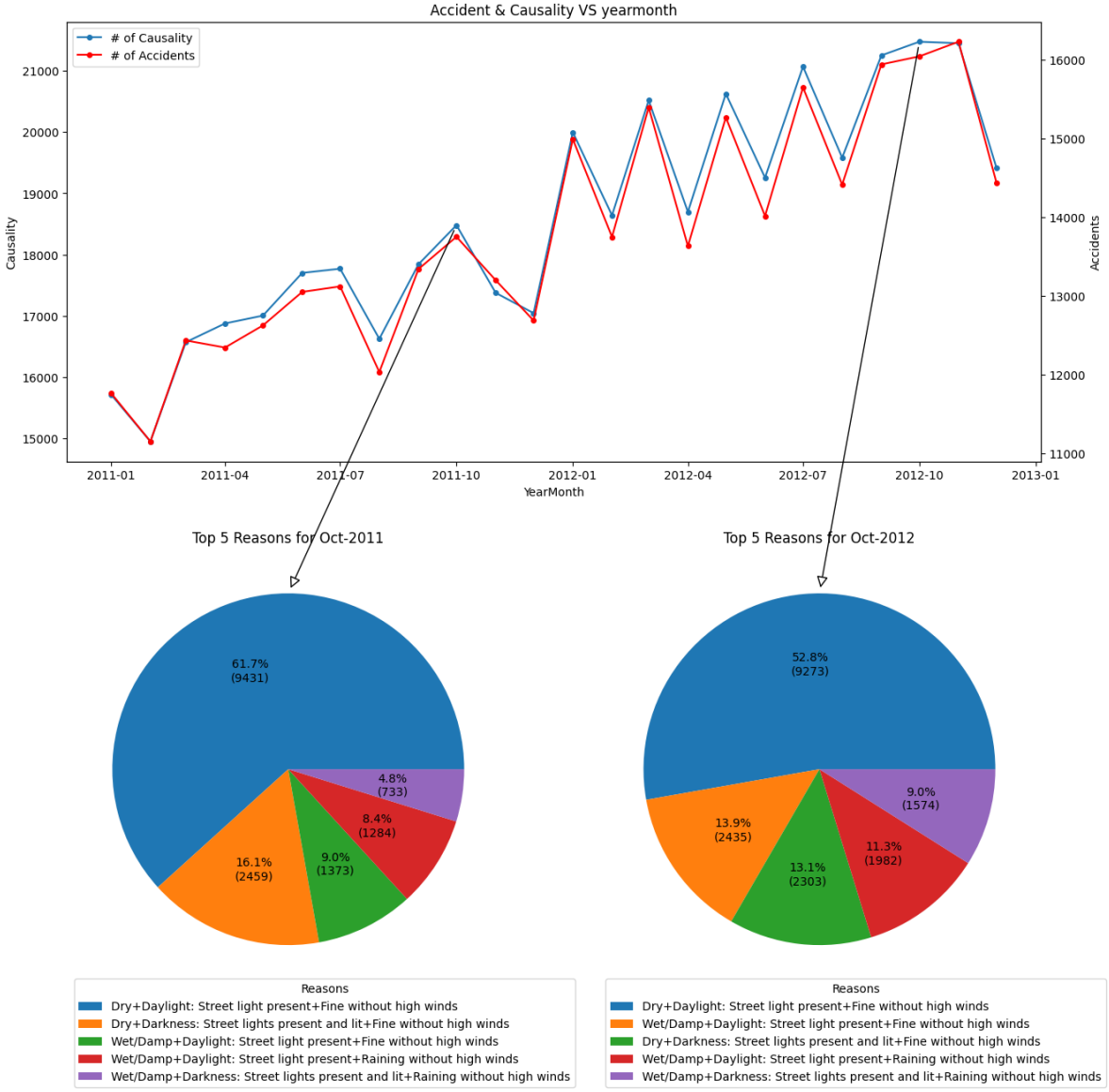
1. There are `frequent peaks and bottoms` in the values of *casualties and accidents* in our data i.e. there is a trough and crest after every month with an `increase` in number of both casualties and accidents.
2. There is a *strong* relationship between the year months and the number of accidents, we can see from the below graph, *title = "Accident & Causality VS year month"* that on the 10th month of each year the number of accidents is higher when in comparison to the rest of the months. For e.g. highest number of casualties in 2011, are reported in Oct(first arrow) i.e. 18479
3. We can see that during the first quarter of 2011, the line of casualties and accidents is *identical* which implies that for every accident there were quite less casualties as compared to others (the first 3 dots where the red and blue are overlapping). After that, the number of casualties started increasing as compared to the number of accidents and then again by the end of 2011 both started to overlap till the 1st month of 2012 and then again, the *same pattern* can be observed in 2012.
4. One more key observation is that during the troughs, casualties are much higher than the accidents as compared to the numbers during crest. This appeared an `interesting discovery` on which further analysis could be performed which after investigation came out that the top 5 reasons for accidents where there is a peak for each year is due to "dry road conditions where there is daylight and no high winds"

Visualisation

For the 4th point the same can be observed in the 2 pie charts displayed below with the title " top 5 reasons for 2011", "top 5 reasons for 2012"

In [50]:

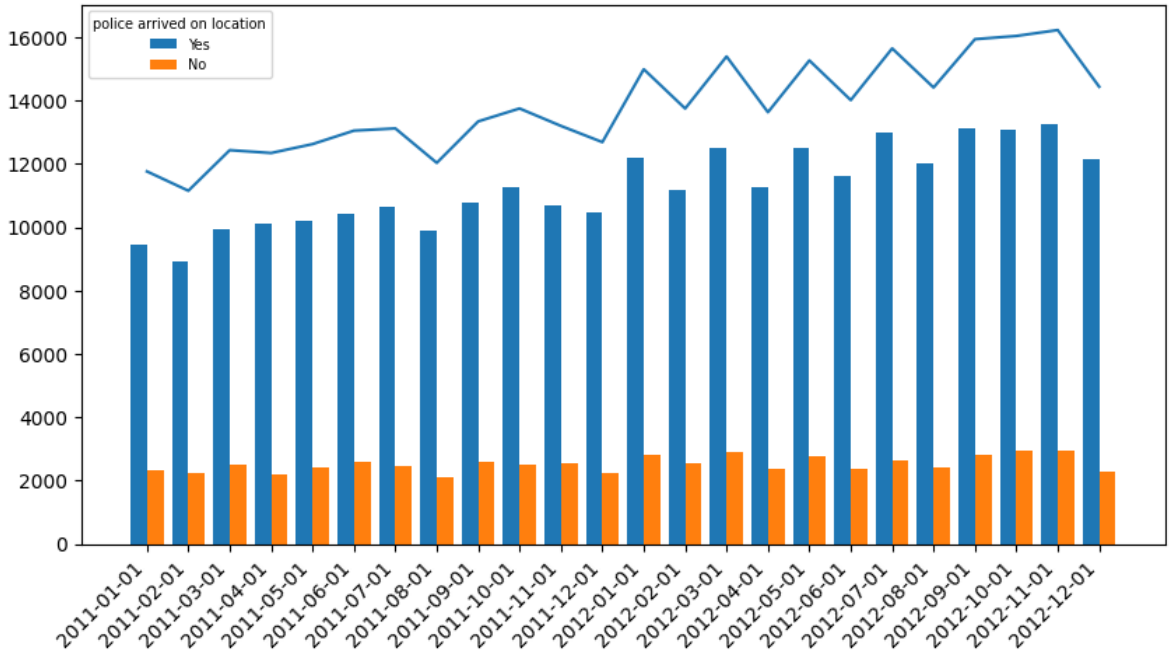
```
1 plot_obj1()
```



The below graph tells us that "police response" are respectivley good when we are checking with the months

In [51]:

```
1 plot_police()
```



Objective 2

Explanation of Results

Analysis on Reason by region

We have identified the top 10 best and the worst regions to live in terms of causalties.

- Best region - ***Orkney Island***
- Worst region - ***Birmingham***

We have identified which reason has the highest impact on the worst places to live in terms of **# of casualty**, as could be seen from the **Worst Places to live in** heat map in the command below. For all the places **R4** `(Dry+Daylight: Street light present+Fine without high winds)` reason comes out to be dominant. In the same manner, we have identified the reason of accidents at the best places in the **best places to live in** heat map and there **R3** `(Dry+Daylight: Street light present+Fine without high winds)` comes out to be dominating.

Note - Here you might be wondering why the different names R3 and R4 have the same reason in both scenarios. The reason behind that is, that for each scenario we are doing analysis, first identifying the top 10 reasons for that cluster. for e.g. we are identifying the top 10 reasons for the worst cities and those top 10 might be different from the top 10 reasons for the best cities hence the different abbreviations for each scenario.

Analysis of Severity by region

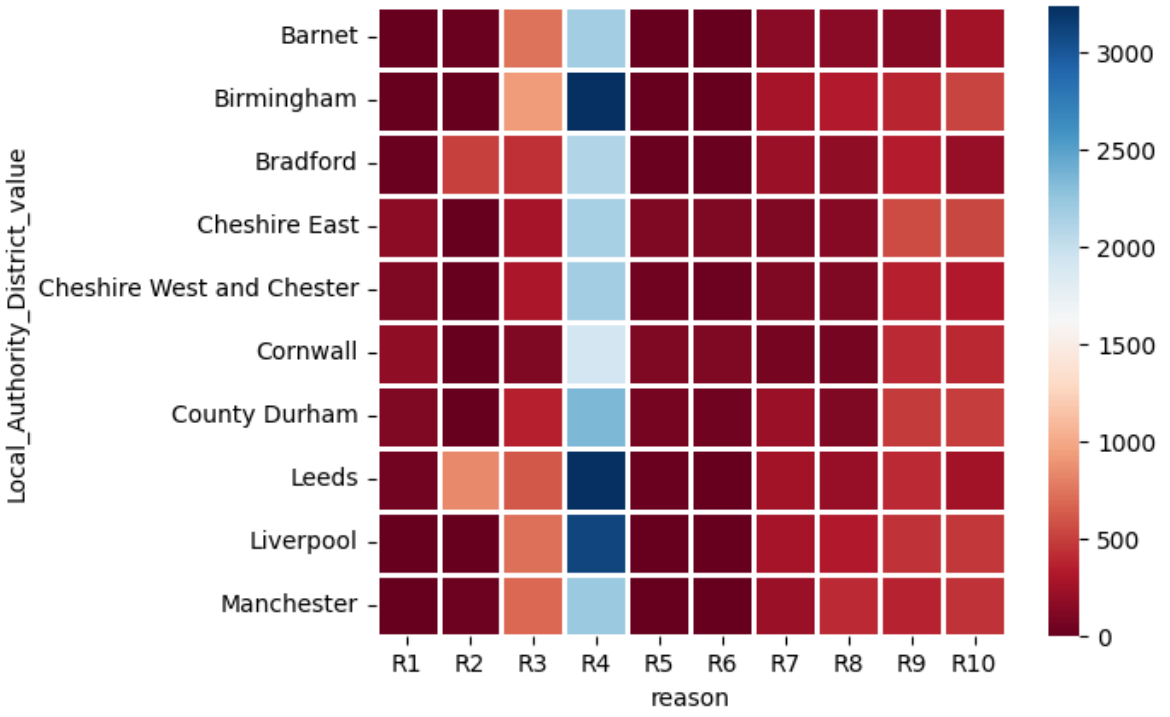
We have identified the cities which have least and maximum number of casualties as per the severities during accidents in both best and worst places to live in.
If a person wants to live in the Worst region cluster and still wants to be safe then we can identify that as well. From the **causality by region and severity** graph we can see **Barnet** has the least number of fatal accidents but **County Durham** has the highest number of fatal accidents in that region. In the same way, if a person wants to be extra safe and wants to identify the safest place from the best region to live in the cluster that would be **Western isles** it has 0 fatal accidents and highest number of **fatal** accidents in this are in ***Rutland***

Visualisation

Worst places to live in

In [52]:

```
1 heatmap()
```

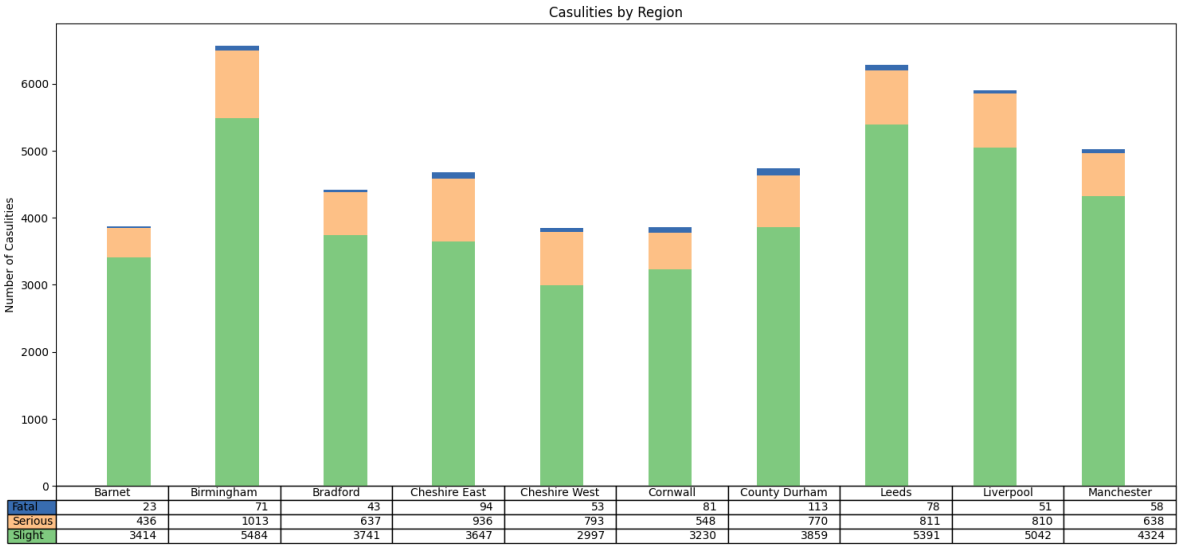


Fullform

R1	Dry+Darkeness: No street lighting+Fine without high winds
R2	Dry+Darkness: Street lighting unknown+Fine without high winds
R3	Dry+Darkness: Street lights present and lit+Fine without high winds
R4	Dry+Daylight: Street light present+Fine without high winds
R5	Wet/Damp+Darkeness: No street lighting+Fine without high winds
R6	Wet/Damp+Darkeness: No street lighting+Raining without high winds
R7	Wet/Damp+Darkness: Street lights present and lit+Fine without high winds
R8	Wet/Damp+Darkness: Street lights present and lit+Raining without high win
R9	Wet/Damp+Daylight: Street light present+Fine without high winds
R10	Wet/Damp+Daylight: Street light present+Raining without high winds

In [78]:

```
1 CbyR()
```

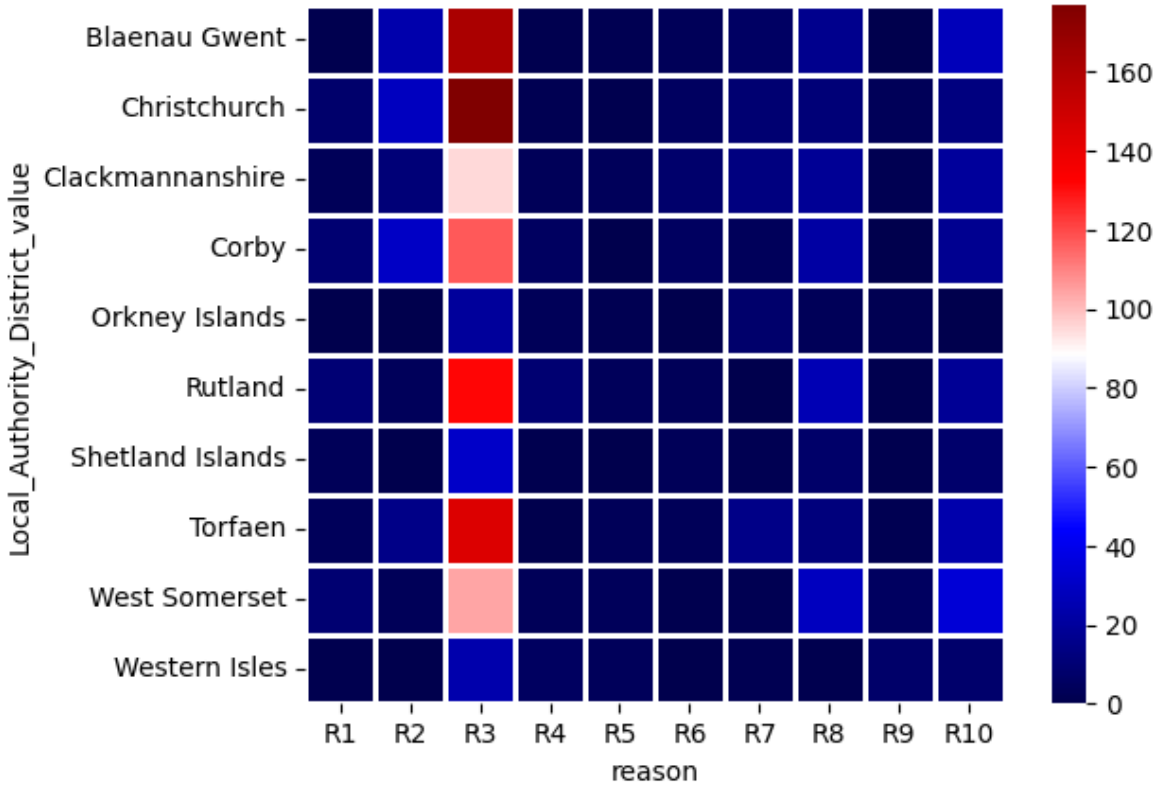


	Barnet	Birmingham	Bradford	Cheshire East	Cheshire West	Cornwall	County Durham	Leeds	Liverpool	Manchester
Fatal	23	71	43	94	53	81	113	78	51	58
Serious	436	1013	637	936	793	548	770	811	810	638
Slight	3414	5484	3741	3647	2997	3230	3859	5391	5042	4324

Best places to live in

In [54]:

```
1 heatmap2()
```

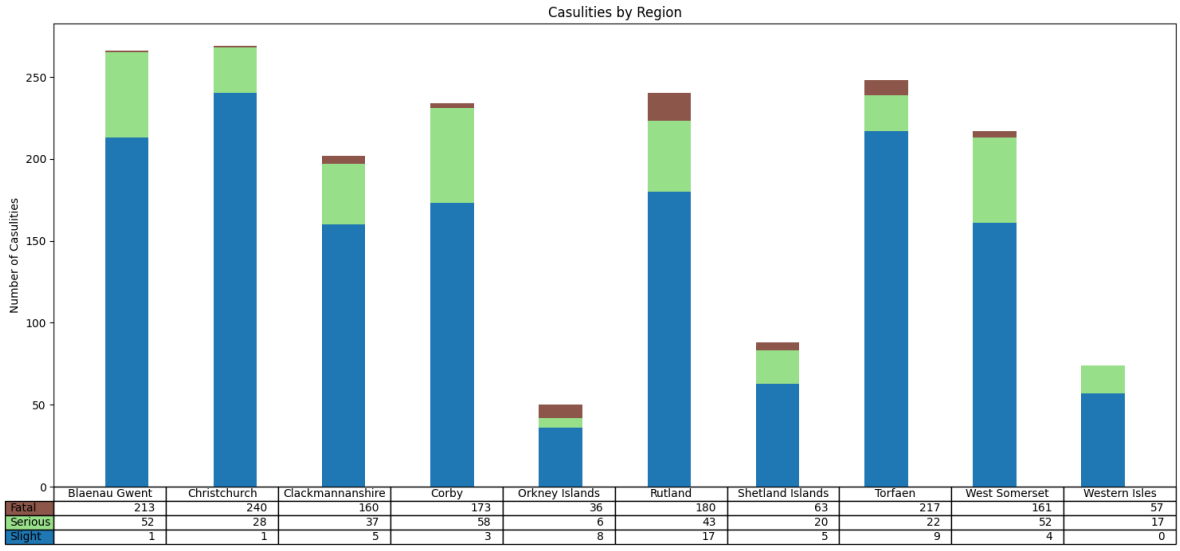


Fullform

R1	Dry+Darkeness: No street lighting+Fine without high winds
R2	Dry+Darkness: Street lights present and lit+Fine without high winds
R3	Dry+Daylight: Street light present+Fine without high winds
R4	Wet/Damp+Darkeness: No street lighting+Fine without high winds
R5	Wet/Damp+Darkeness: No street lighting+Raining without high winds
R6	Wet/Damp+Darkness: Street lights present and lit+Fine without high winds
R7	Wet/Damp+Darkness: Street lights present and lit+Raining without high wi
R8	Wet/Damp+Daylight: Street light present+Fine without high winds
R9	Wet/Damp+Daylight: Street light present+Raining without high winds
R10	Wet/Damp+Daylight: Street light present+Raining with high winds

In [71]:

```
1 CbyR_best()
```



	Blaenau Gwent	Christchurch	Clackmannanshire	Corby	Orkney Islands	Rutland	Shetland Islands	Torfaen	West Somerset	Western Isles
fatal	213	240	160	173	36	180	63	217	161	57
Serious	52	28	37	58	6	43	20	22	52	17
Slight	1	1	5	3	8	17	5	9	4	0

Objective 3

Here we are identifying the month from the 2 years which has the highest number of ***fatal***,***serious*** and ***slight*** accidents. From the analysis we can conclude through the below table:

Highest number of accidents	Month
Fatal	Dec, 2012
Serious	May, 2012
Slight	Oct, 2012

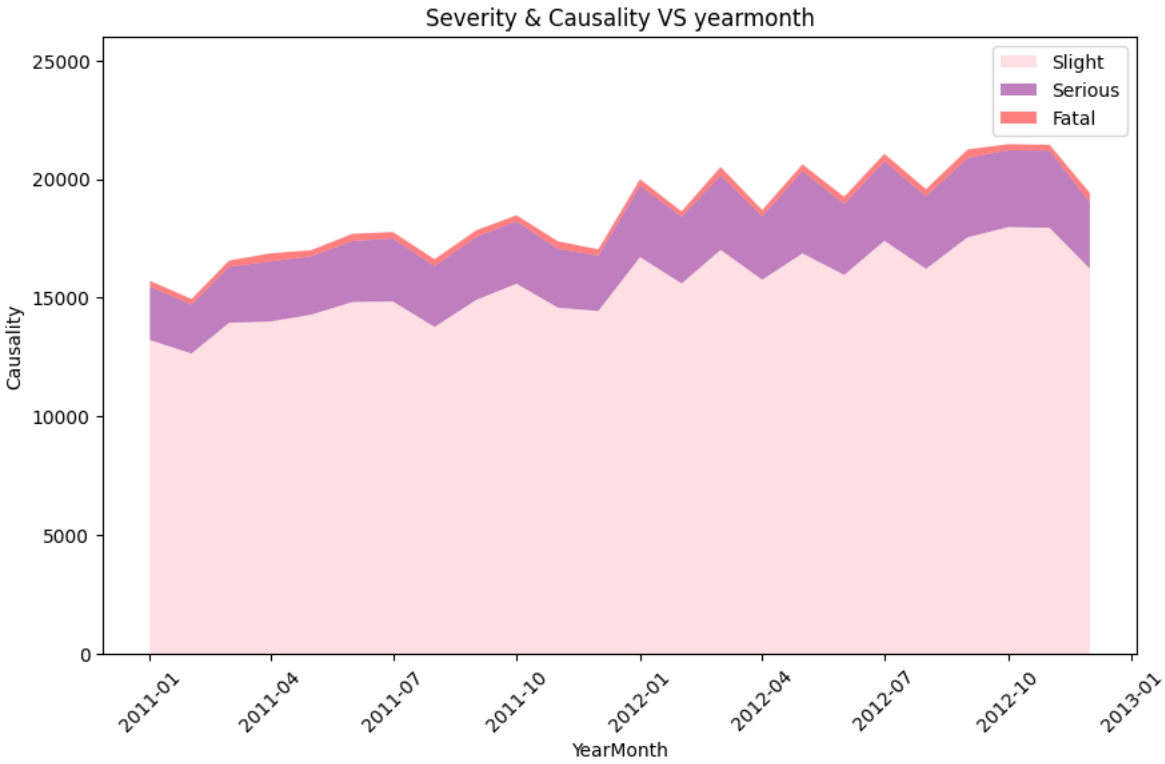
Secondly, we are identifying the top 10 reasons which have the highest number of casualties by severity, then we can see that **R4** `(Dry+Daylight: Street light present+Fine without high winds)` sores through the graph and has the maximum number of accidents across all the severities i.e. ***fatal***,***serious***,***slight***

Visualisation

In [56]:

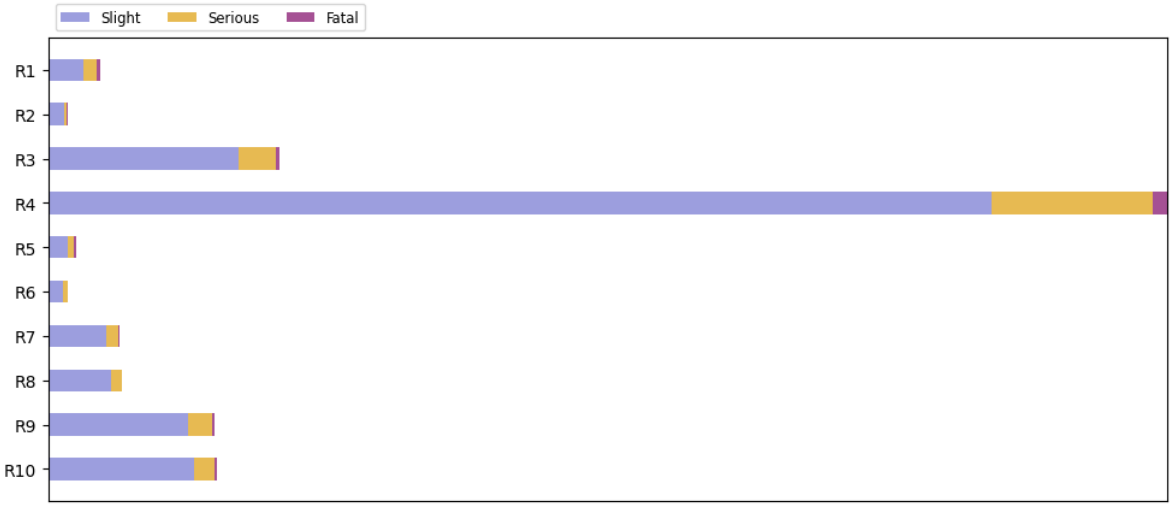
```
1 scvsYear()
```

```
C:\Users\kriti\AppData\Local\Temp\ipykernel_23672\569235331.py:20: MatplotlibDeprecationWarning: Passing the emit parameter of set_ylim() positionally is deprecated since Matplotlib 3.6; the parameter will become keyword-only two minor releases later.
  ax1.set_ylim(0, 26000,5000)
```



In [57]:

```
1 severity()
```



Objective 4

Explanation of Results

The number of casualties for the reasons R1, R2 and R8 for 2011 and 2012 have some significant amount of difference. Whereas the reasons R5, R6, R7 and R10 have comparatively lesser differences among them in both years.

R8 contributed to have the **most number of casualties** and on the other hand, **R7** has the **least no of casualties**. There can be below mentioned 2 explanations for the pattern observed in the reasons where we have **R8** (`dry road conditions where there is daylight and no high winds` dominating all the reasons:

1. These accidents are happening because people are driving for long periods, and they are dehydrated and hence resulting in accidents.
2. The dry weather is causing mirages and causing accidents.

Unfortunately, we don't have the data to make a conclusion on the same.

Police perfomrace over reasons

It can be clearly understood that the police made sure to attend the scene of the accidents no matter what the reason was. The number of times police did not attend was very less compared to the number of times they did attend the accident scene.

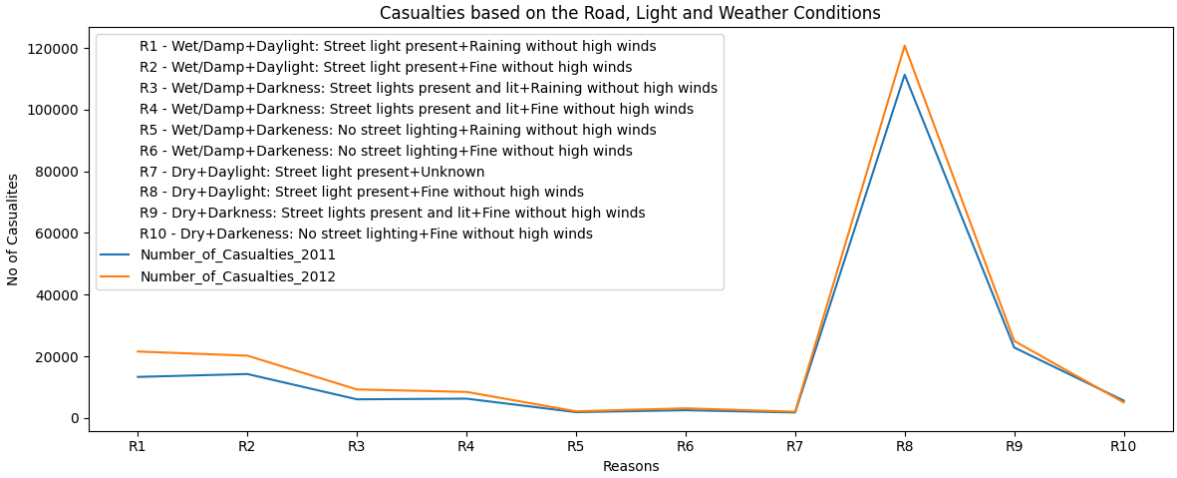
Dry road conditions where there is daylight and no high winds contributes to be the reason due to which the most number of casualties were there for both the cases where the police attend and do not attend the scene of the accident and on the other hand **R2** (`Wet road conditions where there is Daylight and Raining with high winds`) has the least no of casualties probably they are enjoying the weather from a safe distance.

Visualisation

Interpretation from the line graph: The Reasons(R1-R10) in the labels above are a combination of Road conditon + Lightning condition + Weather condition

In [58]:

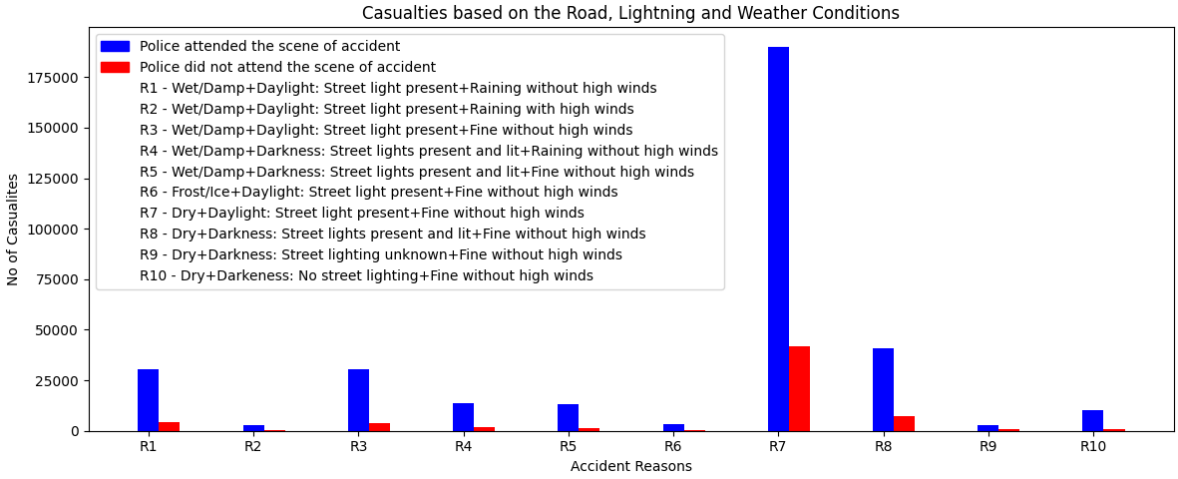
```
1 reason_casulities()
```



Interpretation from the bar graph: The Reasons(R1-R10) in the labels above are a combination of Road conditon + Lightning condition + Weather condition

In [59]:

```
1 Police_severity()
```



Conclusion (5 marks)

Achievements

Unfortunately, the highest number of accidents were not during the conditions such as raining, snowing or at night, Instead, they have happened **during day-light, when the roads were clean** and there were no external forces that would act as catalyst. The **highest** number of accidents have taken place in the autumn month of both the **years(October)** which is considered to be one of the best months to visit UK. The **total number of accidents** for both years are **0.44 million**. Happily, the **fatal accidents are very low** i.e. in k's (thousands), to be precise 6.8k. Serious accidents are relatively high but still low in thousands i.e. 66K and when moving to **Slight accidents** they are approaching to **half a million** i.e. 0.37 million. One of the great things about UK is its Police Force Response, it doesn't matter what the condition is or the time of the year is, **Police response has**

always been good. Lastly we have seen that the highest number of accidents have taken place in **Birmingham followed by Leeds, Liverpool, and Manchester** which pushes us to investigate more on the reason behind these particular cities and any other external condition that can be an important factor.

Limitations

Due to the limited set of factors such as surroundings, human control or physical facilities at the site of an accident but there is no information present if the **person was drunk driving or how was their mood or how is their physical and mental health or was they even eligible to ride the vehicle if they were riding one during accident**. We don't have the details of the casualty such as their age and sex itself and hence the precise reason for the high number of accidents in big cities such as Birmingham, Manchester, Leeds and Liverpool can't be commented on.

Future Work

In Future work we would like to obtain more data related to the casaulties as mentioned in the limitations as with that data we would be able to comment on the reason even more precisely and we hypothesise that there would be a **relationship between the drinking, licenced person with the accidents.**