A Comprehensive Investigation of Traffic Accidents in UK through Statistical methods, ML algorithms, and Deep learning

Student ID: 77302132

Name: Md Asif Mahmod Tusher Siddique

Abstract

Traffic casualties are a significant concern for modern civilization since it is amongst the leading key factors responsible for congestion, demise, emissions, and financial damage. When confronted with tragic and unanticipated road accidents, understanding what occurred previously and uncovering aspects which link with them, plus issuing early admonition, might contribute significantly to road network control as well as accident mitigation. So, in this study we have conducted a thorough research of the road traffic casualties that took place in Great Britain from 2016 to 2020. Four levels of disparate statistical and machine learning analysis is done in this study. T-test, one way ANOVA, and Chi square test of independence is done to uncover and experiment features that are strongly correlated to the accidents. Among all the machine learning models, accuracy score of decision tree with a few hyperparameter tuning was the superlative. LSTM was found to be the most accurate among all the deep learning algorithms and overall, the best algorithm tested to predict road accidents in this experiment.

Keywords: *Road Traffic Accidents (RTA), Descriptive statistics, Inferential statistics, Deep learning, LSTM, GRU, T-test, one way ANOVA, Chi square test of independence, Pearson's correlation.*

Table of Contents

Abstract 2

List of Figures 5

List of Abbreviations 7

1. Introduction 8

1.1 Aims and Objectives 9

1.2 Scoping of the research 12

1.3 Structure of the report 12

2. Literature Review 13

Risk variables of road accidents 13

Behavioural interpretation of accidents 13

Surrounding conditions of road accidents 14

Techniques used to analyse and predict road accidents 14

3. Methodology 16

3.1 Macro level Methodology 16

3.1.1 Data collection 17

3.1.2 Data Understanding 18

3.1.3 Data description 20

3.1.4 Data preparation 26

3.1.5 Data Quality 30

3.3 Micro Level Methodology 31

4. Results and Discussion 32

4.1 Descriptive Statistical Analysis 32

4.2 Inferential Statistical Analysis 46

4.2.1 Correlation 47

4.2.2 T-Test 52

4.2.3 One-way ANOVA 54

4.2.4 Chi Square test of independence 56

4.3 Machine Learning 60
4.3.1 Decision tree 61
4.3.2 Random Forest 62
4.3.3 Logistic Regression 62
4.3.4 KNN 64
4.4 Deep Learning 65
4.4.1 Neural Net 65
4.4.2 Neural Net with different optimizers 66
4.4.3 LSTM 67
4.4.4 GRU 68
5. Conclusion and Recommendations 70
References 71
Appendix 74

List of Figures

Figure 1: Deaths in road accidents in United Kingdom from 2004 to 2019 (Department of transport, UK, 2020) 8
Figure 2: Data Processing stages 17
Figure 3: First few rows of the Road Accident dataset (Part 1) 18
Figure 4: First few rows of the Road Accident dataset (Part 2) 18
Figure 5: First few rows of the Road Accident dataset (Part 3) 18
Figure 6: Last few rows of the Road Accident dataset (Part 1) 18
Figure 7: Last few rows of the Road Accident dataset (Part 2) 18
Figure 8: Last few rows of the Road Accident dataset (Part 3) 19
Figure 9: First few rows of the Casualty Statistics dataset (part 1) 19
Figure 10: First few rows of the Casualty Statistics dataset (part 2) 19
Figure 11: First few rows of the Casualty Statistics dataset (part 3) 19
Figure 12: Last few rows of the Casualty Statistics dataset (Part 1) 19
Figure 13: Last few rows of the Casualty Statistics dataset (Part 2) 20
Figure 14: Last few rows of the Casualty Statistics dataset (Part 3) 20
Figure 15: Tally of Null values in the Road Accident dataset 21
Figure 16: Description of each column of the Road accident dataset 24
Figure 17: Number of Null values in the Casualty statistics dataset 25
Figure 18: Description of each column of the casualty statistics dataset 26
Figure 19: First few columns of merged dataset (part 1) 27
Figure 20: First few columns of merged dataset (part 2) 28
Figure 21: Columns after removing duplicates 29
Figure 22: Null values in the dataframe 29
Figure 23: Column names and shape after data cleaning 30
Figure 24: First few columns of final data 32
Figure 25: Statistical description of the dataset 33
Figure 26: Features histogram 34
Figure 27: Road accidents over last 5 years 35
Figure 28: Trend of accidents at each month 35
Figure 29: Accidents at each month of each week 36
Figure 30: Road casualties in each day of the week 36

Figure 31: Accidents to the time of the day 37
Figure 32: Number of accidents with time 37
Figure 33: Accidents in UK map 38
Figure 34: Accident severity with respect to the locations 38
Figure 35: Accidents with respect to area type 39
Figure 36: Area of accident to casualty type 39
Figure 37: Age to the number of accidents 40
Figure 38: Sex of the drivers in accidents 41
Figure 39: Accidents severity 41
Figure 40: Severity of accidents with respect to gender 42
Figure 41: Accidents severity to the road type 42
Figure 42: Accident rates to the light conditions 43
Figure 43: Number of accidents to the weather conditions 44
Figure 44: Accidents to the road surface conditions 44
Figure 45: Special conditions at site to the accidents 45
Figure 46: Carriageway hazards influence on the road accidents 46
Figure 47: Pearson's correlation among all the features 47
Figure 48: Correlation of all features in 2016 48
Figure 49: Correlation of all features in2017 49
Figure 50: Correlation map with other columns in 2018 (without mask) 50
Figure 51: Correlation map with other columns (without mask) 51
Figure 52: Correlation map with other features in 2020 (Without mask) 52
Figure 53: Architecture of the Deep Neural network 65
Figure 54: Loss with the epoch runs in Deep Neural Net 66
Figure 55: Architecture of the NN to test performance of optimizers 66
Figure 56: Performance comparison of optimizers 67
Figure 57: Model's loss with each epoch 68
Figure 58: Architecture of GRU algorithm 68
Figure 59: Accuracy with epoch of GRU 69
Figure 60: Models loss with each epoch for GRU 69

List of Abbreviations

Abbreviation	Full form
RTA	Road Traffic Accidents
SEMMA	Sample, Explore, Modify, Model, and Assess
DM	Data Mining
CRISP-DM	Cross-Industry Standard Process for Data Mining
KDD	Knowledge discovery in databases
KNN	K nearest neighbor
CNN	Convolutional Neural Network
LSTM	Long Short-term memory
GRU	Gated recurrent unit
RMSE	Root mean square error

1. Introduction

Since the introduction of the vehicle, traffic accidents have been a major social concern. Automotive engineers and researchers have attempted to develop and manufacture safer vehicles, yet traffic accidents are inescapable. Everyday a myriad of road accidents occurs all around the world, resulting in numerous deaths and property losses, as well as a slew of other issues. About 1.35 million individuals worldwide perish every annum as a consequence of road casualties, as reported by the World Health Organization (2018). Around the world almost 3.700 persons die each day in traffic casualties involving pedestrians or other modes of transportation such as cars, buses, motorbikes, bicycles, and trucks around the world. Pedestrians, motorcyclists, and cyclists constitute the majority of those fatalities (Jian et al., 2019). Road traffic accidents damage many countries almost three of their total Gross Domestic Product, and traffic accidents are the main reason behind the mortality among kids and youthful people aged 5 to 29 years (WHO, 2018). Traffic accidents continue to result in a significant number of casualties and hospitalizations, despite updated car safety measures. The primary causes of traffic accidents may be separated into three categories: accommodating human mistake, vehicle mechanical state, and transportation systems (Petridou et al., 2000).

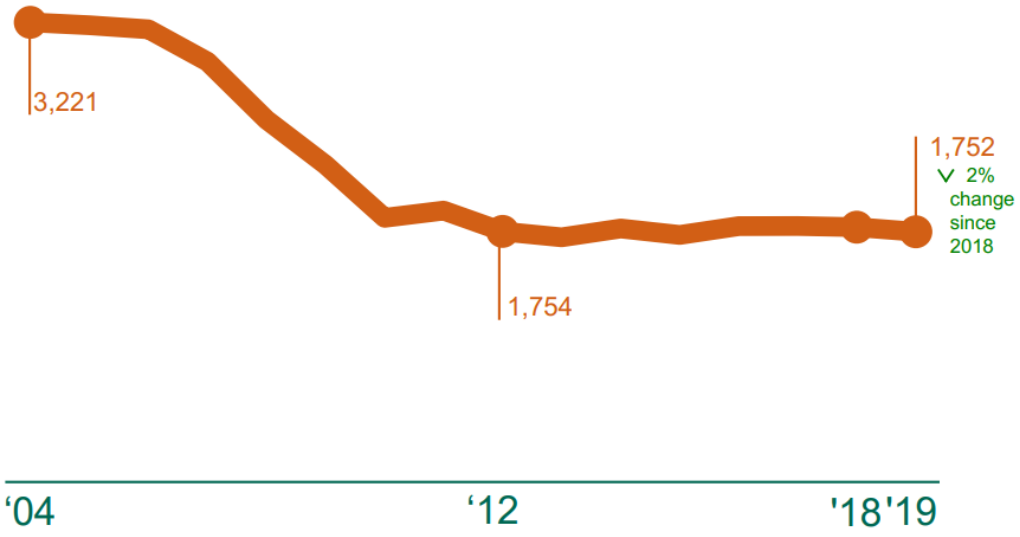


Figure 1: Deaths in road accidents in United Kingdom from 2004 to 2019 (Department of transport, UK, 2020)

The situation in UK is similarly dreadful as the rest of the world in terms of road accidents and road casualties. Between 2012 and 2019, the figure of traffic demise was approximately 1,850 every annum, or around 5 on average everyday (Brake, 2020), in Great Britain. Even though the COVID pandemic had a major influence on the road accidents as contrasted to the 3-year mean from 2017 to 2019. In April 2020, the monthly sharp drop in road fatalities was the highest, at 68 percent (National statistics, 2020). According to national statistics (2020), in 2020, there were an anticipated 1,460 traffic fatalities and 23,529 killed or seriously injured

(KSI) fatalities, with an aggregate of four months of nationwide lockdown. Finally, in the end of 2021, an approximate 1,390 traffic fatalities were recorded, down 11% from the previous annum. In comparison to the three-year mean from 2017 to 2019, which was before the pandemic (COVID-19), there was a 22 percent drop (National statistics, 2021).

This research presents critical overview and data analytics of the road accidents data of Great Britain (STATS19) which was collected from the UK govt website through the use of statistical tools, machine learning and deep learning techniques. We tried to identify all the zones, age groups, gender, road conditions and other key elements that might contribute to the traffic casualties by visualizing them. Then used statistical tools to find correlations and make inferences on them. Ultimately, we used cutting-edge time series, machine learning, and deep learning techniques to predict future casualties.

1.1 Aims and Objectives

This section illustrates the experiment questions and objectives we want to concentrate on this report. In this research the initial goal is to identify critical characteristics and patterns that influence accidents and its severity. The second goal is to create ML and deep learning models that can reliably forecast the accidents. To be more explicit, the goal is to create AI models that are expected to be able to forecast accidents without any comprehensive information about the accident, such as driver qualities or vehicle type. Overall, in this experiment four levels of analysis will be done on more than half a million data from 2016 to 2020 of road accidents in UK.

Broad Research Aims

- Finding proper comprehensive dataset on Road Accidents on UK to analyse and make inference on the road accidents.
- Do a critical literature review on the research and publications on the road accidents.

RO Level 1: Descriptive Statistical Analysis

Penalizing the factors leading to the road accidents in UK and visualizing through various diagrams and visualising appropriate data on the map.

RO 1.1. To find Mean, median, mode, standard deviation, min, and max of all the features of the dataset.

RO 1.2. Visualise all the columns through histogram to see the trend in the data.

RO 1.3. Visualise the trend of accidents over the years from 2016 to 2020.

RO 1.4. The frequency of accidents at each month of the year visualised through a time series diagram.

RO 1.5. Visualise the trend of accidents over the days of the week. Is there day connection among the casualty constancy and the days of the week.

RO 1.6. Through bar chart show the trend of accidents at disparate hours of the day to see when the most accidents happen.

RO 1.7. Using the latitude and longitude features show the casualties in the UK map to visually interpret the zones of casualties.

RO 1.8. Using the latitude and longitude features show the accidents severity map through scatter graph to find the zone of mostly fatal accidents.

RO 1.9. Show the casualty type of the accidents in the map.

RO 1.10. visualise the accidents trend to the age of the vehicle operator to interpret is there a relationship between the accidents and the age group.

RO 1.11. To find if there is any relationship among the sex of the driver and the accidents visualise it through bar graph.

RO 1.12. Through bar chart show what severity of the accidents is most prevalent and subsequently show the severity with sex of the driver.

RO 1.13. Visualise the severity of the accidents with the road types to find type of road has most accidents.

RO 1.14. The accidents trend with the weather, surface conditions, and light situations present at the moment of the accident.

RO 1.15. The accidents trend with the special conditions that is present on the site and carriageway hazards.

RO Level 2: Inferential Statistical Analysis

Inferences that can be made on the dataset will be analysed with various Inferential statistical tools such as one-way ANOVA, T-Test, test of equality of variance via Levene's test, Pearson Correlation test, Chi-Square test, and OLS Regression analysis.

RO 2.1 Through Pearson's correlation matrix find each year's correlation with the other features of the dataset.

RO 2.2 Use –T-test to investigate if the gender of the driver has any impact on the severity of the accidents.

RO 2.3. Use –T-test to investigate if the location has any impact on the accident's severity?

RO 2.4. Are young drivers more likely to be in dangerous casualties than older counterparts?

RO 2.5 Use T-test to investigate if bad lighting conditions contribute more to severe road accidents.

RO 2.6. Use One-Way ANOVA (Analysis of variance) to investigate if different days of the week's have disparate casualty type.

RO 2.7. Use One-Way ANOVA (Analysis of variance) to investigate if different periods have different severity of accidents.

RO 2.8. Use One-Way ANOVA (Analysis of variance) to investigate if different road class have various accident severity.

RO 2.9. Use One-Way ANOVA (Analysis of variance) to investigate if different junction type has different casualty type.

RO 2.10. Use One-Way ANOVA (Analysis of variance) to investigate if different road surface conditions have different severity of the accidents.

RO 2.11. Use Chi-square test of independence to investigate if there is any association among the severity of the accidents and the sex of the driver?

RO 2.12. Use Chi-square test of independence to investigate if there is any association among the severity of the accidents and the casualty home area type?

RO 2.13. Use Chi-square test of independence to investigate if there is any interrelation among the casualty's severity and the road type?

RO 2.14. Use Chi-square test of independence to investigate if there is any interrelation among the casualty's severity and the weather conditions?

RO 2.15. Use Chi-square test of independence to investigate if there is any interrelation among the casualty's severity and the light conditions?

RO 2.16. Use Chi-square test of independence to investigate if there is any association amongst the severity of the accidents with the speed limit?

RO Level 3: Data Analytics with Machine Learning

Various Machine learning algorithms to forecast the road casualty's contingent on time, area, weather, road type, speed limit, etc. Here, mainly we will focus on ML algorithms for example, Decision tree classifier, Logistic regression, and Random Forest classifier. At the end, the results will be analysed and compared with one another to find the best Machine learning algorithm to predict the road accident.

RO 3.1. Use decision tree algorithm to predict the severity of the accidents.

RO 3.2. Use Random Forest algorithm to predict the severity of the accidents.

RO 3.3 Use Logistic regression to predict the severity of the accidents

RO 3.4. Implement KNN (K nearest neighbour) while using correlated columns to predict the severity of the accidents.

RO Level 4: Advanced Data analysis with Deep learning

Here, Deep learning algorithms such as LSTM, CNN and other advanced data analysis will be implemented on the dataset. The prediction outcome will also be contrasted with other deep learning models to find which algorithm is performing best.

RO 4.1. Implement a simple Neural net with correlated features of the dataset to predict the accident's severity.

RO 4.2. Implement a Neural network with different optimizers and make comparison of the performance of different optimizers.

RO 4.3. Construct a LSTM (Long Short-term memory) network and use various highly correlated features of the dataset to forecast the accidents severity.

RO 4.R. Using GRU (Gated recurrent unit) neural network predict to accidents severity.

1.2 Scoping of the research

The scope of this study is confined to the last 5 years data of road accidents of United Kingdom. There are many aspects that are specific for the regions of UK, such as location maps and predictions of the accidents. Predictions of the people of UK also might not fully align with the people from other parts of the world, as they might have very different road and traffic regulations. The UK government's official website is the source of all the data that is used in this experiment. This information will be evaluated using descriptive statistical analysis, inferential statistical analysis, several machine learning techniques, and some deep learning algorithms. The main focus of this research is to uncover noteworthy aspects and patterns in the collected data in terms of location, time, age, gender, and an overall scenario of casualties from 20016 to 2020.

1.3 Structure of the report

This thesis report is mainly organized in five sections. The structure of report by each chapter is given below:

Introduction: In this chapter the important background of road accidents, the scope of the research, the aims and objectives and contribution of this work is given.

Literature review: a critical literature review of statistical analysis of road accidents as well as machine learning, dep learning, dig data approaches to predict road accidents.

Methodology: This chapter is divided into the micro and macro level methodology in which all the methodologies, data cleaning, data preparing, and processing is described.

Results and discussion: The result analysis of the data obtained from the experiments conducted and interpretations of those.

Conclusion and recommendations: In this chapter the outcome of the research is discussed along with the probable direction of the further research.

2. Literature Review

Study on disparate aspects of the Road traffic accidents and casualties can be found on the literature. Many studies concentrated on the specific aspects of the road accidents through statistical analysis and others used data mining, machine learning, and deep learning tools to examine road traffic accidents. For example, Time series analysis was utilized by Ljubič et al. (2002) to study the road accidents in UK, Hypothesis testing was employed by Sikdar et al. (2017) to analyse accidents that occurred in India.

Risk variables of road accidents

Haddon and William (1980) identified risk variables in connection to the person, vehicle, and road infrastructure before, during, and after the incident. One should pick certain precautions that would avoid the accident while in the pre-casualty period. The casualty state is connected with correctives that either circumvent or minimise the severity of an injury if it does transpire. At length, the post-casualty state encompasses all actions targeted at reducing the crash's detrimental repercussions after it has happened. The distinct characteristics connected with a crash make it possible to identify and analyse mitigation strategies. Steg et al. (2009) and Nowakowska (2012) investigated many risk variables for individual behaviour. Excessive speed, driving while drunk or beneath the effect of other psychotropic medications, driving without a bike helmet, not wearing a seatbelt, transporting children without child restraints, driver psychical mindset, and distractions while driving are all risk factors that are dependent on the driver. The driver's expertise, skills, observation talents, and capacity to spot potential risks enable him or her to escape potential traffic collisions.

Behavioural interpretation of accidents

Defects, lapses, and infringements are three categories of behaviours that enhance the likelihood of producing an accident (Reason et al., 1990). Perception mistakes and misjudgement of driving circumstances, such as forceful stopping on a slick road, high-speed turning, or an abrupt lane change, can all lead to accidents. Misperception can arise as a result of insufficient processing, the omission of relevant information, or the receipt of wrong information. They influence driving habits, tolerance for risk, and judgement call. For example, Sagberg (1999), worked on the influence of vehicle operators drifting off to nap to the casualties. The ensuing perceptual inaccuracies have consequences for driving safety (Åberg et al., 2005; Eboli et al., 2017). Concentration and cognitive errors, like striking something that one hadn't noticed when reversing or trying to drive away from the road lights in the incorrect gear, are examples of lapses. Novice drivers (Girotto et al., 2016; Reeja and Jayaraj, 2017) or older drivers (Clarke et al., 2006; Zhang, 2010) are the most likely to make a lapse. Violations are departures from the highway code's written prohibitions and necessary restrictions for road users. Excessive speed or inside overtaking are two such examples. Violations seemed to be the most significantly associated to accidents, according to multiple studies (Poulter and Wann, 2013; Amin, 2020; Huang et al., 2018). Al-Balbissi (2003), worked on the influence of sex on the traffic casualties, which included surrounding environmental situations and the age of the driver. This study conducted statistical analysis on the data by utilizing 95% confidence level.

The study found that masculine vehicle operators are much more susceptible to casualties than their feminine counterparts. They also found that peoples aged more than 50 years old were far less likely to be bewildered in the casualties and were much more patient than other age groups. Moreover, Taylor et al., (2000) performed a thorough investigation on the effect of vehicle operators speeding upon that regularity of traffic collisions.

Surrounding conditions of road accidents

Road conditions, layout, lighting context, and weather circumstances also contribute a pivotal part in road accidents. To decrease the threat of harm amongst road users, suitable traffic organisation, altering walkways, bicycle lanes, safe crossing sites, and other signalized intersections measures must be implemented. According to (Przepiorka et al., 2020; Tement et al., 2020), population density is linked to the incidence of casualties. The population in urban areas is significantly more vulnerable to traffic accidents than the inhabitants of remote rural areas. Low-speed driving characterises urban traffic, but inter-urban travel needs greater speeds and uninterrupted flows. Regional and abroad highways have greater death rates versus urban highways. Abdalla et al. (1997) studied a database that was combination of demographic data and casualties in the road data. They found that the accidents were mainly prevalent near residential areas. Moreover, they explored the association among fatality value and indices of socially deprivation areas and found that people from those zones are much more susceptible to casualties than regions where mostly affluent peoples lived. This type of road is specifically constructed for high-speed vehicle traffic. Casselgren and Bodin (2017) states that, road fatalities are more plausible to occur on increased-speed routes as well as highways, but not amid congestions. Even if the overall danger on motorways was minimal, the data provided in (Hyodo et al., 2017) illustrated that the danger under bad road and weather circumstances was greater on highways in contrast to 2 lane and many-lane routes. As per Qiao and Shang (2011), mean speed of running, maximal and median degree of straight bends, quantity of upright bends, nature of road surface, period of day, and quantity of cars per hour are perhaps the most cogent determinants to traffic casualties in precarious places.

Techniques used to analyse and predict road accidents

Numerous researchers have looked at various elements of traffic casualties, with the majority focusing on scrutiny utilising ML techniques (Nandurge et al., 2017) and data mining (Li et al., 2017) approaches. Clustering techniques were utilized to discover high-accident locations, next utilized frequent patterns to find characteristics that influence road incidents in that place by Kumar and Toshniwal (2016). Another research done by these authors previously used k-modes cluster plus correlation methods to find probable relationships and correlations in a data mining framework for analysing road accidents (Kumar and Toshniwal, 2015). Chen et al. (2018) established a method for evaluating road accident clustering then organising those to one's relevance, resulting in the identification of maximum-risk sites. Hadoop framework was utilized in research works to manage massive datasets then create a prediction system to solve the issue of imbalance of highway road casualties' data. AHP and Apriori technique were being used by Xi et al. (2016) to identify accident variables and also to examine the extent of an incident or the amount of effect. Road casualties' patterns were examined, after that time series models were used to anticipate those in the coming years by (Yousefzadeh-Chabok et al., 2016). A BP neural network employed by (Ming and Xiucheng, 2009) to investigate the

association among incidence assessing score and potential accident numbers. Ossenbruggen et al. (2001), employed a logistic regression model to find statistical significance indicators that influence the occurrence of collisions and injuries, with the goal of applying those models to estimate the danger of a specific location. Such models were created using variables such as land uses, roadside layout, traffic control device usage, and traffic exposures to represent a location. Villages are less dangerous than residential and retail areas, according to their research.

3. Methodology

Here in this study, we will mainly perform two types of methodology and data cleaning to assess the traffic accident. First a macro methodology which will go through the bigger picture of the road accidents, then we will take a look at the dataset and do necessary pre-processing and at the end in the micro methodology we will perform various operations on the dataset to analyse and predict road traffic accidents.

3.1 Macro level Methodology

In this stage the dataset is collected and done all the pre-processing required to make ready dataset to do statistical and ML analysis on them. There are many techniques by which this can be done like SEMMA (Azevedo and Santos, 2008), KDD (Fayyad et al., 1996; Brachman and Anand, 1994), and CRISP-DM (Chapman et al., 2000). SEMMA consists of 5 stages and those are (i) Sapling the data (By taking a section of a huge data set vast sufficient to include the critical info but small enough to alter fast, you may sample the data), (ii) Exploring the dataset (Investigation of data and obtaining insight, ideas by looking for unexpected patterns and abnormalities), (iii) Modifying it (concentrate the models selection procedure by modifying the data by generating, choosing, and manipulating the elements), (iv) Modelling the data (By enabling the programme to look for an amalgamation of data that consistently forecasts a coveted result, you may model the data), and (v) Assessing the dataset (appraising the data by determining the relevance and trustworthiness of the data mining method' results and estimating how effectively it operates). This technique was put forward by the SAS institute. KDD or knowledge discovery in dataset is standardized process of data mining. KDD is method of obtaining whatever is regarded information utilising DM approaches according to the definition of measurements and standards, employing a database and any appropriate pre-treatment, sub sampling, and DB transformations (Azevedo and Santos, 2008). There are five steps to process data in KDD process, and they are: (i) Selection (Developing a goal set of data for discovery, or concentrating on a subset of data or samples), (ii) Pre-processing (cleansing and pre-processing of data with the goal of obtaining comparable data), (iii) Transformation (employing dimension chopping or translation strategies to convert data), (iv) Data Mining (Depending on the data mining purpose, scanning for trends of significance in a certain emblematical for), (v) Evaluation/Interpretation (The mined trends' translation plus assessment). The other technique CRISP-DM is an evolved version of KDD and is a very popular choice among data scientists. Data Processing will be done here mainly in 4 steps. Each of these steps are explained in the following paragraphs. Here is the overview of all the steps:

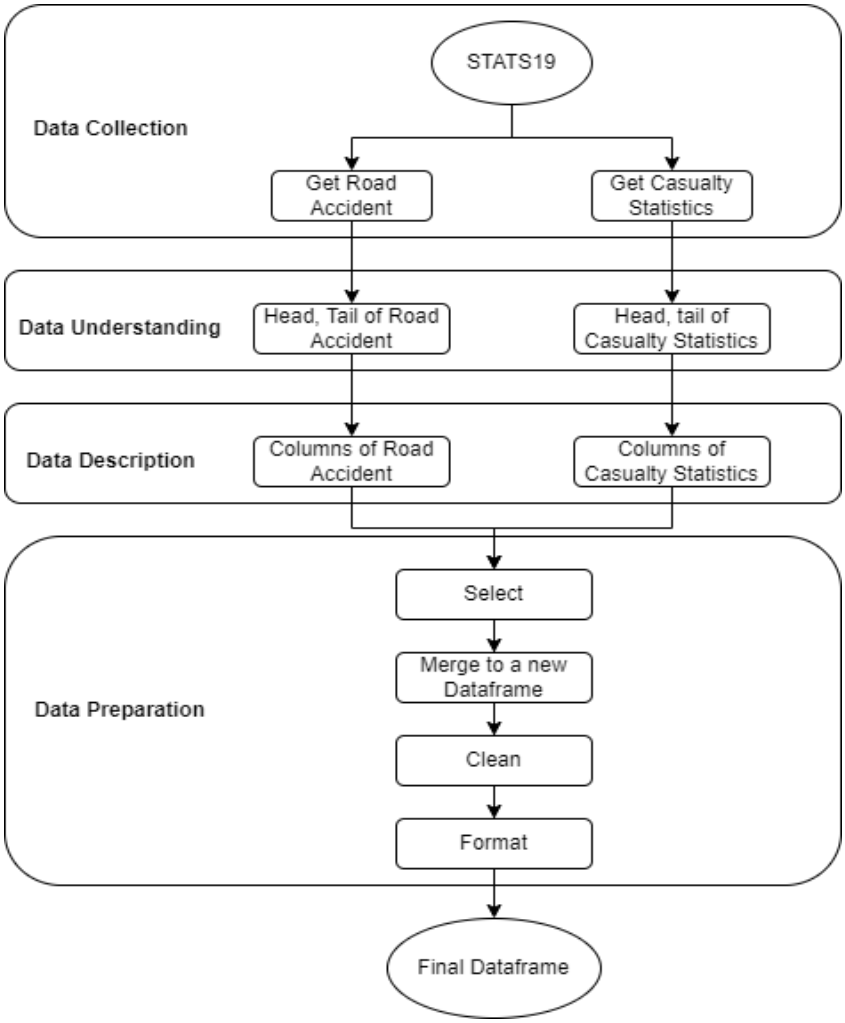


Figure 2: Data Processing stages

3.1.1 Data collection

Two different datasets were obtained from UK police department (STATS19 collection) for this research. The first dataset contains road accident information, which is mainly about the location, police response, and conditions of the accidents that happened across United Kingdom. The second dataset is comprised of the road casualty information and mainly holds information about the persons concerning those casualties in around UK. Both of these datasets are collected and represented in time series perspective and encompasses information over 5-year period from 2016 to 2020. As we are using two different datasets in this study it is imperative that we understand, clean, and merge these datasets properly. These procedures will be followed in the next steps.

3.1.2 Data Understanding

In this experiment, we are working with two datasets and before merging together let us take a look at those datasets. There are in total 597,972 records in the Road Accident dataset and for each record there are 35 informative columns. All the values in the dataset are numeric entities. Now, let us have a look at a first few values of the dataset:

	accident_index	accident_year	accident_reference	location_easting_osgr	location_northing_osgr	longitude	latitude	police_force	accident_severity
0	2.02E+12	2016	10000005	519310.0	188730.0	-0.279323	51.584754	1	3
1	2.02E+12	2016	10000006	551920.0	174560.0	0.184928	51.449595	1	3
2	2.02E+12	2016	10000008	505930.0	183850.0	-0.473837	51.543563	1	3
3	2.02E+12	2016	10000016	527770.0	168930.0	-0.164442	51.404958	1	3
4	2.02E+12	2016	10000018	510740.0	177230.0	-0.406580	51.483139	1	3

Figure 3: First few rows of the Road Accident dataset (Part 1)

number_of_vehicles	...	pedestrian_crossing_physical_facilities	light_conditions	weather_conditions	road_surface_conditions	special_conditions_at_site
2	...	0	5	1	1	0
1	...	0	4	1	1	0
1	...	0	4	1	1	0
1	...	0	1	1	1	0
2	...	0	1	1	1	0

Figure 4: First few rows of the Road Accident dataset (Part 2)

carriageway_hazards	urban_or_rural_area	did_police_officer_attend_scene_of_accident	trunk_road_flag	lsoa_of_accident_location
0	1	1	2	E01000543
0	1	1	2	E01000375
0	1	1	2	E01033725
0	1	1	2	E01003379
0	1	1	2	E01002583

Figure 5: First few rows of the Road Accident dataset (Part 3)

Last few rows of the Road accident dataset look like:

	accident_index	accident_year	accident_reference	location_easting_osgr	location_northing_osgr	longitude	latitude	police_force
597968	2.02E+12	2020	991027064	343034.0	731654.0	-2.926320	56.473539	99
597969	2.02E+12	2020	991029573	257963.0	658891.0	-4.267565	55.802353	99
597970	2.02E+12	2020	991030297	383664.0	810646.0	-2.271903	57.186317	99
597971	2.02E+12	2020	991030900	277161.0	674852.0	-3.968753	55.950940	99
597972	2.02E+12	2020	991032575	240402.0	681950.0	-4.561040	56.003843	99

Figure 6: Last few rows of the Road Accident dataset (Part 1)

accident_severity	number_of_vehicles	...	pedestrian_crossing_physical_facilities	light_conditions	weather_conditions	road_surface_conditions
2	2	...	0	1	1	1
3	1	...	0	1	1	1
2	2	...	0	1	1	1
3	2	...	0	1	1	1
3	1	...	0	1	1	1

Figure 7: Last few rows of the Road Accident dataset (Part 2)

special_conditions_at_site	carriageway_hazards	urban_or_rural_area	did_police_officer_attend_scene_of_accident	trunk_road_flag	lsoa_of_accident_location
0	0	1	1	-1	-1
0	0	1	2	-1	-1
0	0	2	1	-1	-1
0	0	1	2	-1	-1
0	2	1	1	-1	-1

Figure 8: Last few rows of the Road Accident dataset (Part 3)

In the 'Casualty Statistics' dataset, there are in total 781,716 data entries and each of these rows is represented by 18 different columns. Just like the Road accident dataset here also all the entries are numeric. Let us have a look at the first and last few rows of the dataset:

	accident_index	accident_year	accident_reference	vehicle_reference	casualty_reference	casualty_class	sex_of_casualty	age_of_casualty
0	2016010000005	2016	010000005	2	1	1	1	23
1	2016010000006	2016	010000006	1	1	1	2	36
2	2016010000008	2016	010000008	1	1	1	1	24
3	2016010000016	2016	010000016	1	1	2	2	59
4	2016010000018	2016	010000018	2	1	1	1	28

Figure 9: First few rows of the Casualty Statistics dataset (part 1)

age_band_of_casualty	casualty_severity	pedestrian_location	pedestrian_movement	car_passenger	bus_or_coach_passenger
5	3	0	0	0	0
7	3	0	0	0	0
5	3	0	0	0	0
9	3	0	0	0	3
6	3	0	0	0	0

Figure 10: First few rows of the Casualty Statistics dataset (part 2)

car_passenger	bus_or_coach_passenger	pedestrian_road_maintenance_worker	casualty_type	casualty_home_area_type	casualty_imd_decile
0	0	0	2	1	4
0	0	0	9	1	10
0	0	0	9	1	8
0	3	0	11	1	4
0	0	0	1	1	6

Figure 11: First few rows of the Casualty Statistics dataset (part 3)

Now, let us have a look at the last few rows of the dataset:

	accident_index	accident_year	accident_reference	vehicle_reference	casualty_reference	casualty_class	sex_of_casualty	age_of_casualty
781711	2020991027064	2020	991027064	2	1	1	1	11
781712	2020991029573	2020	991029573	1	1	3	2	63
781713	2020991030297	2020	991030297	2	1	1	1	38
781714	2020991030900	2020	991030900	2	1	1	1	76
781715	2020991032575	2020	991032575	1	1	3	1	48

Figure 12: Last few rows of the Casualty Statistics dataset (Part 1)

age_band_of_casualty	casualty_severity	pedestrian_location	pedestrian_movement	car_passenger	bus_or_coach_passenger
3	2	0	0	0	0
9	3	10	1	0	0
7	2	0	0	0	0
11	3	0	0	0	0
8	3	9	9	0	0

Figure 13: Last few rows of the Casualty Statistics dataset (Part 2)

car_passenger	bus_or_coach_passenger	pedestrian_road_maintenance_worker	casualty_type	casualty_home_area_type	casualty_imd_decile
0	0	0	1	1	2
0	0	0	0	1	10
0	0	0	5	2	9
0	0	0	1	1	9
0	0	0	0	1	1

Figure 14: Last few rows of the Casualty Statistics dataset (Part 3)

3.1.3 Data description

In this step we will have a closer look at those datasets by going through each column. There are in total 36 columns in the Road accident dataset. To get to know the dataset properly let us go through each column of the dataset:

Accident Index: Each accident has its own unique value. The accident index creates a unique ID by combining the accident year and the accident reference number. It may be utilised to connect to the Vehicle and accident networks.

Accident year: The year of the accident.

Accident reference: The police used the year to relate to a crash. Beyond the period, this is not distinct; use the incident index to link towards other periods.

Location easting osgr: Location related to easting osgr. It is kept Null if not available. In the dataset there are 123 Null values.

Location northing osgr: Location related to northing osgr. It is Null when empty. Just like location easting it also has 123 missing values.

Longitude: Longitude of the accident. Null when kept empty. There are 133 missing values in this column.

Latitude: Latitude of the accident. It's Null when data is not present on the record. 133 missing values here too.

```
accident_index                                  0
accident_year                                   0
accident_reference                              0
location_easting_osgr                         123
location_northing_osgr                        123
longitude                                     133
latitude                                      133
police_force                                    0
accident_severity                               0
number_of_vehicles                              0
number_of_casualties                            0
date                                            0
day_of_week                                     0
time                                            0
local_authority_district                        0
local_authority_ons_district                    0
local_authority_highway                         0
first_road_class                                0
first_road_number                               0
road_type                                       0
speed_limit                                    37
junction_detail                                 0
junction_control                                0
second_road_class                               0
second_road_number                              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
trunk_road_flag                                 0
lsoa_of_accident_location                       0
dtype: int64
```

Figure 15: Tally of Null values in the Road Accident dataset

Police force: The dataset contains records of in total 52 police stations which are numbered from 1 to 99. Each number represents an individual police station, for example: 1 represents Metropolitan police, 3 represents Cumbria police force. All the police force name and corresponding numbers can be found in the appendix.

Accident severity: This column is divided into three disparate brackets, they are Fatal, Serious, and Slight. In the dataset 1 is used for Fatal, 2 Represents Serious, and 3 is used Slight.

Number of vehicles: The total quantity of automobiles engaged in the collision.

#Number of casualties: Quantity of casualties happened in the incident.

Date: Specific date of the incident.

Day of week: In which day of the week the casualty happened. Numeric values from 1 to 7 is used to represent days of the week respectively from Sunday to Saturday.

Time: Time variable of the incidents.

Local authority district: In this dataset there are records of in total 416 districts. Here also, those are represented from number 1 to 941. For example, 931 represents North Ayrshire and 941 represents Western Isles. All the names and corresponding district numbers are given in the appendix.

Local authority ons district: In total 422 categories are in this field. These are represented by combining a letter (Either E, S, or W) followed by 8-digit numbers. For example: S12000046 represents Glasgow city, and W06000024 represents Merthyr Tydfil. Look at the appendix to see all the values.

Local authority highway: Local authority highway is also represented in the same way just like local authority ons district, one letter followed by 8-digit numbers. In this there are 207 local highway authority. See appendix for full details.

First road class: There are 6 kinds of data here which is represented using numbers 1 to 6. This is the first entry of road class. Here, Motorway is represented by 1 and 6 is used to denote unclassified.

First road number: Numbers between 1 to 9999 is used to illustrate the road number. -1 is used for unknown road number.

Road type: There are 8 types of roads in the dataset. Here -1 renders missing data and all the other numbers from 1 to 12 stands for different road types.

Speed limit: We know speed contributes a pivotal part in traffic casualties and may be one of the most important contributors to road accidents. That is why disparate routes have dissimilar speed limits and all those limits are directly given in the dataset if the record is available. For missing or out of range values -1 is used and if the speed limit is unknown/self-reported 99 is used.

Junction detail: This column depicts in what kind of junction the accident happened. In this dataset there are mainly 9 types of junctions and 99/-1 is used for self-reported and missing junction detail entries.

Junction control: What kind of junction control was in place. For instant, was it an authorised person or auto traffic signal controlling the signal? This data is given in this column.

Second road class: Same as first road class and is used to depict additional data.

Second road number: Similar to first road number.

Pedestrian crossing human control: How was the pedestrians regulated in the place of incident is illustrated in this field. 0 is used if there was no one to control the pedestrians within 50 meters, 1 for controlled by school authorised patrol, and 2 for controlled by sanctioned person.

Pedestrian crossing physical facilities: This data is given to understand if there were any physical facilities like Zebra crossing or footbridge in the place of accident for crossing of

pedestrians. Each type of pedestrian crossing physical facility with corresponding numeric value is given in the appendix.

Light conditions: what kind of light condition was there in the time of accident is represented in this column. Here, darkness is additionally divided into 4 different classes on was there lighting in the place or not.

Weather conditions: Weather conditions are divided into 8 different classes to understand how the weather was like at the moment of experiment.

Road surface conditions: what was the surface of the road like at the place of incident is given in this column. 7 types of road surfaces are considered here and details of those are in the appendix.

Special conditions at site: To understand any exceptional situation at the site like roadblocks or mud this column is used. Here, mainly 7 types of special conditions data are given.

Carriageway hazards: Was there any dog on the carriageway at the time of accident? Or was there any pedestrian on the road? These kinds of information can be found in this column. In this dataset 7 kinds of carriageway hazards are considered.

Urban or rural area: Details of urban or rural area where the incident occurred is given in this column.

Did police officer attend scene of accident: Was the incident delineated by an individual or the police attended the scene is answered in this column.

Trunk road flag: If the road is managed by Highways England or not is given in this column. 1 is used for trunk and 2 is used for non-trunk.

Lsoa of accident location: This field contains information about England and Wales, and whole Great Britain is not covered in here.

To get a basic view of this dataset let us have a look a at count, mean, standard deviation, min, max, etc of each of the columns of the dataset:

	count	mean	std	min	25%	50%	75%	max
accident_year	597973.0	2017.827266	1.381319	2016.000000	2017.000000	2018.000000	2019.000000	2.020000e+03
location_easting_osgr	597850.0	451703.358935	94929.997358	64084.000000	388020.000000	457990.500000	528980.000000	6.553910e+05
location_northing_osgr	597850.0	282043.804647	152864.644216	10235.000000	175705.000000	222711.500000	387381.000000	1.209512e+06
longitude	597840.0	-1.260225	1.392031	-7.525273	-2.180570	-1.143562	-0.141011	1.759641e+00
latitude	597840.0	52.425656	1.376865	49.912362	51.467411	51.888019	53.380777	6.076372e+01
police_force	597973.0	28.574257	24.915142	1.000000	5.000000	23.000000	45.000000	9.900000e+01
accident_severity	597973.0	2.792099	0.437881	1.000000	3.000000	3.000000	3.000000	3.000000e+00
number_of_vehicles	597973.0	1.842209	0.709933	1.000000	1.000000	2.000000	2.000000	2.400000e+01
number_of_casualties	597973.0	1.307276	0.756494	1.000000	1.000000	1.000000	1.000000	5.900000e+01
day_of_week	597973.0	4.109935	1.924954	1.000000	2.000000	4.000000	6.000000	7.000000e+00
local_authority_district	597973.0	328.621888	257.290799	-1.000000	90.000000	305.000000	513.000000	9.410000e+02
first_road_class	597973.0	4.197810	1.463992	1.000000	3.000000	4.000000	6.000000	6.000000e+00
first_road_number	597973.0	778.149082	1580.905318	0.000000	0.000000	30.000000	523.000000	9.302000e+03
road_type	597973.0	5.196603	1.689462	-1.000000	6.000000	6.000000	6.000000	9.000000e+00
speed_limit	597936.0	37.137533	14.027131	-1.000000	30.000000	30.000000	40.000000	7.000000e+01
junction_detail	597973.0	2.991710	8.650574	-1.000000	0.000000	2.000000	3.000000	9.900000e+01
junction_control	597973.0	1.736229	2.495472	-1.000000	-1.000000	2.000000	4.000000	9.000000e+00
second_road_class	597973.0	2.685998	3.158066	-1.000000	-1.000000	3.000000	6.000000	9.000000e+00
second_road_number	597973.0	219.168503	926.539753	-1.000000	-1.000000	0.000000	0.000000	9.177000e+03
pedestrian_crossing human_control	597973.0	0.204884	1.289745	-1.000000	0.000000	0.000000	0.000000	9.000000e+00
pedestrian_crossing_physical_facilities	597973.0	1.044415	2.260054	-1.000000	0.000000	0.000000	0.000000	9.000000e+00
light_conditions	597973.0	2.024188	1.714039	-1.000000	1.000000	1.000000	4.000000	7.000000e+00
weather_conditions	597973.0	1.627798	1.791257	-1.000000	1.000000	1.000000	1.000000	9.000000e+00
road_surface_conditions	597973.0	1.363968	0.886029	-1.000000	1.000000	1.000000	2.000000	9.000000e+00
special_conditions_at_site	597973.0	0.195765	1.163435	-1.000000	0.000000	0.000000	0.000000	9.000000e+00
carriageway_hazards	597973.0	0.151627	1.049526	-1.000000	0.000000	0.000000	0.000000	9.000000e+00
urban_or_rural_area	597973.0	1.331642	0.471219	-1.000000	1.000000	1.000000	2.000000	3.000000e+00
_police_officer_attend_scene_of_accident	597973.0	1.316391	0.537858	-1.000000	1.000000	1.000000	2.000000	3.000000e+00
trunk_road_flag	597973.0	1.669672	0.863693	-1.000000	2.000000	2.000000	2.000000	2.000000e+00

Figure 16: Description of each column of the Road accident dataset

There are 18 columns in the Casualty statistics dataset. Description of each of those columns is given below:

Accident index: Same as the road accident dataset.

Accident year: Same as the road accident dataset.

Accident reference: Same as the road accident dataset.

Vehicle reference: Each car in a single accident has a distinct value. It's possible to attach a Casualty to a car using this.

Casualty reference: Each casualty in a single accident has a distinct value. A single car in a traffic collision may have its own set of historical years.

Casualty class: Describes to whom the casualty has happened to. Driver is classified as 1, passenger as 2, and pedestrian as 3.

```
accident_index                          0
accident_year                           0
accident_reference                      0
vehicle_reference                       0
casualty_reference                      0
casualty_class                          0
sex_of_casualty                         0
age_of_casualty                         0
age_band_of_casualty                    0
casualty_severity                       0
pedestrian_location                     0
pedestrian_movement                     0
car_passenger                           0
bus_or_coach_passenger                  0
pedestrian_road_maintenance_worker      0
casualty_type                           0
casualty_home_area_type                 0
casualty_imd_decile                     0
dtype: int64
```

Figure 17: Number of Null values in the Casualty statistics dataset

Sex of casualty: This column denotes the sexual orientation of the person bewildered in the casualty.

Age of casualty: Age of the person involved in the casualty is given in this column.

Age band of casualty: Age of people involved in the casualty is divided into equally spaced 11 groups where difference between each group is 5 years

Casualty severity: same as accident severity of the road accidents dataset.

Pedestrian location: If in the casualty pedestrian is involved in this field the location of the pedestrian can be found. If the data is missing its -1 and for unknown, it's 0.

Pedestrian movement: Was the pedestrian crossing from drivers near/far side, was he in a carriageway, or was he walking along the carriageway? This information is given in this column.

Car passenger: If there was any involvement of car passenger in the accident, they are divided into 2 groups. 1 is for the front seat passenger and 2 is for the back seat passenger.

Bus or coach passenger: This entity gives information bus or coach passengers like the previous column. It can be Boarding or Alighting or standing passenger or seated passenger.

Pedestrian road maintenance worker: Road maintenance can sometimes cause casualties. If it was the case the data can be found in this column. For missing or out of range values -1 is used. Details of each type and it's corresponding numeric value can be found in the index.

Casualty type: In this dataset there are recorded 30 types of casualties' record. These are also represented by number values details of which can be found in the appendix section.

Casualty home area type: In what kind of area the casualty occurred the data is given here. This is similar to the Urban or rural area entity of road accidents dataset. For missing or out of range

values -1 is used. Details of each type and its corresponding numeric value can be found in the index.

To get a better interpretation of this casualty statistics dataset let us have a look at count, mean, standard deviation, min, max, etc of each of the columns of the dataset:

	count	mean	std	min	25%	50%	75%	max
accident_year	781716.0	2017.808837	1.377568	2016.0	2017.0	2018.0	2019.0	2020.0
vehicle_reference	781716.0	1.477553	1.711188	1.0	1.0	1.0	2.0	999.0
casualty_reference	781716.0	1.393182	2.454991	1.0	1.0	1.0	1.0	992.0
casualty_class	781716.0	1.485053	0.722330	1.0	1.0	1.0	2.0	3.0
sex_of_casualty	781716.0	1.395642	0.502698	-1.0	1.0	1.0	2.0	9.0
age_of_casualty	781716.0	36.660602	19.393478	-1.0	22.0	33.0	50.0	102.0
age_band_of_casualty	781716.0	6.306182	2.414841	-1.0	5.0	6.0	8.0	11.0
casualty_severity	781716.0	2.825593	0.407362	1.0	3.0	3.0	3.0	3.0
pedestrian_location	781716.0	0.730536	2.080773	-1.0	0.0	0.0	0.0	10.0
pedestrian_movement	781716.0	0.570428	1.871116	-1.0	0.0	0.0	0.0	9.0
car_passenger	781716.0	0.252173	0.626313	-1.0	0.0	0.0	0.0	9.0
bus_or_coach_passenger	781716.0	0.068772	0.501691	-1.0	0.0	0.0	0.0	9.0
pedestrian_road_maintenance_worker	781716.0	0.059004	0.340125	-1.0	0.0	0.0	0.0	3.0
casualty_type	781716.0	7.270959	8.371928	-1.0	2.0	9.0	9.0	99.0
casualty_home_area_type	781716.0	1.009290	0.977786	-1.0	1.0	1.0	1.0	3.0
casualty_imd_decile	781716.0	4.232354	3.288848	-1.0	2.0	4.0	7.0	10.0

Figure 18: Description of each column of the casualty statistics dataset

3.1.4 Data preparation

As there are two datasets in this study this data preparation is one of the key steps of the research. First of all, both of these datasets are loaded in the python Jupyter notebook as Dataframe's.

Select: In order to do analysis, the required data frames are selected. This entails taking into account the goals as well as any limitations imposed by the data itself. In case of two datasets, it is inspected that if there were any common elements to merge the datasets on. After close inspection it was found that all the records in road accident datasets have a similar accident reference number in casualty statistics dataset. So basically, on common rows these records can be merged together.

Merge: In the road accident data frame there were in total 597,973 data records and 781,716 in casualty statistics data frame. So, in this step these data frames are merged together via inner joins to a new data frame to get all the matched rows. After merging together, the data frames it was found that the new dataframe has 597,973 entities. So, all the data points in the road accident dataset were also present in the casualty statistics dataset.

There were in total 36 columns in road accident data frame and 18 columns in the casualty statistics dataframe. The newly created data frame has 54 columns. So, at this step it was necessary to clean the dataframe and remove all the unnecessary columns that won't be used in this experiment. Let us have a look at the first few columns of the newly merged dataframe:

	0	1	2	3	4
accident_index	2.02E+12	2.02E+12	2.02E+12	2.02E+12	2.02E+12
accident_year	2016	2016	2016	2016	2016
accident_reference	10000005	10000006	10000008	10000016	10000018
location_easting_osgr	519310.0	551920.0	505930.0	527770.0	510740.0
location_northing_osgr	188730.0	174560.0	183850.0	168930.0	177230.0
longitude	-0.279323	0.184928	-0.473837	-0.164442	-0.40658
latitude	51.584754	51.449595	51.543563	51.404958	51.483139
police_force	1	1	1	1	1
accident_severity	3	3	3	3	3
number_of_vehicles	2	1	1	1	2
number_of_casualties	1	1	1	1	1
date	1/11/2016	1/11/2016	1/11/2016	1/11/2016	1/11/2016
day_of_week	3	3	3	3	3
time	2:30	0:37	1:25	9:15	7:53
local_authority_district	28	18	26	22	25
local_authority_ons_district	E09000005	E09000004	E09000017	E09000024	E09000018
local_authority_highway	E09000005	E09000004	E09000017	E09000024	E09000018
first_road_class	3	3	3	3	3
first_road_number	4006	207	4020	217	312
road_type	6	6	1	6	3
speed_limit	30.0	30.0	30.0	30.0	40.0
junction_detail	0	9	1	3	0
junction_control	-1	4	4	2	-1
second_road_class	-1	6	3	3	-1
second_road_number	-1	0	4020	217	-1
pedestrian_crossing human_control	0	0	0	0	0
pedestrian_crossing_physical_facilities	0	0	0	0	0

Figure 19: First few columns of merged dataset (part 1)

light_conditions	5	4	4	1	1
weather_conditions	1	1	1	1	1
road_surface_conditions	1	1	1	1	1
special_conditions_at_site	0	0	0	0	0
carriageway_hazards	0	0	0	0	0
urban_or_rural_area	1	1	1	1	1
did_police_officer_attend_scene_of_accident	1	1	1	1	1
trunk_road_flag	2	2	2	2	2
lsoa_of_accident_location	E01000543	E01000375	E01033725	E01003379	E01002583
accident_index	2016010000005	2016010000006	2016010000008	2016010000016	2016010000018
accident_year	2016	2016	2016	2016	2016
accident_reference	010000005	010000006	010000008	010000016	010000018
vehicle_reference	2	1	1	1	2
casualty_reference	1	1	1	1	1
casualty_class	1	1	1	2	1
sex_of_casualty	1	2	1	2	1
age_of_casualty	23	36	24	59	28
age_band_of_casualty	5	7	5	9	6
casualty_severity	3	3	3	3	3
pedestrian_location	0	0	0	0	0
pedestrian_movement	0	0	0	0	0
car_passenger	0	0	0	0	0
bus_or_coach_passenger	0	0	0	3	0
pedestrian_road_maintenance_worker	0	0	0	0	0
casualty_type	2	9	9	11	1
casualty_home_area_type	1	1	1	1	1
casualty_imd_decile	4	10	8	4	6

Figure 20: First few columns of merged dataset (part 2)

Clean: The clean data step entails analysing the selected data to determine whether it is acceptable for the analysis' intended aims or whether additional activities are necessary. lost data, data mistakes, and coding discrepancies should all be considered. After merging the datasets, it was found that there are 3 identical features (Accident Index, Accident year, Accident reference). So, duplicate columns were removed from the dataframe.

```
Index(['accident_index', 'accident_year', 'accident_reference',
       'location_easting_osgr', 'location_northing_osgr', 'longitude',
       'latitude', 'police_force', 'accident_severity', 'number_of_vehicles',
       'number_of_casualties', 'date', 'day_of_week', 'time',
       'local_authority_district', 'local_authority_ons_district',
       'local_authority_highway', 'first_road_class', 'first_road_number',
       'road_type', 'speed_limit', 'junction_detail', 'junction_control',
       'second_road_class', 'second_road_number',
       'pedestrian_crossing human_control',
       'pedestrian_crossing_physical_facilities', 'light_conditions',
       'weather_conditions', 'road_surface_conditions',
       'special_conditions_at_site', 'carriageway_hazards',
       'urban_or_rural_area', 'did_police_officer_attend_scene_of_accident',
       'trunk_road_flag', 'lsoa_of_accident_location', 'vehicle_reference',
       'casualty_reference', 'casualty_class', 'sex_of_casualty',
       'age_of_casualty', 'age_band_of_casualty', 'casualty_severity',
       'pedestrian_location', 'pedestrian_movement', 'car_passenger',
       'bus_or_coach_passenger', 'pedestrian_road_maintenance_worker',
       'casualty_type', 'casualty_home_area_type', 'casualty_imd_decile'],
      dtype='object')
(597973, 51)
```

Figure 21: Columns after removing duplicates

Then the amount of null values in the dataset was looked at:

```
accident_index                                0
accident_year                                 0
accident_reference                            0
location_easting_osgr                       123
location_northing_osgr                      123
longitude                                   133
latitude                                    133
police_force                                  0
accident_severity                             0
number_of_vehicles                            0
number_of_casualties                          0
date                                          0
day_of_week                                   0
time                                          0
local_authority_district                      0
local_authority_ons_district                  0
local_authority_highway                       0
first_road_class                              0
first_road_number                             0
road_type                                     0
speed_limit                                  37
junction_detail                               0
junction_control                              0
second_road_class                             0
second_road_number                            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
trunk_road_flag                               0
lsoa_of_accident_location                     0
vehicle_reference                             0
casualty_reference                            0
casualty_class                                0
sex_of_casualty                               0
age_of_casualty                               0
age_band_of_casualty                          0
casualty_severity                             0
pedestrian_location                           0
pedestrian_movement                           0
car_passenger                                 0
bus_or_coach_passenger                        0
pedestrian_road_maintenance_worker            0
casualty_type                                 0
casualty_home_area_type                       0
casualty_imd_decile                           0
dtype: int64
```

Figure 22: Null values in the dataframe

So, there are only 5 columns that has null values. At this stage the dataset is closely inspected to find all the columns that won't be useful in our research, so that the dataset looks concise and clean. From the road accident dataset accident index, accident reference, location easting osgr, location northing osgr, police force, number of casualties, local authority ons district, local authority highway, first road number, second road class, second road number, urban or rural area, did police officer attend scene of accident, pedestrian crossing human control, pedestrian crossing physical facilities, pedestrian movement, pedestrian location, trunk road flag, lsoa of accident location is removed. From the casualty statistics dataset the same was done for columns named vehicle reference, casualty reference, casualty class, car passenger, bus or coach passenger, pedestrian road maintenance worker, casualty imd domicile, pedestrian road maintenance worker. After removing these columns finally there were 24 columns in the dataframe.

```
Index(['accident_year', 'date', 'day_of_week', 'time', 'longitude', 'latitude',
       'local_authority_district', 'casualty_home_area_type',
       'sex_of_casualty', 'age_of_casualty', 'age_band_of_casualty',
       'accident_severity', 'casualty_type', 'number_of_vehicles',
       'first_road_class', 'road_type', 'speed_limit', 'junction_detail',
       'junction_control', 'light_conditions', 'weather_conditions',
       'road_surface_conditions', 'special_conditions_at_site',
       'carriageway_hazards'],
      dtype='object')
(597973, 24)
```

Figure 23: Column names and shape after data cleaning

Format: The final stage in data preparation is to prepare the dataframe so that it could be utilised by the modelling programme. The data does not need to be restructured in this scenario since the format was created while loading the dataset in the dataframe, as previously mentioned.

3.1.5 Data Quality

All personal injury highway traffic incident reported to the authorities is documented in an administrative structure with a statistical reporting component. Through using Stats19 variables, the statistical collection is carried out in a consistent manner across the United Kingdom. The data are often collected mostly by law enforcement officer reporting the incident in the MG NSRF form or a locally designed form (which incorporates the basic Stats19 variables). Conversely, within about one third of the cases, it is documented by citizens at a police station some times after the event; this is for cases where no police officer was present at the casualty location. As people always don't report incidents to the police these datasets obtained from UK police department may not fully represent all the accidents and casualties that happen across UK. These data cover all incidents that were recorded by the department of police that happened on a highway entangling minimum one motorcycle, horse rider, or pedal cycle, and in which minimum one person was wounded. Even though a lot of non-fatal accident are not recorded in the law enforcement department, after comparing these road accident and casualty statistics dataset with death registration, survey, hospital, and insurance claims a very few numbers of incidents were found to be not reported. The Stats19 variables are used to offer

extensive information on the accident, the circumstances surrounding it, the cars involved, and the casualties that resulted. The data is authenticated through different processes (conducted by police departments, municipal highway bodies, and national governments) before being transmitted to the Department of Transportation for national statistical reasons. The information is also shared with local road authority, who are legally responsible for promoting road safety. This data also revised every 5 years.

3.3 Micro Level Methodology

Analysis and prediction of the datasets are done by using different statistical analysis tools, visualisation tools, machine learning tools and deep learning tools in four distinct stages. In the first level various descriptive statistical analysis tools are used to find the number of items, number of unique values of dataset, how much spread out the data entities are, minimum value for each column, maximum value of each column, if the data is skewed or not, quartile values of each column, make boxplot of the entities, show histogram of the datasets, visualize using bar chart, scatter plot, wave fig and other types of diagrams. In the second stage various inferential statistical tools such as 1 sample T-test, 2 sample T-test, Matched pair T-test to analyse weather there is enough difference among the means of two samples, One way ANOVA (Analysis of Variance), Pearson's test to find the degree of linear correlation among two numeric entities, Heatmaps for visualising the correlations, Chi-square test of independence to examine if there is any dependence among 2 different categorical entities.

In the third stage various Machine learning algorithms will be used to make prediction of road casualties' severity such as Logistic regression, Random Forest classifier, KNN, Decision tree, naïve bayes. In fourth level of analysis disparate Deep learning algorithms such as LSTM, Neural network, GRU will be used to make prediction of road casualties.

4. Results and Discussion

After completing all the pre-processing with the dataset, the first few columns of the dataset:

	accident_year	date	day_of_week	time	longitude	latitude	local_authority_district	casualty_home_area_type	sex_of_casualty	age_of_casualty
0	2016	1/11/2016	3	2:30	-0.279323	51.584754	28	1	1	23
1	2016	1/11/2016	3	0:37	0.184928	51.449595	18	1	2	36
2	2016	1/11/2016	3	1:25	-0.473837	51.543563	26	1	1	24
3	2016	1/11/2016	3	9:15	-0.164442	51.404958	22	1	2	59
4	2016	1/11/2016	3	7:53	-0.406580	51.483139	25	1	1	28

oad_class	road_type	speed_limit	junction_detail	junction_control	light_conditions	weather_conditions	road_surface_conditions	special_conditions_at_site
3	6	30.0	0	-1	5	1	1	0
3	6	30.0	9	4	4	1	1	0
3	1	30.0	1	4	4	1	1	0
3	6	30.0	3	2	1	1	1	0
3	3	40.0	0	-1	1	1	1	0

Figure 24: First few columns of final data

4.1 Descriptive Statistical Analysis

According to Vergura et al. (2009), descriptive statistics can be defined as 'A branch of statistics that is useful for gathering, categorising, as well as scrutinizing a known collection of data, referred to as a population, by assigning a correct comprehensive model or distribution group to it'. The goal of this level is to do many of these descriptive statistical inspect on the data. The figure below shows the mean, max, min, and other descriptions of the dataset.

	count	mean	std	min	25%	50%	75%	max
accident_year	597973.0	2017.827266	1.381319	2016.000000	2017.000000	2018.000000	2019.000000	2020.000000
day_of_week	597973.0	4.109935	1.924954	1.000000	2.000000	4.000000	6.000000	7.000000
longitude	597840.0	-1.260225	1.392031	-7.525273	-2.180570	-1.143562	-0.141011	1.759641
latitude	597840.0	52.425656	1.376865	49.912362	51.467411	51.888019	53.380777	60.763722
local_authority_district	597973.0	328.621888	257.290799	-1.000000	90.000000	305.000000	513.000000	941.000000
casualty_home_area_type	597973.0	0.974205	0.982771	-1.000000	1.000000	1.000000	1.000000	3.000000
sex_of_casualty	597973.0	1.401291	0.498781	-1.000000	1.000000	1.000000	2.000000	9.000000
age_of_casualty	597973.0	36.520527	19.375982	-1.000000	22.000000	33.000000	50.000000	102.000000
age_band_of_casualty	597973.0	6.290008	2.414589	-1.000000	5.000000	6.000000	8.000000	11.000000
accident_severity	597973.0	2.792099	0.437881	1.000000	3.000000	3.000000	3.000000	3.000000
casualty_type	597973.0	7.211219	8.089992	-1.000000	2.000000	9.000000	9.000000	99.000000
number_of_vehicles	597973.0	1.842209	0.709933	1.000000	1.000000	2.000000	2.000000	24.000000
first_road_class	597973.0	4.197810	1.463992	1.000000	3.000000	4.000000	6.000000	6.000000
road_type	597973.0	5.196603	1.689462	-1.000000	6.000000	6.000000	6.000000	9.000000
speed_limit	597936.0	37.137533	14.027131	-1.000000	30.000000	30.000000	40.000000	70.000000
junction_detail	597973.0	2.991710	8.650574	-1.000000	0.000000	2.000000	3.000000	99.000000
junction_control	597973.0	1.736229	2.495472	-1.000000	-1.000000	2.000000	4.000000	9.000000
light_conditions	597973.0	2.024188	1.714039	-1.000000	1.000000	1.000000	4.000000	7.000000
weather_conditions	597973.0	1.627798	1.791257	-1.000000	1.000000	1.000000	1.000000	9.000000
road_surface_conditions	597973.0	1.363968	0.886029	-1.000000	1.000000	1.000000	2.000000	9.000000
special_conditions_at_site	597973.0	0.195765	1.163435	-1.000000	0.000000	0.000000	0.000000	9.000000
carriageway_hazards	597973.0	0.151627	1.049526	-1.000000	0.000000	0.000000	0.000000	9.000000

Figure 25: Statistical description of the dataset

Following is the histogram of all the features of the dataset:

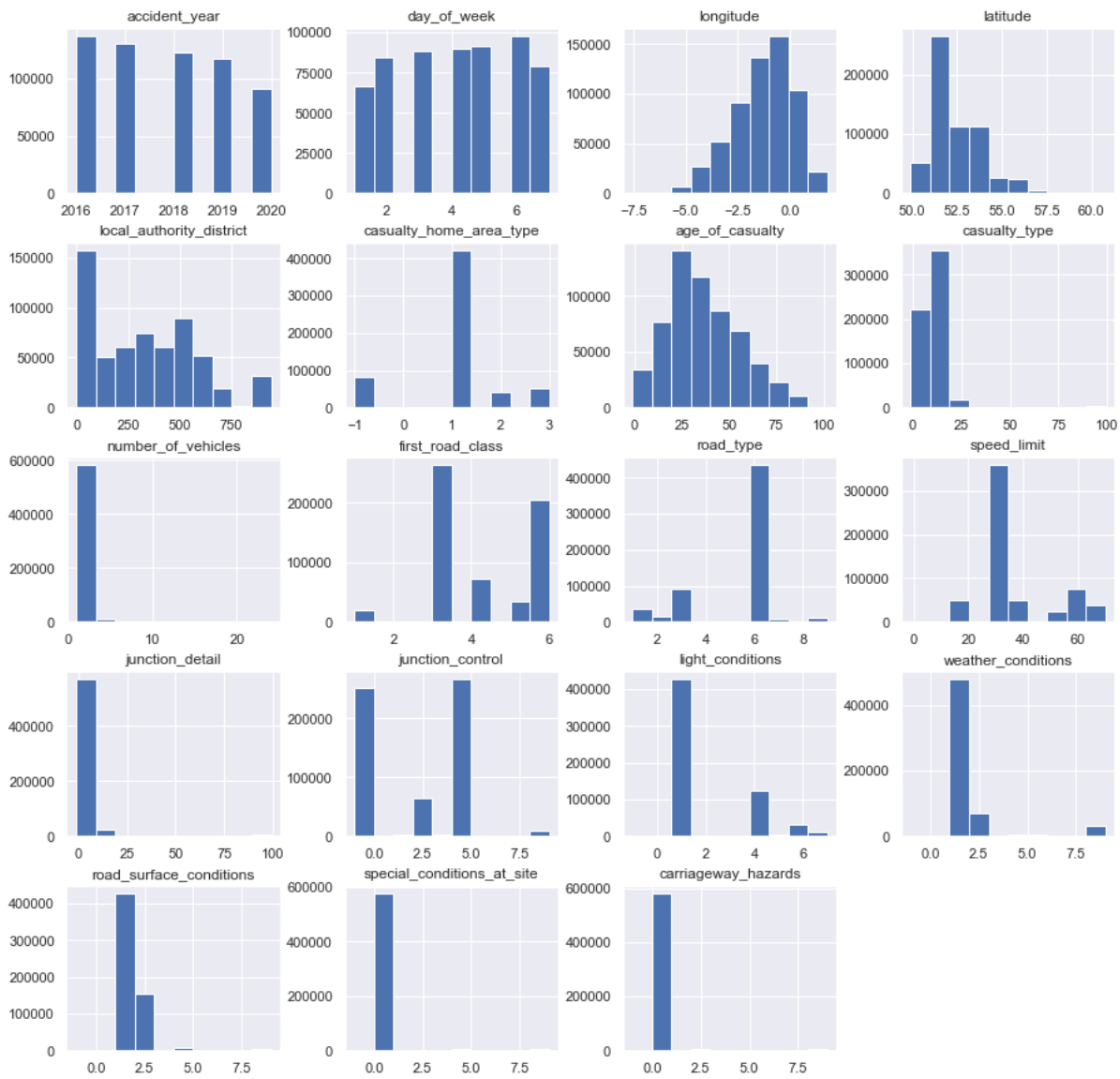


Figure 26: Features histogram

Time:

To get the intuition of time of the accidents it is first looked at the trend of accidents over the years. The following bar graph illustrates the trend of casualties over the years. From the graph it can be interpreted that over the years the amount of accidnets has been decreasing.

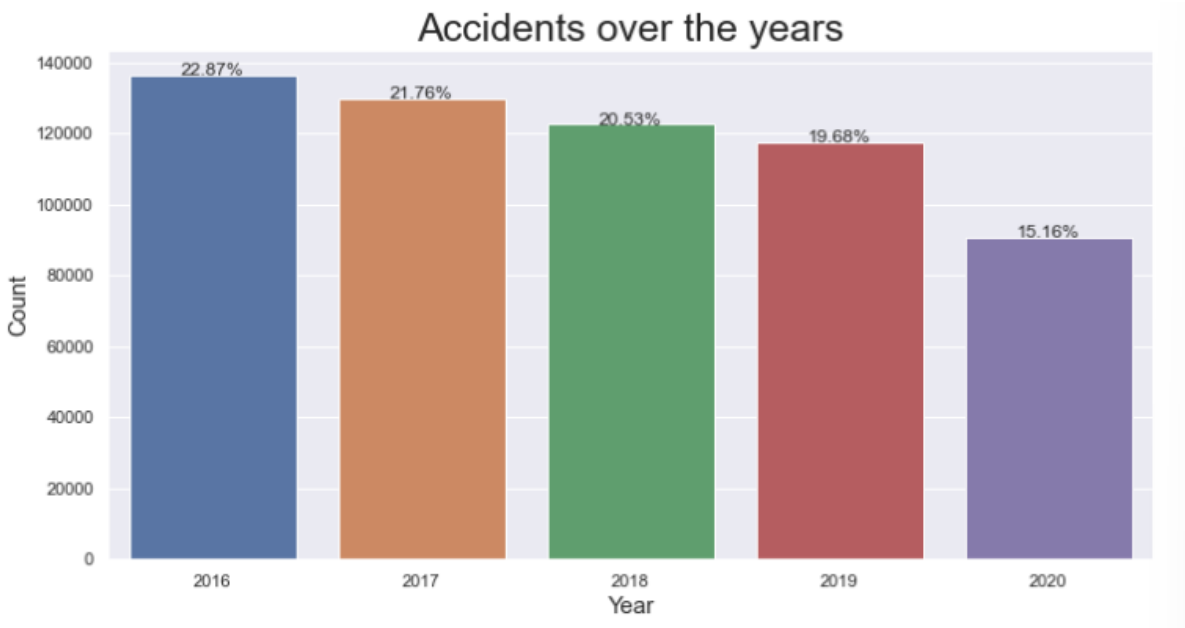


Figure 27: Road accidents over last 5 years

At this stage the trend of accidents at each month is looked at. The figure below shows the quantity of average casualties that occur each month over last 5 years. From the line graph it can be interpreted that in the month of November most accidents and the number is lowest at the month of April.

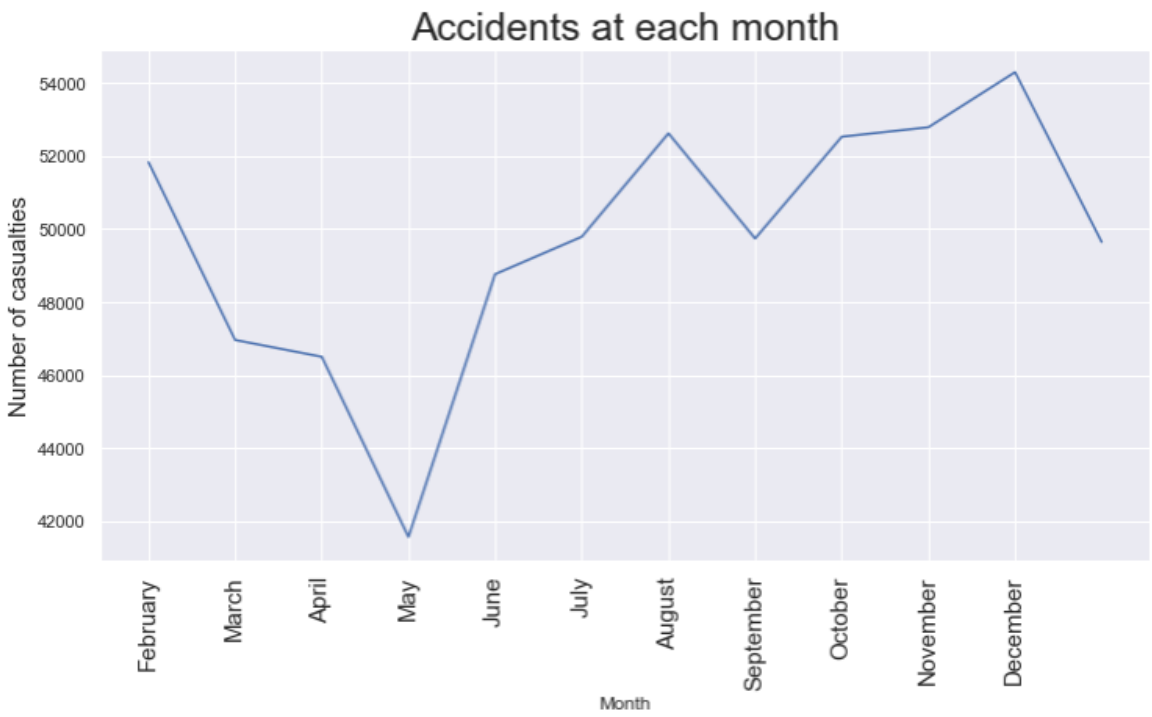


Figure 28: Trend of accidents at each month

The number of accidents and casualties at each month of every year can also be interpretated from the following figure:

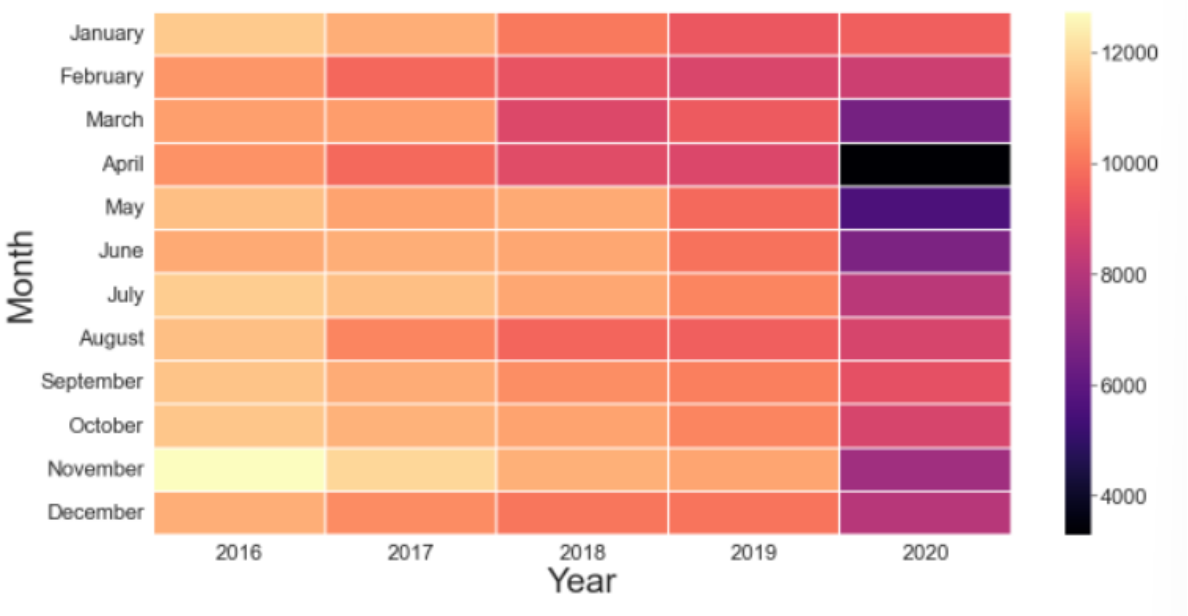


Figure 29: Accidents at each month of each week

From the bar graph it can be observed that Friday has the greatest quantity of incidents and on Sunday least amount of casualties took place, which makes sense as most individuals are in rush to go enjoy their vacation on Friday and on Sunday people mostly stay at home which corresponds to least number of road accidents.

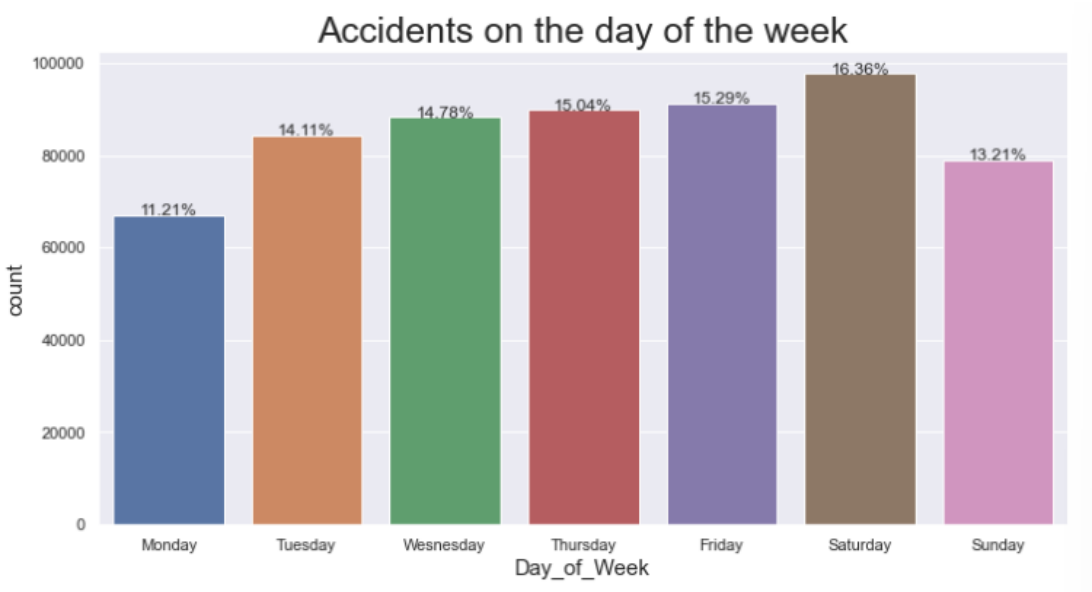


Figure 30: Road casualties in each day of the week

In the next figure the number of accidents at every hour of the day is illustrated. It can be observed that the quantity of accident is highest at evening and it's the least after 12 in the night.

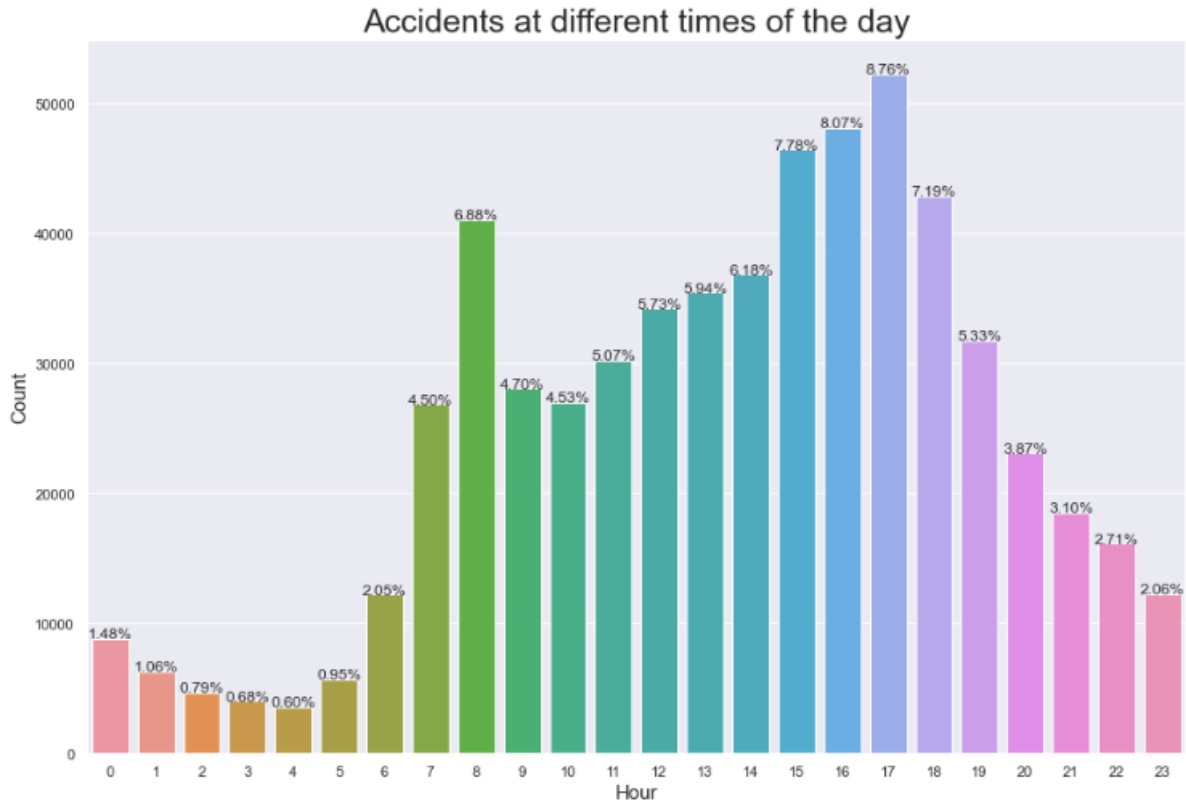


Figure 31: Accidents to the time of the day

Following time series graph shows the accidents over the years:

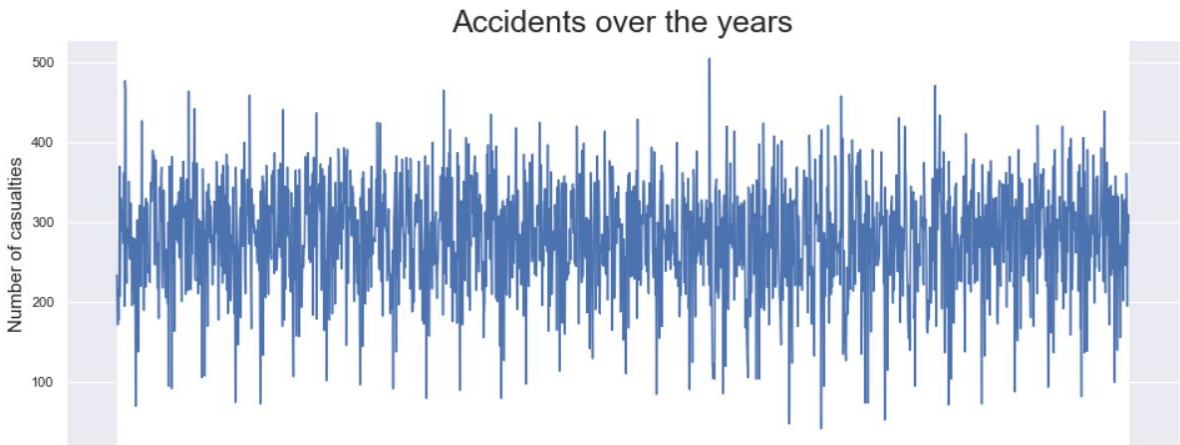


Figure 32: Number of accidents with time

Location

To visualize where the accidents were in different locations of UK longitude and latitude information of the dataset accidents were put on the map using scatterplot. However, intuitively this actually depends on how much traffic each area has, as busy areas might produce more accidents. But it is also possible that a specific road because of its curve/visibility result in more accidents.

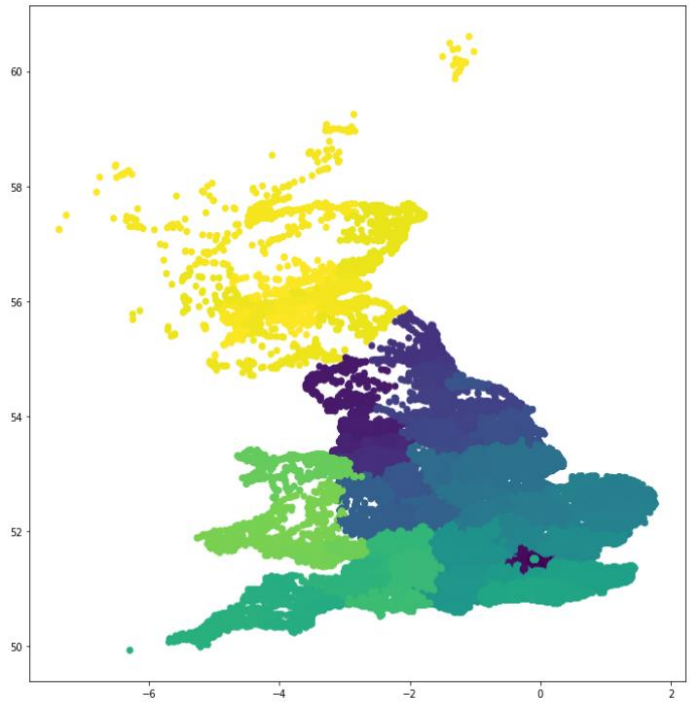


Figure 33: Accidents in UK map

In the following map the accidents are shown according to their severity:

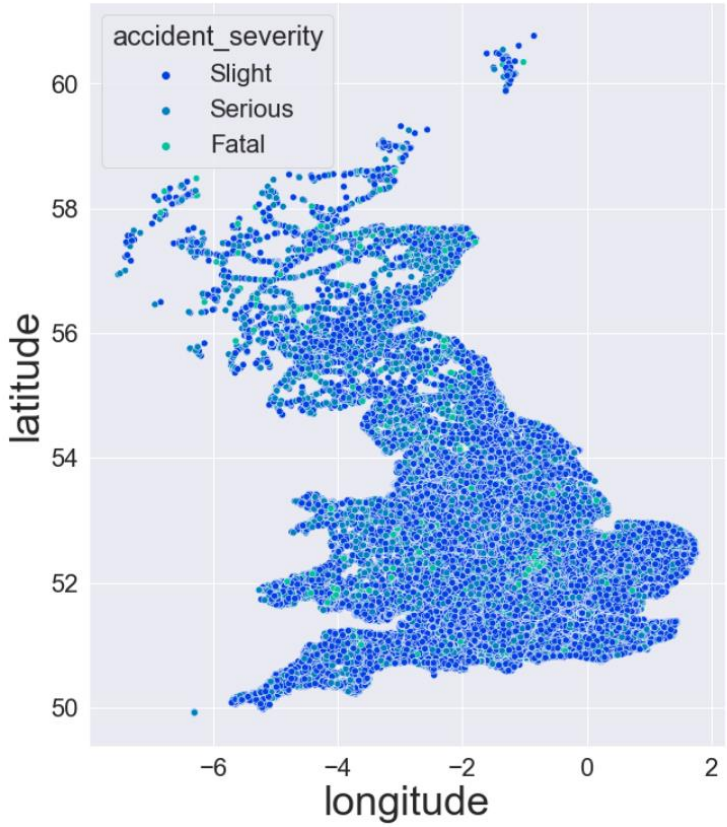


Figure 34: Accident severity with respect to the locations

Following map shows the accident according to the casualty area type, which is divided into 3 types, 1: Urban area, 2: Small town, 3: Rural area

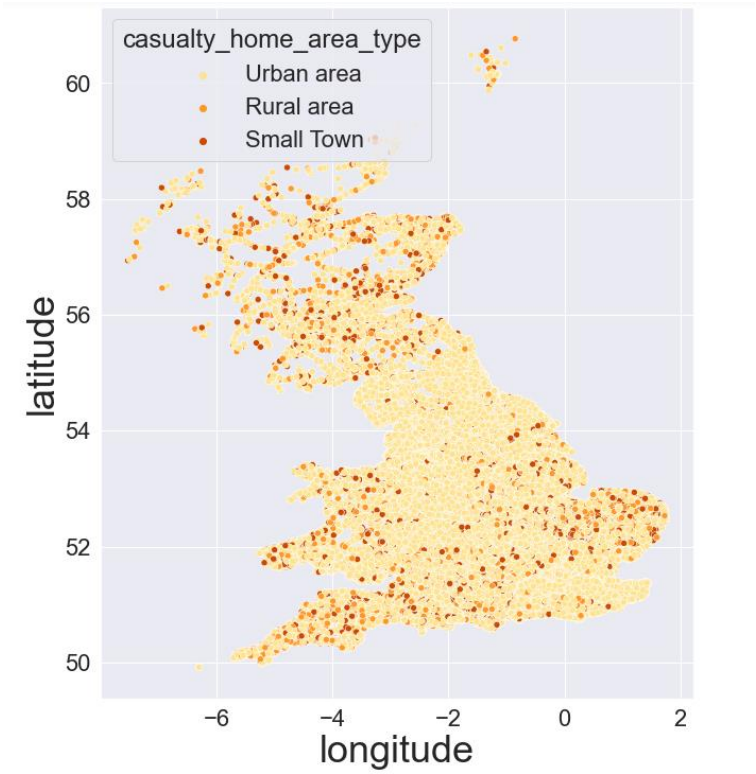


Figure 35: Accidents with respect to area type

In the following figure the relation of casualty type and the place of the accident is illustrated:

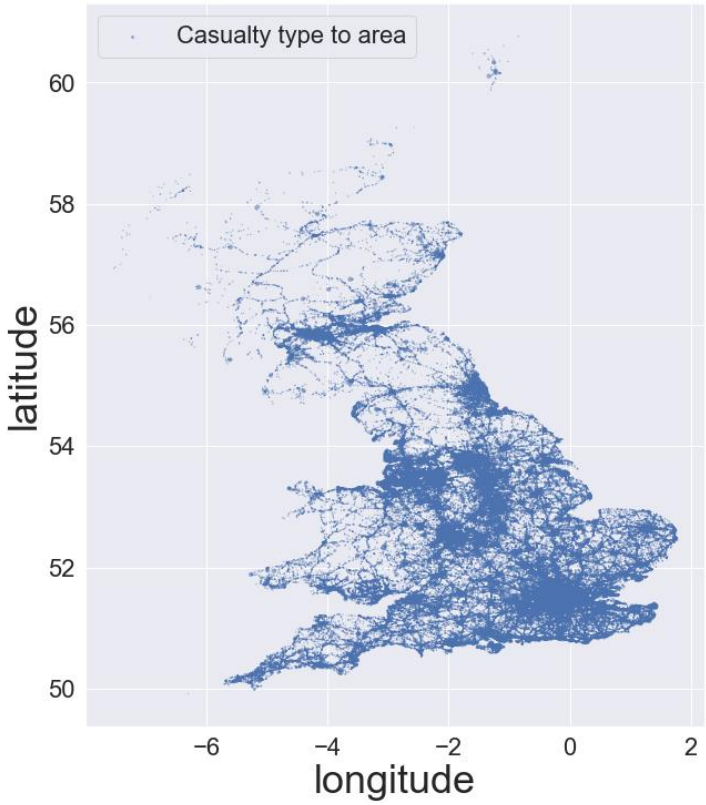


Figure 36: Area of accident to casualty type

People

Age says a lot about peoples driving behaviour. In the dataset there are 11 age bands in 'age band of casualty' column. Quantity of accidents to the age band is displayed in the following diagram:

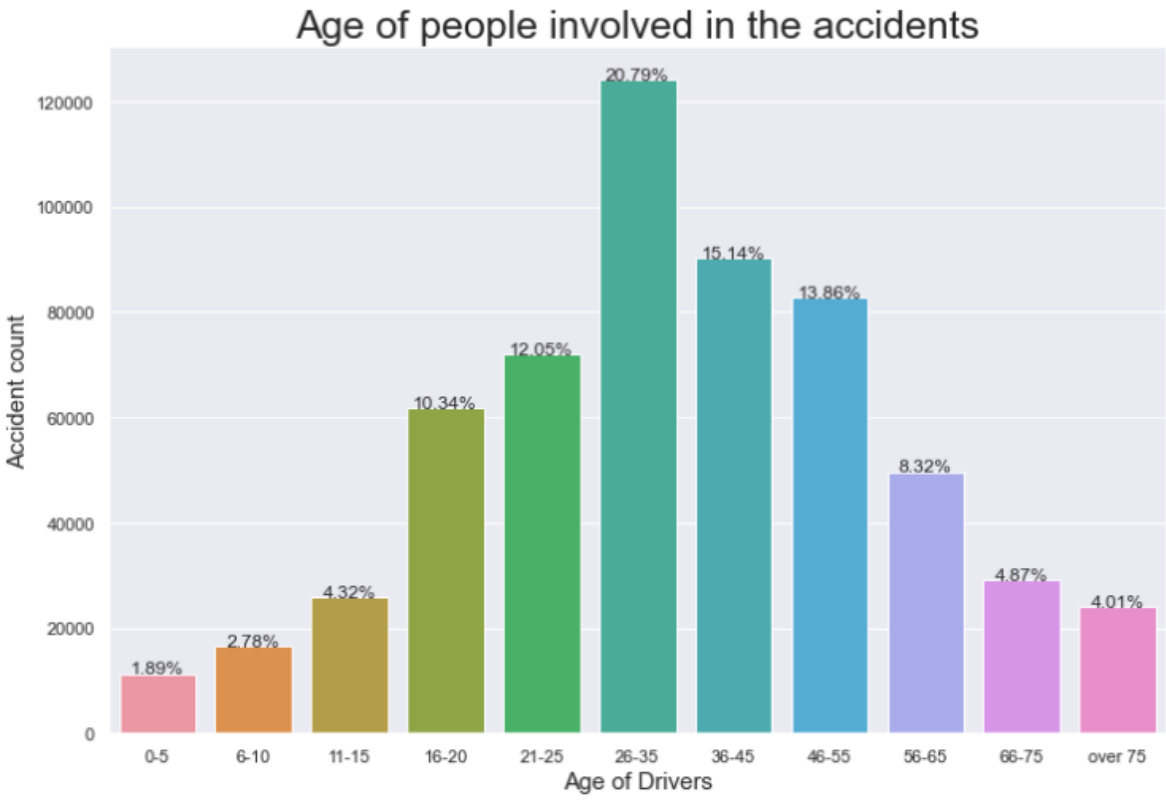


Figure 37: Age to the number of accidents

It can be discerned that people aged between 26-35 are mostly involved in the accidents around UK and as people got older the number of accidents is less. But there is no data to compare this to of how many drivers were in the road to begin with. If there are more people aged between 26 and 35 are on the road it is naturally mean more accidents for that age group.

Now let us have a look at the gender of drivers involved in the road accidents. As we can discern from the figure below, males are much more engaged in the traffic casualties. On the other hand, females contribute only around 40% whereas males are involved in 60% of the road accidents.

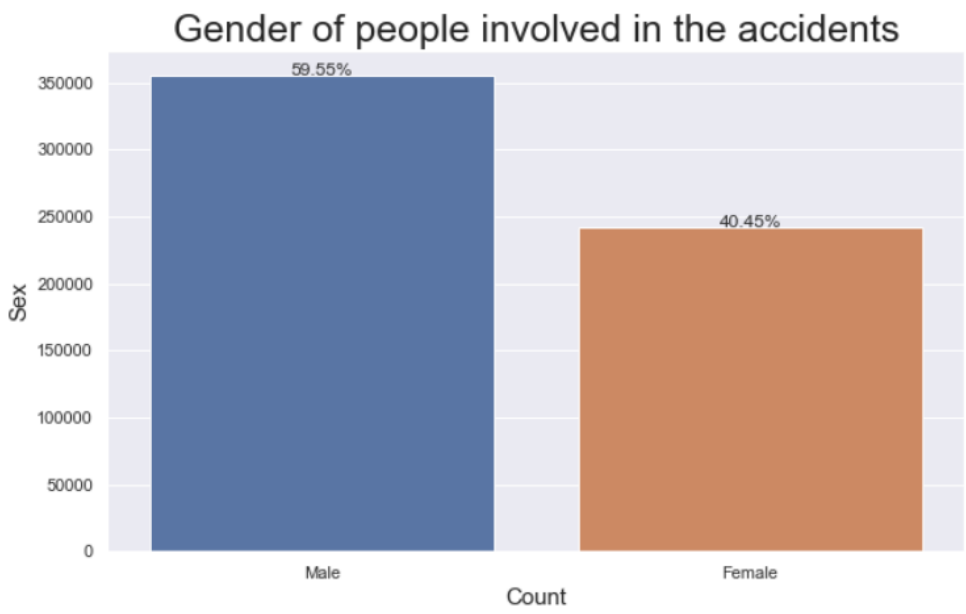


Figure 38: Sex of the drivers in accidents

Severity

There are 3 types of severity in the severity of the accident column. 1: Fatal, 2: Serious, and 3: Slight. The numbers are first mapped to their corresponding severity levels and then in the following figure it is illustrated through a bar plot.

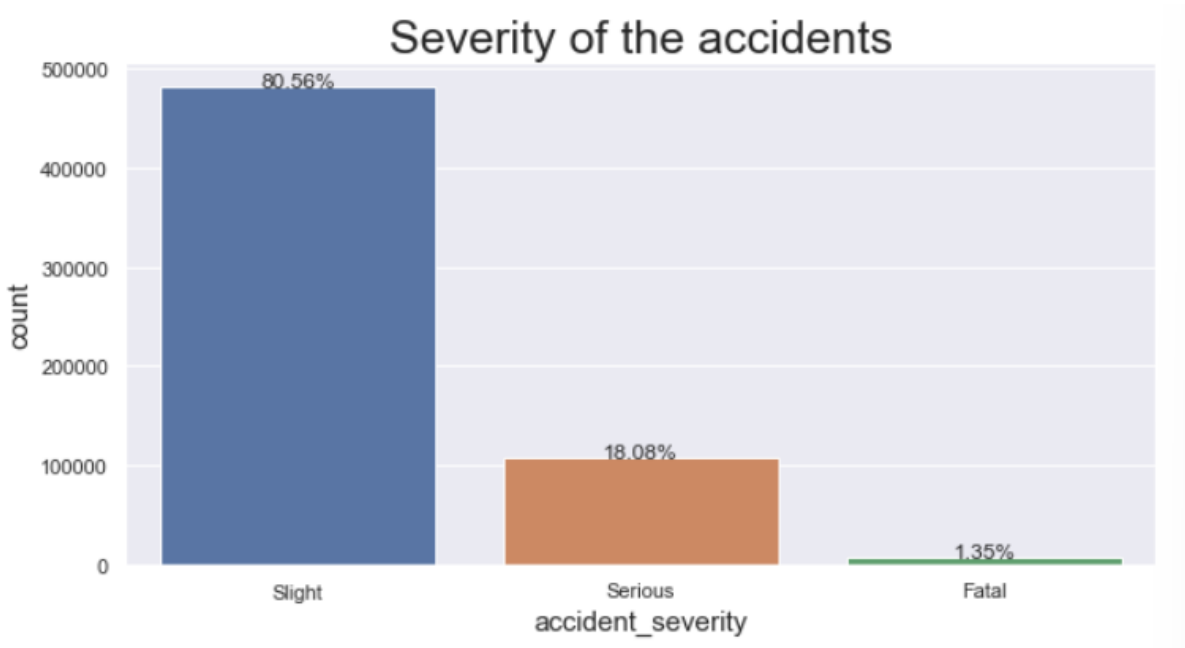


Figure 39: Accidents severity

In the following figure the data is shown according to the gender how severity varies:

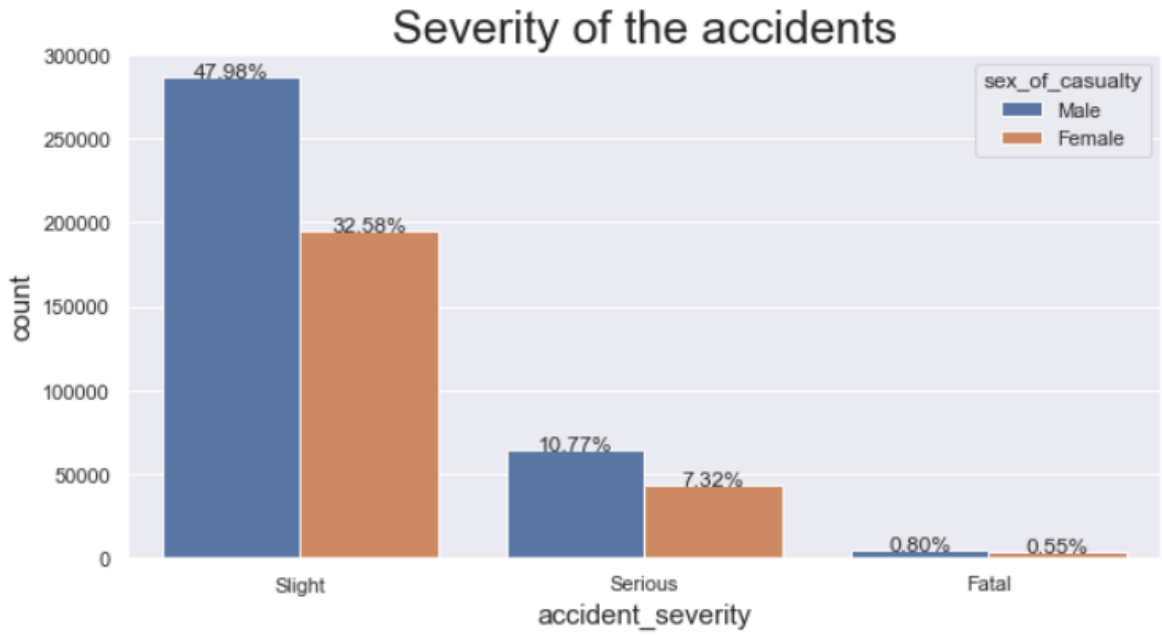


Figure 40: Severity of accidents with respect to gender

As males are more involved in the accidents in general, this trend can also be seen in here, as in all categories males were more involved in.

In the following figure the accidents severity is illustrated with respect to the type of roads:

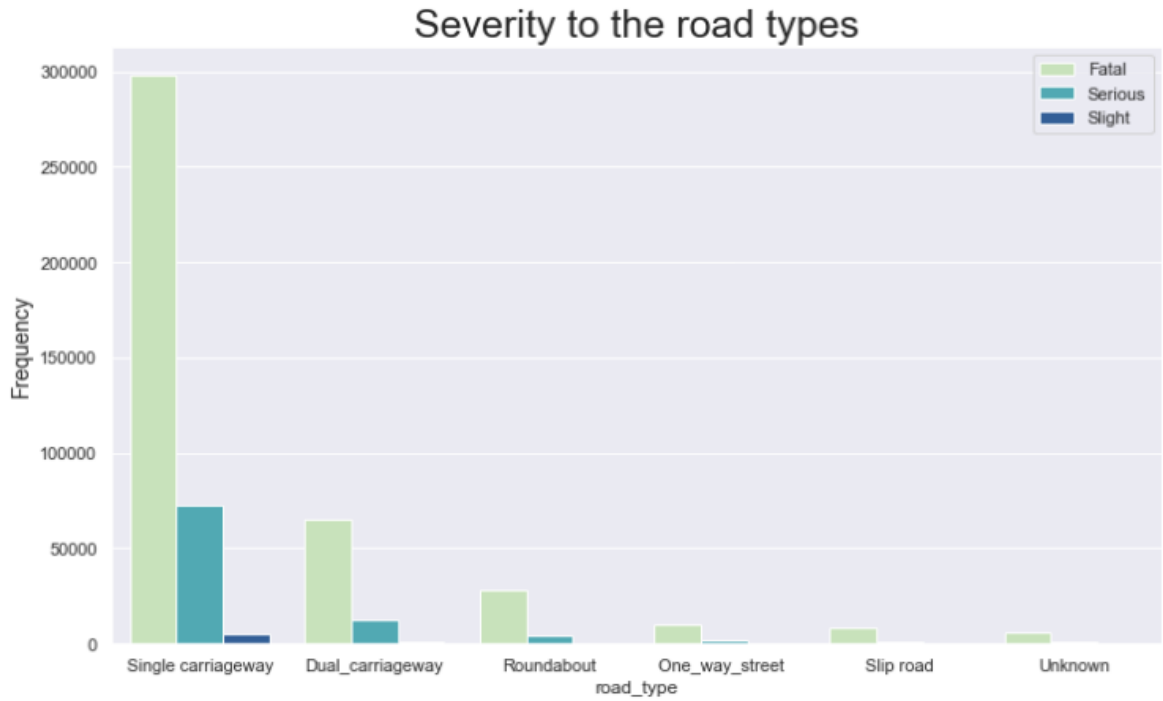


Figure 41: Accidents severity to the road type

Weather and road conditions

Weather surrounding also contributes to the road accidents. In the following figure we will have a look at the accident rates to the light conditions. Here in the dataset, there were 6 light conditions, here 1 is unknown. Those data are dropped, and then other numbers are mapped to their corresponding light conditions. These are illustrated in the following figure:

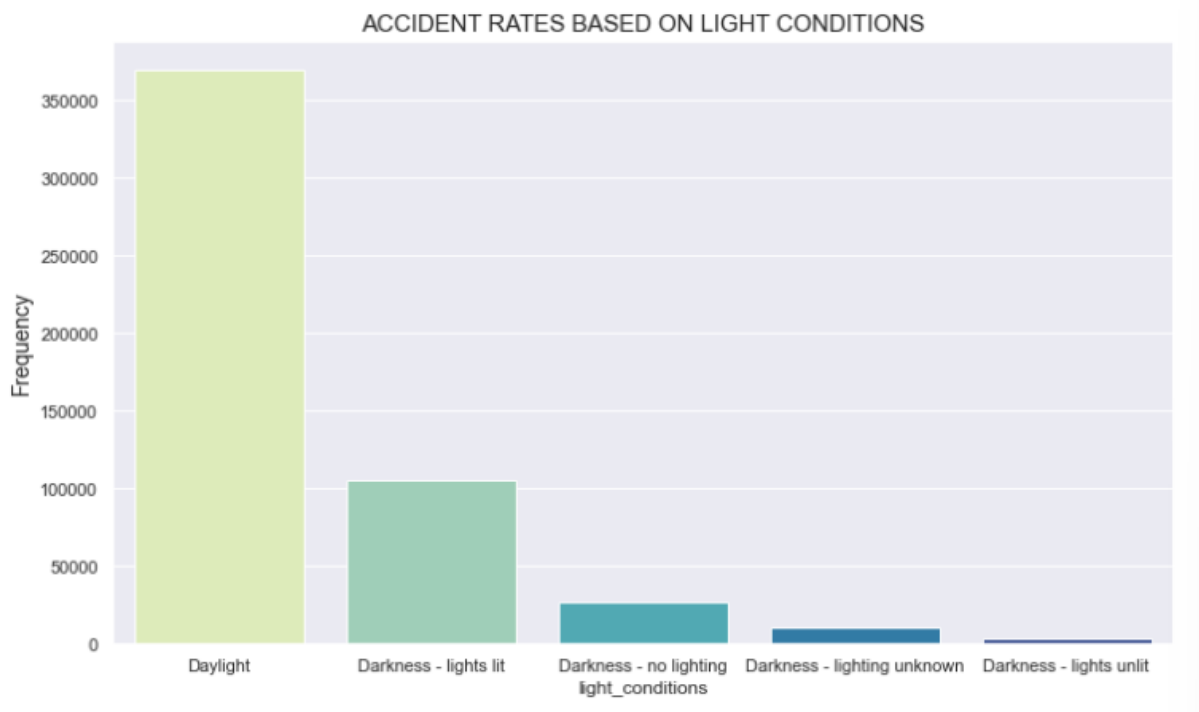


Figure 42: Accident rates to the light conditions

As it can be discerned from the graph most casualties occurred while in the daytime. Here least of the accident happened when its dark.

Now in the following figure let us have a look at the weather conditions effect on the road accidents. From the graph below we can observe that the weather conditions do not affect a lot in road conditions in UK, as most of the accidents mainly occur when the weather condition is just fine. Even though at rainy weather conditions there was more than 9% of all the accidents, but it is still far below the normal weather conditions 69%. Moreover, as UK see a lot of rainfall all year around, we can safely infer that the weather conditions does not affect the road accidents.

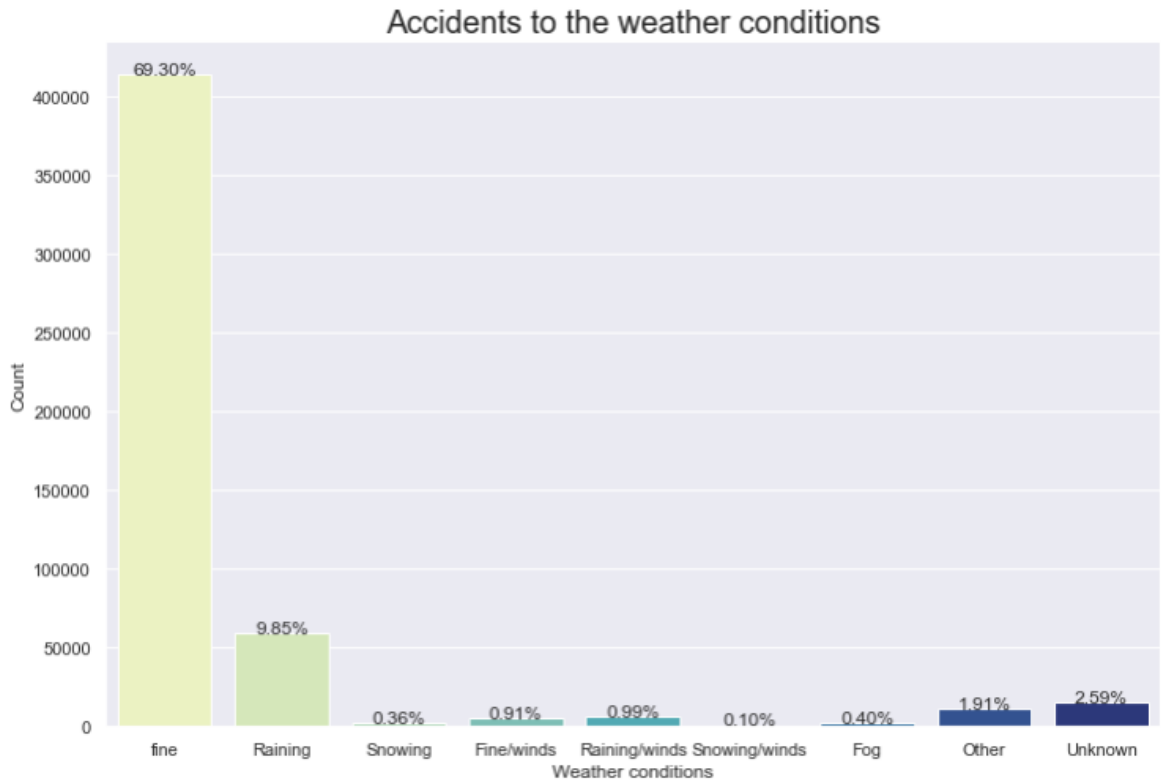


Figure 43: Number of accidents to the weather conditions

In the following figure we can observe the road surface conditions and accidents correlation. We can discern that wet road surface conditions mostly contributes to the road accidents, as more than 60% of all the accidents happened in UK between 2016 and 2021 was in wet road conditions. 2nd most risky road surface condition is when it snows. More than 22% of the accidents happened was in snow road surface conditions.

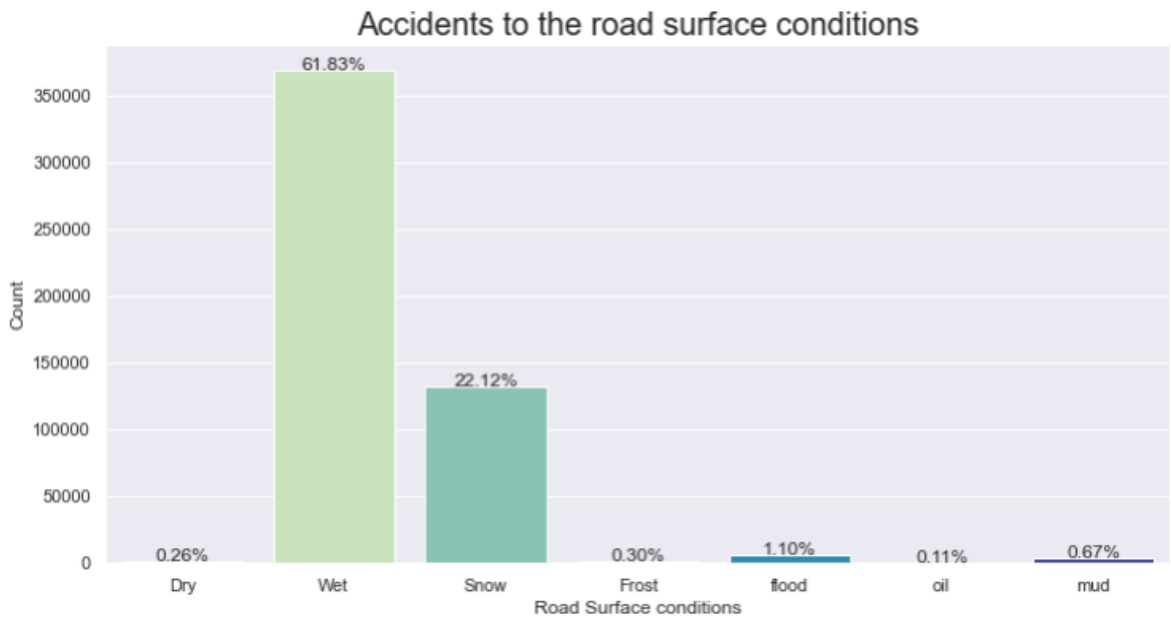
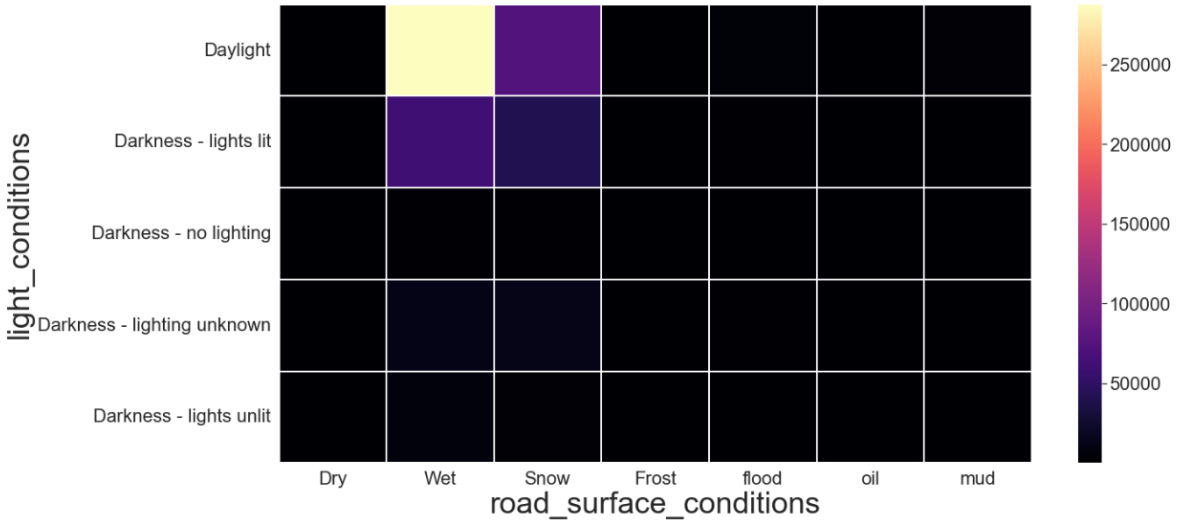


Figure 44: Accidents to the road surface conditions

The following figure shows the heatmap of light conditions to the road surface conditions.



Next the accidents dependence on the special road conditions are explored. In can be seen from the bar graph that, most of the accidents happened is not affected by special road conditions. Other than roadworks other special conditions are not even 1% mark.

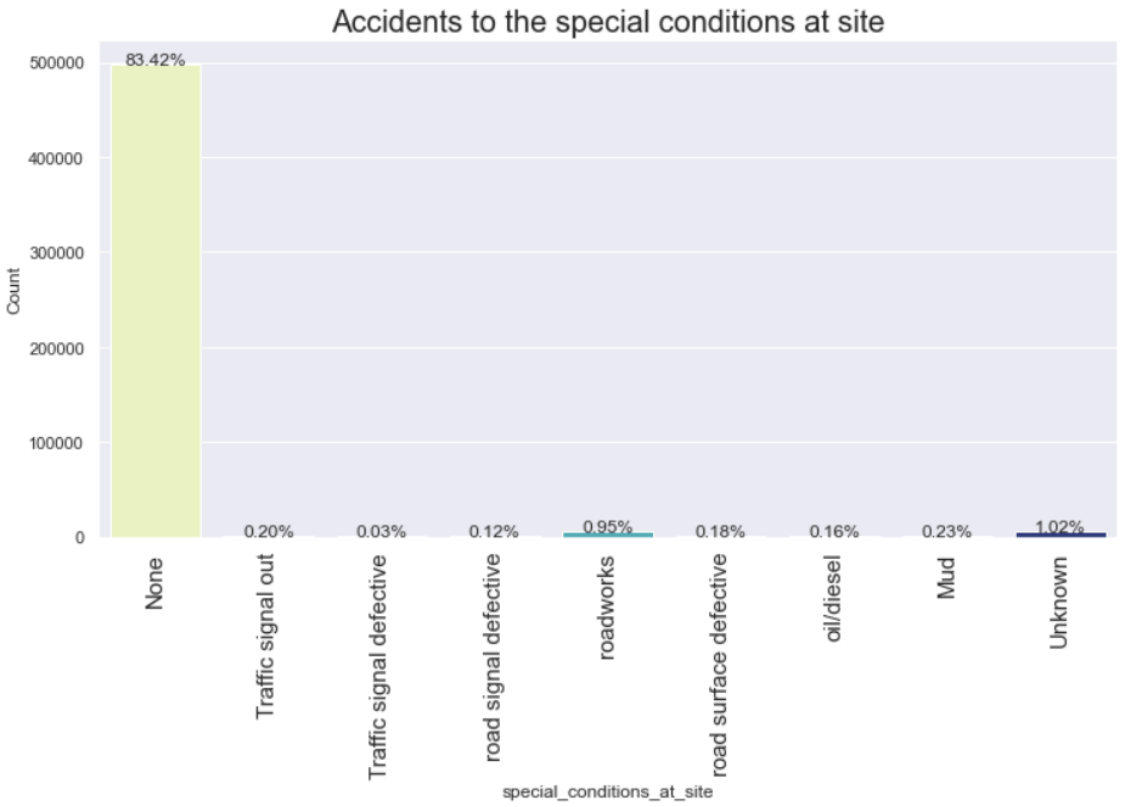


Figure 45: Special conditions at site to the accidents

Now the influence of carriageway hazards such as pedestrian, animal on the road accidents is explored. From the figure below it can be observed that the quantity is very low, and most of the accidents are not a result of carriageway hazards.

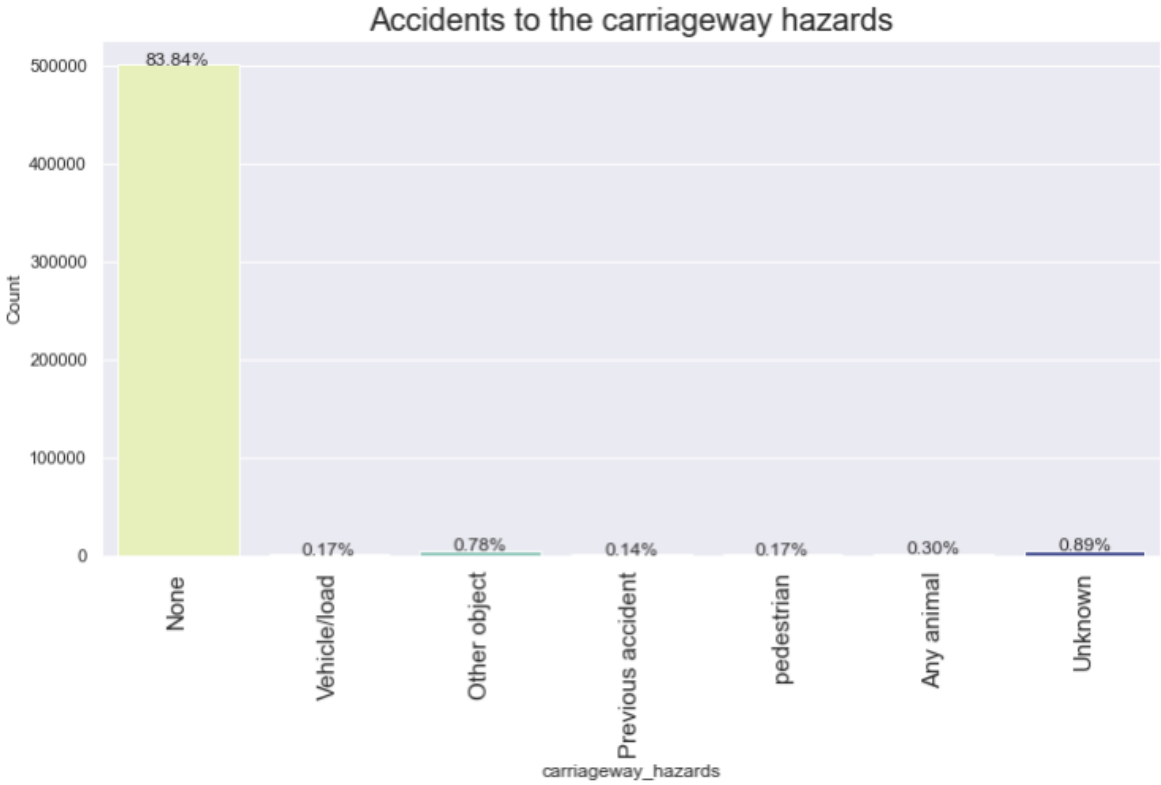


Figure 46: Carriageway hazards influence on the road accidents

4.2 Inferential Statistical Analysis

In this level of statistical analysis, the correlations and the inferential statistical tests are performed. Let us first have a look at the correlations between variables of road accidents through Pearson's correlation test:

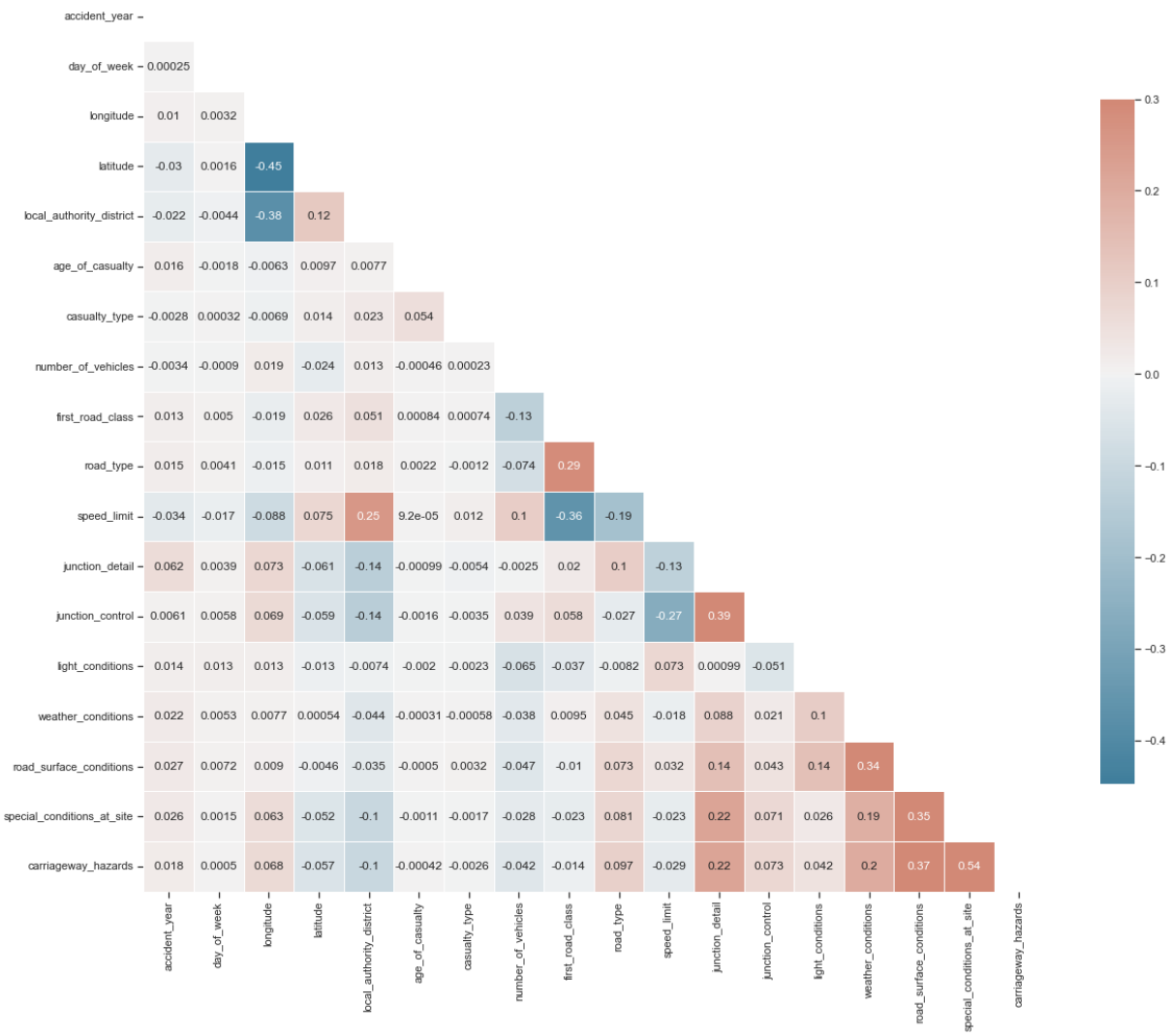


Figure 47: Pearson's correlation among all the features

4.2.1 Correlation

In this stage the correlation of accidents to the year 2016 is seen through Pearson correlation map. Here the strongest correlation is found with the junction control in 2016

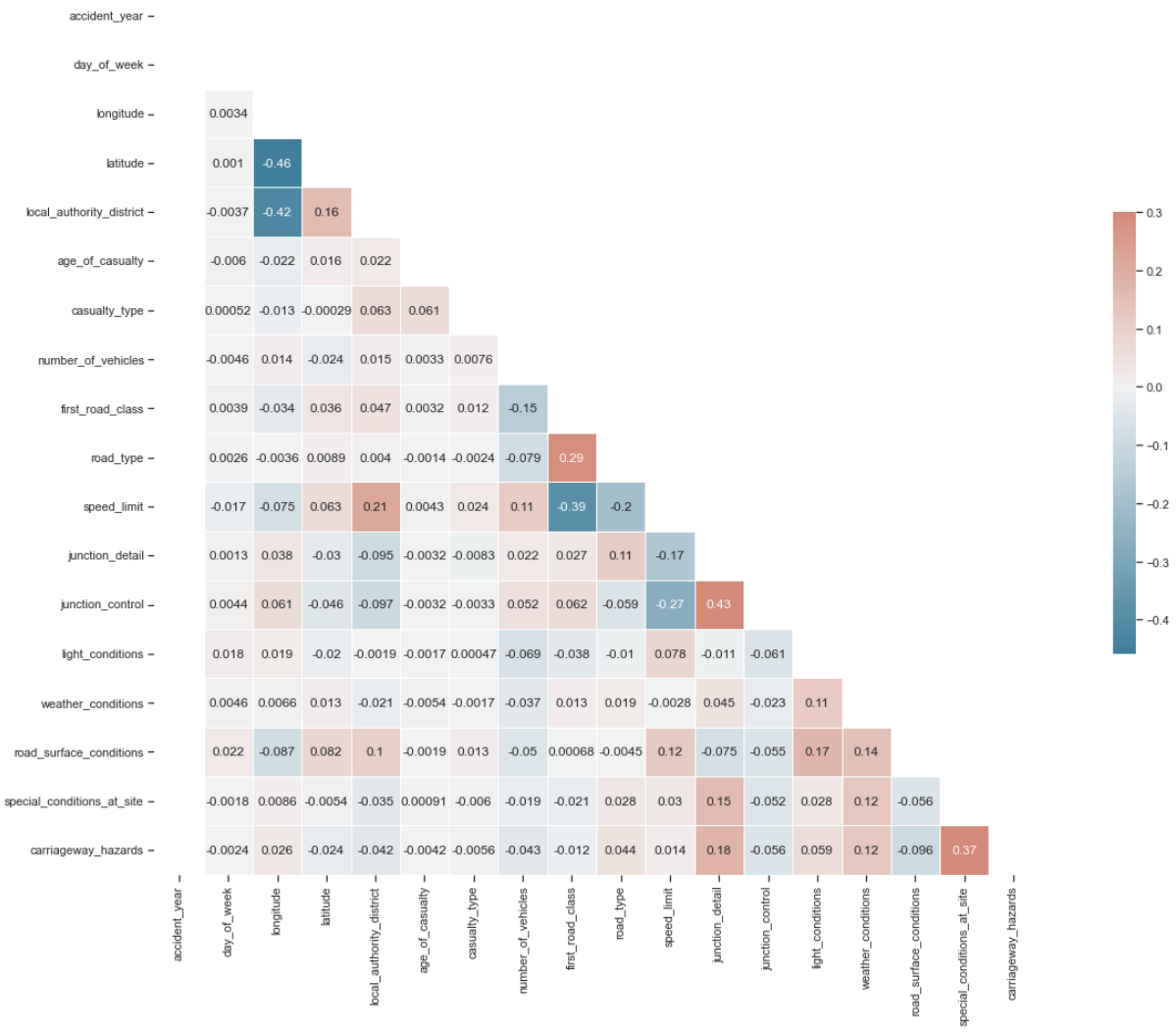


Figure 48: Correlation of all features in 2016

In this stage the correlation of accidents to the year 2017 is seen through Pearson correlation map. Here the strongest correlation is found with the carriageway hazards in 2017

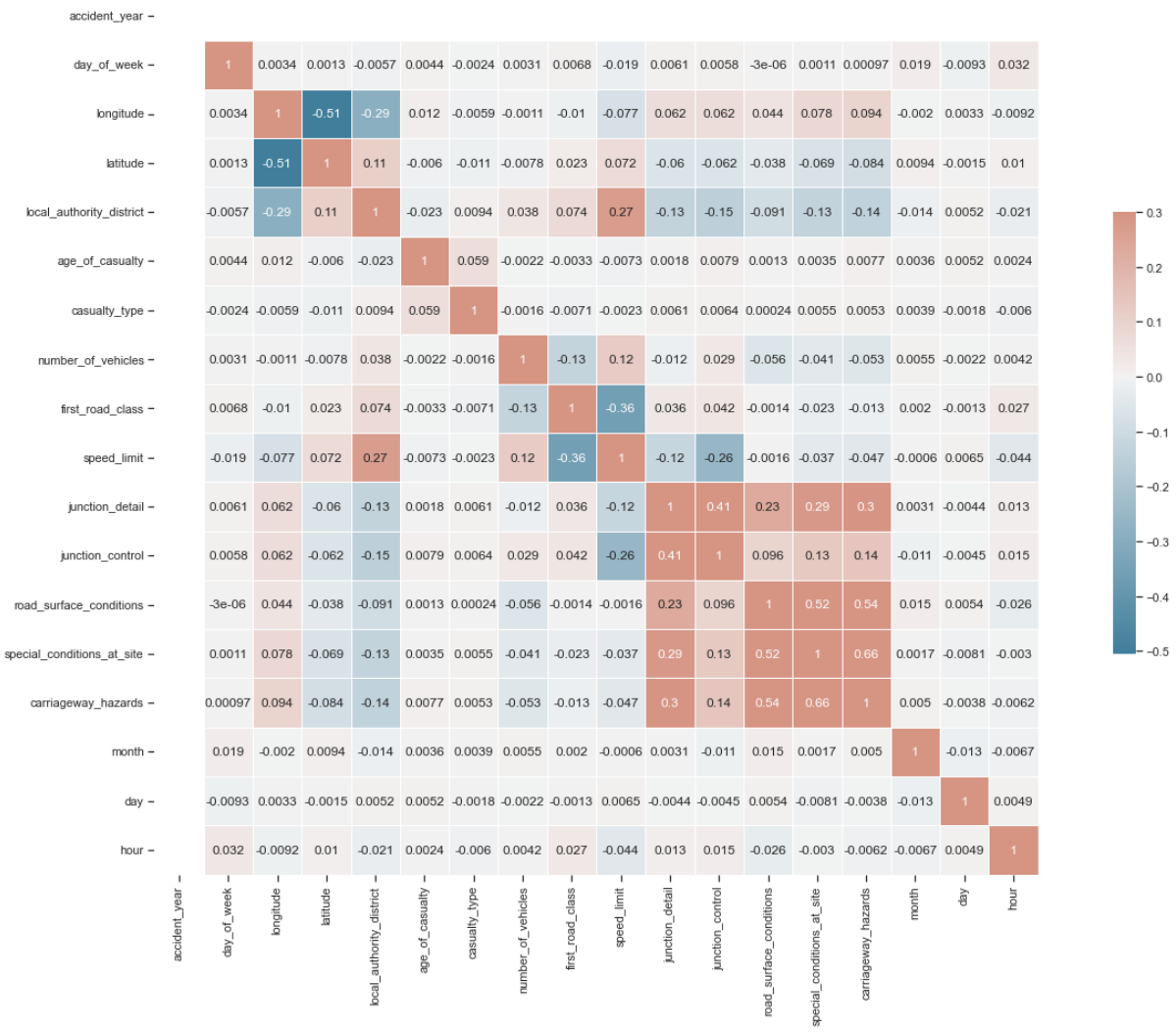


Figure 49: Correlation of all features in2017

In this stage the correlation of accidents to the year 2018 is seen through Pearson correlation map. Here the strongest correlation is found with the carriageway hazards in 2018

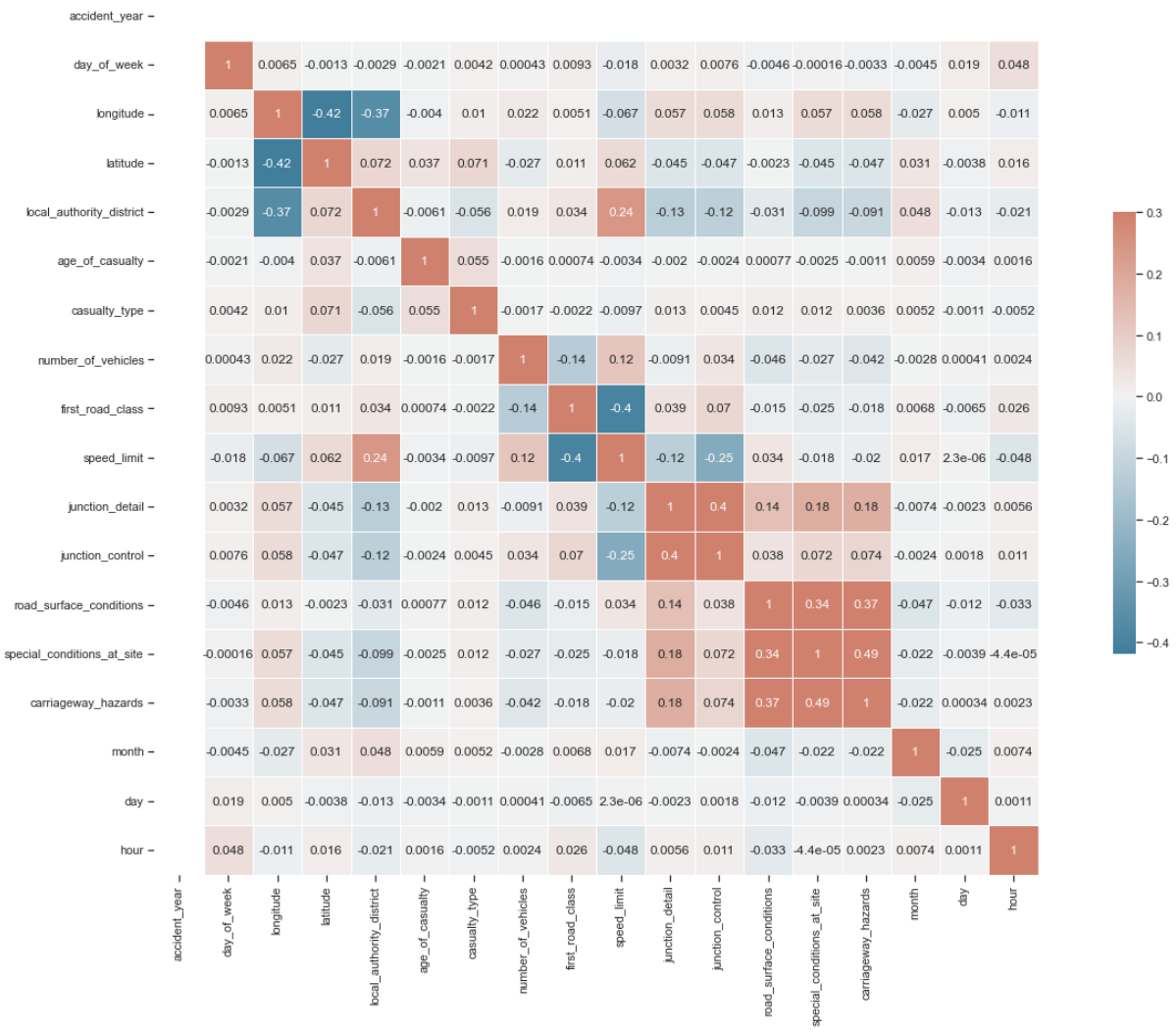


Figure 50: Correlation map with other columns in 2018 (without mask)

In this stage the correlation of accidents to the year 2019 is seen through Pearson correlation map. Here the strongest correlation is found with the carriageway hazards in 2019

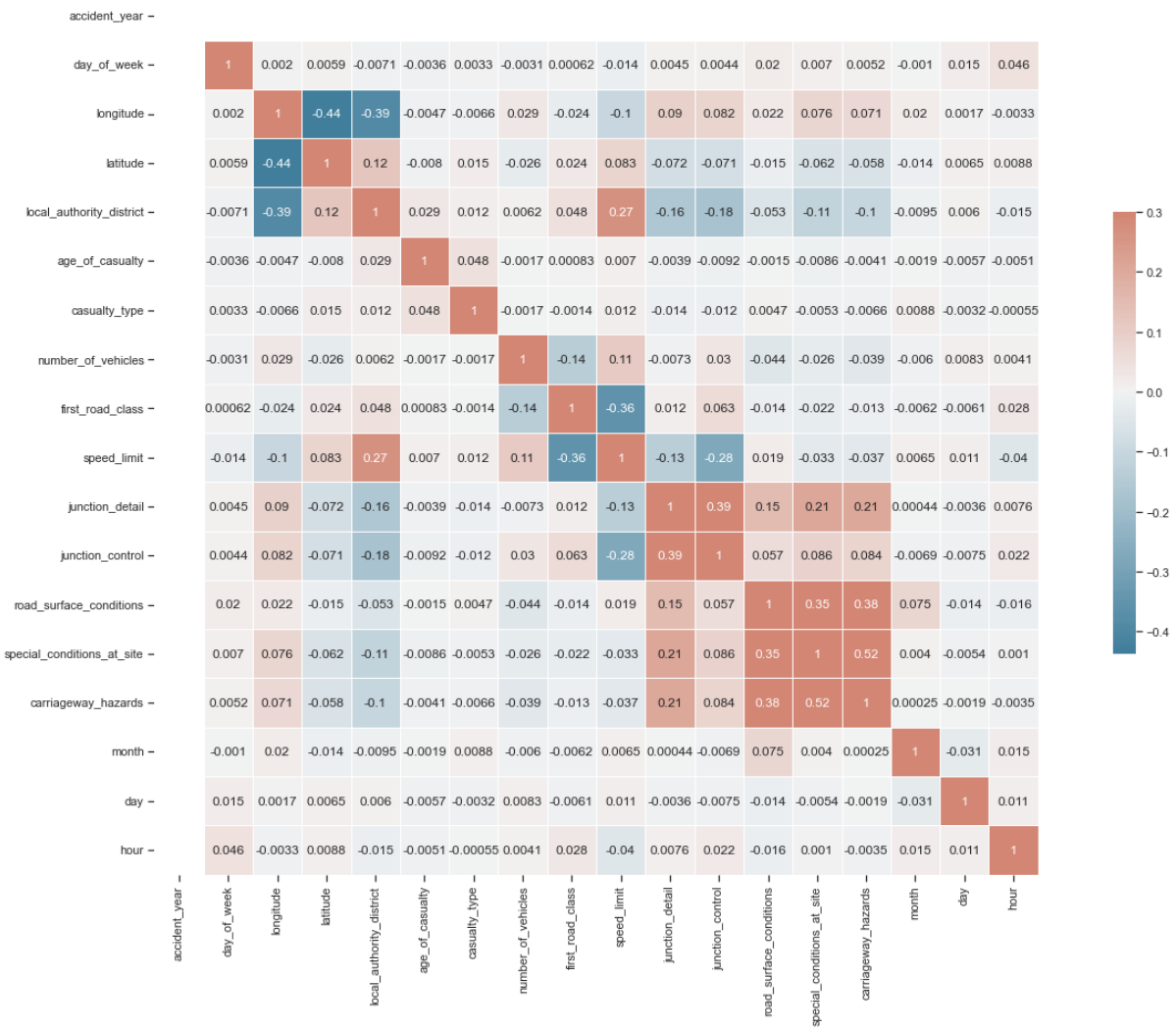


Figure 51: Correlation map with other columns (without mask)

In this stage the correlation of accidents to the year 2020 is seen through Pearson correlation map. Here the strongest correlation is found with the carriageway hazards in 2020

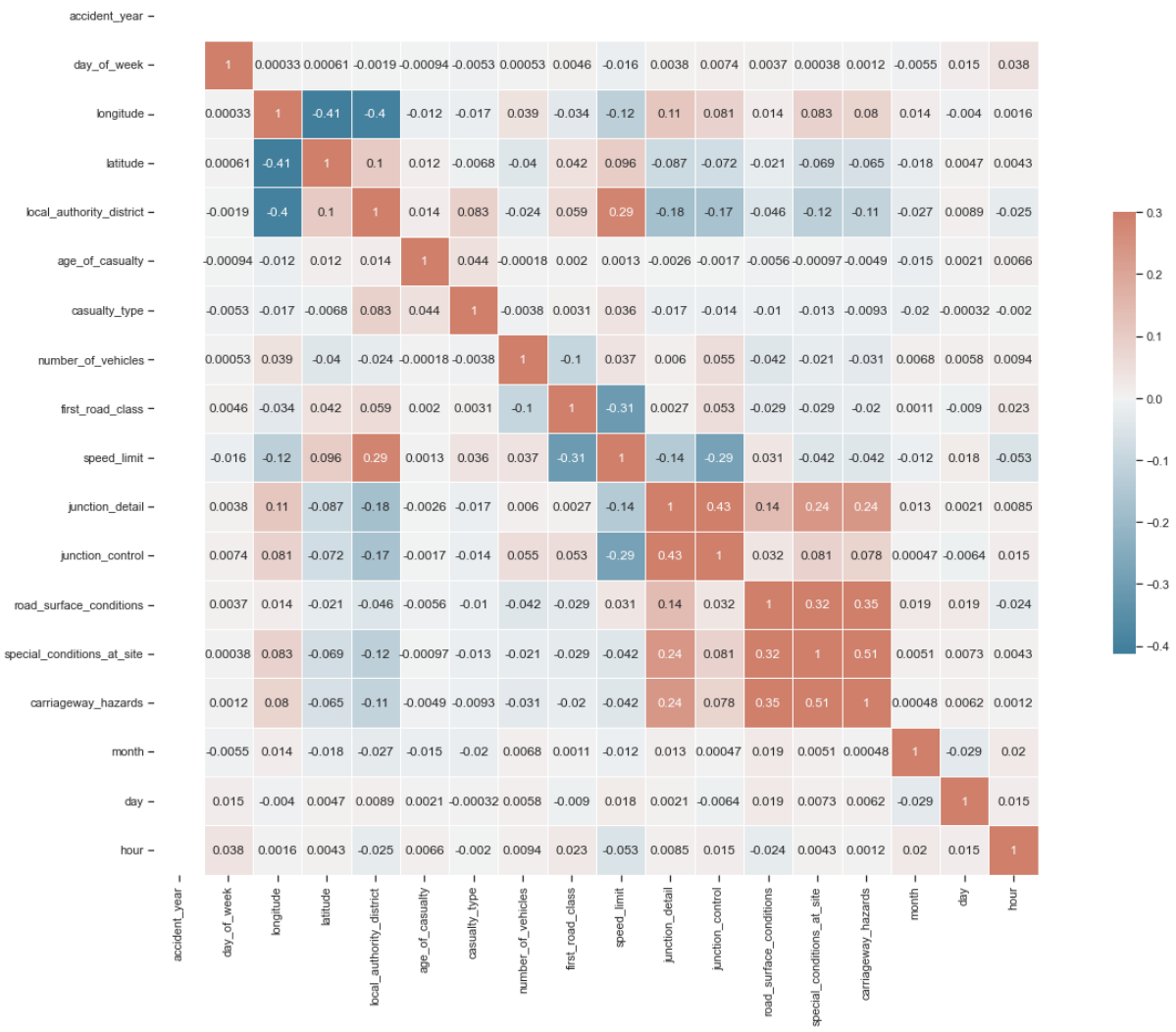


Figure 52: Correlation map with other features in 2020 (Without mask)

4.2.2 T-Test

At this step of the experiment T test is performed on the dataset for those columns that needs to be compared on 2 features, if there are more 2 features one way ANOVA is performed on that, which will be done on the next stage of the experiment. Here we are using 95% confidence level to test the statistical significance of the features, which means that the alpha value is 0.05.

Does sex of the driver affect the accident severity?

To test weather sex of the driver has an effect on the severity of accident or not 2 sample t-test is performed based on the following hypothesis:

H0: The male and female have same level of severity of accidents

H1: The male and female have different level of severity of accidents

Before doing the T-test lets first check weather variance of the two groups of data are equal or not. To check the equality of variance Levene's test is performed here. From the Levene's test we get the following results:

```
LeveneResult(statistic=0.7883350893860946, pvalue=0.3746044477196969)
```

As the p value is more than 0.37 so according to Levene's test they have homogeneity of variance. So, we can move ahead and perform the T-test. From the T-test we get the following results:

```
The value of P: 0.6400610385731413
The statistic value: -0.46761373523828986
The null hypothesis is accepted
```

As the p value is fewer than the 0.05, we cannot discard H0. So, we accept the null hypothesis that severity of the male vehicle operator is same as the female drivers.

Does location affect the severity of the accidents?

Here, at this step we would like to test if area has an effect on the accidents or not. 2 sample t-test is performed based on the following hypothesis:

H0 Null hypothesis: Urban and rural areas have the same levels of accident

H1 Alternative hypothesis: Accidents on urban areas are more prevalent than that of rural areas

From the test we get the following results:

```
The value of P: 0.006738855128995241
The statistic value: -2.709492034892519
The null hypothesis is rejected
```

As the p value is more than 0.05, we can discard the null hypothesis. So, we accept the alternate hypothesis that there is more accident in the city areas.

Are young people's more likely to be in severe accidents than old people?

In this step we want to test if age of the driver has an effect on the severity of accident or not. 2 sample t-test is performed based on the following hypothesis:

H0 Null hypothesis: The young and old drivers have same probability of accident

H1 Alternative hypothesis: The young and old drivers have different probability of accident

From the test we get the following results:

```
The value of P: 0.008774165212417218
The statistic value: -2.620740278507992
The null hypothesis is rejected
```

As the p value is more than 0.05, we can discard the original hypothesis. So, we accept H1 that older people and younger peoples have different driving habits and have different probability of being in an accident.

Do bad lighting conditions contribute to the road accidents severity?

In this step we want to test if lighting conditions has an effect on the severity of accident or not. 2 sample t-test is performed based on the following hypothesis:

H0 Null hypothesis: Accidents severity in day is same as accidents at night

H1 Alternative hypothesis: Accidents severity in daylight is different from night-time

From the test we get the following results:

```
The value of P: 0.008774165212417218
The statistic value: -2.620740278507992
The null hypothesis is rejected
```

As the p value is more than 0.05, we can discard reject the original hypothesis. So, we accept H1 hypothesis that accidents severity in daylight is different from night-time.

4.2.3 One-way ANOVA

At this step of the experiment a series of one-way ANOVA (Analysis of variance) is performed on the dataset to find if there are features that are significantly different. For this research the alpha value chosen is 0.05 (95% confidence level).

Is casualty type dependent on the day of the week?

To test if the casualty type of the accidents in different days of the week are statistically significant or not one way ANOVA is performed. The test is performed based on the following hypotheses:

H0: There is no significant difference of casualty type in different days of the week

H1: There is significant difference between at least one of them

After performing the test, the following results are found:

```
The value of P: 0.6281527164942428
The statistic value: 0.7265783613118897
The null hypothesis is accepted
```

So, we found a p value of 0.62 which is higher than 0.05. So, we cannot reject he original hypothesis and we can say that there is no significant difference in casualty type in different days of the week.

Is accident severity independent of year of the accident?

To test if the severity of the accidents in different years are statistically significant or not one-way ANOVA is performed. The test is performed based on the following hypotheses:

H0: There is no significant difference of severity in different years

H1: There is significant difference between at least one of them

After performing the test, the following results are found:

```
The value of P: 4.6907876277626996e-198
The statistic value: 230.45587881291263
The null hypothesis is rejected
```

As we can see the value of p is smaller than 0.05. So, we can discard he original hypothesis and we can say that the difference in severity of the accidents in different years is statistically significant.

Does the road class affect accident severity?

To test if the severity of the accidents in different road classes are statistically significant or not one way ANOVA is performed. The test is performed based on the following hypotheses:

H0 Null hypothesis: There is no difference of severity in different road classes

H1 Alternative hypothesis: There is difference between at least one of them

After performing the test, the following results are found:

```
The value of P: 3.5696354847308164e-76
The statistic value: 72.52425586501744
The null hypothesis is rejected
```

As we can see the value of p is smaller than 0.05. So, we can discard the orginal hypothesis and we can say that the difference in severity of the accidents in different road classes is statistically significant.

Does junction type affect the casualty type?

To test if the casualty type of the accidents in different road junction types are statistically significant or not one way ANOVA is performed. The test is performed based on the following hypotheses:

H0: There is no significant difference of casualty in different road junction types

H1: There is significant difference between at least one of them

After performing the test, the following results are found:

```
The value of P: 0.03482884494257662
The statistic value: 2.156142083387309
The null hypothesis is rejected
```

As we can see the value of p is smaller than 0.05. So, we can discard the orginal hypothesis and we can say that the difference in severity of the accidents in different road classes is statistically significant.

Does road surface condition affect severity of the accidents?

To test if the severity of the accidents in different road surface conditions are statistically significant or not one way ANOVA is performed. The test is performed based on the following hypotheses:

H0: There is no significant difference of severity in different road surface conditions

H1: There is significant difference between at least one of them

After performing the test, the following results are found:

```
The value of P: 1.2431654732826588e-95
The statistic value: 90.59292097244156
The null hypothesis is rejected
```

As we can see the value of p is smaller than 0.05. So, we can discard the original hypothesis and we can say that the difference in severity of the accidents in different road surface conditions is statistically significant.

4.2.4 Chi Square test of independence

Chi square test is performed mainly between the categorical features. So, at this stage of the experiment Chi-square test is performed to test the independence of one data to another. Here, we are using an alpha value of 0.05 to test the hypothesis.

Is there any association between the severity of the accidents and the sex of the driver?

To test if the severity of the accidents is independent from the sex of the driver or not Chi square test is performed. The test is performed based on the following hypotheses:

H0 Null hypothesis: Sex of the driver is independent of the severity of the accidents

H1 Alternative hypothesis: They are dependent on each other

To perform chi square test of independence first crosstab of the columns is calculated, which is later fed into the test to get results. Here the crosstab is:

accident_severity	Fatal	Serious	Slight
sex_of_casualty			
Female	168307	38023	2835
Male	246357	55464	4135

Now the crosstab data is used to perform the Chi-square test of independence. After performing the test, the following results are found:

```
The Chi square value is: 260.71702016732104
The p value is: 1.1973570627755645e-58
The value for degree of freedom: 1
Expected cell counts is: [[193.47474747 120.52525253]
 [355.52525253 221.47474747]]
```

As we can see the value of p is smaller than 0.05. So, we can discard the original hypothesis and we can say that the severity of the accidents is independent of the sex of the driver.

Is there any connection among the accident's severity and the type of accident home area?

To test if the severity of the accidents is independent from the casualty home area type or not Chi square test is performed. The test is performed based on the following hypotheses:

H0 Null hypothesis: Casualty home area type is independent of the severity of the accidents

H1 Alternative hypothesis: They are dependent on each other

To perform chi square test of independence first crosstab of the columns is calculated, which is later fed into the test to get results. Here the crosstab is:

accident_severity	Fatal	Serious	Slight
casualty_home_area_type			
Rural area	41915	9717	668
Small Town	34073	7921	570
Urban Area	338676	75849	5732

Now the crosstab data is used to perform the Chi-square test of independence. After performing the test, the following results are found:

```
The Chi square value is: 17.60692688482232
The p value is: 0.0014725970936065745
The value for degree of freedom: 4
Expected cell counts are: [[ 42100.64664419   9491.69243731    707.6609185 ]
 [ 34263.32550216   7724.74946275    575.92503509]
 [338300.02785365  76270.55809994   5686.41404641]]
```

As we can see the value of p is smaller than 0.05. So, we can discard the original hypothesis and we can say that the severity of the accidents is independent of the area of accident.

Is there any association amongst the accident's severity and the road type?

To test if the severity of the accidents is independent from the road type or not Chi square test is performed. The test is performed based on the following hypotheses:

H0 Null hypothesis: Road type is independent of the severity of the accidents

H1 Alternative hypothesis: They are dependent on each other

To perform chi square test of independence first crosstab of the columns is calculated, which is later fed into the test to get results. Here the crosstab is:

accident_severity	Fatal	Serious	Slight
road_type			
Duel carriageway	64770	12897	1382
One way street	10374	1911	74
Roundabout	27799	4201	116
Single carriageway	297728	72703	5295
Slip road	5872	916	74

Now the crosstab data is used to perform the Chi-square test of independence. After performing the test, the following results are found:

```
The Chi square value is: 1683.1758699305292
The p value is: 0.0
The value for degree of freedom: 8
Expected cell counts are: [[6.34974425e+04 1.44674514e+04 1.08410610e+03]
 [9.92757519e+03 2.26192908e+03 1.69495722e+02]
 [2.57977187e+04 5.87783109e+03 4.40450248e+02]
 [3.01808246e+05 6.87649136e+04 5.15284002e+03]
 [5.51201723e+03 1.25587486e+03 9.41079089e+01]]
```

As we can see the value of p is smaller than 0.05. So, we can discard the original hypothesis and we can say that the severity of the accidents is independent of the road type.

Is there any connection among the accident's severity and the weather conditions?

To test if the severity of the accidents is independent from the weather conditions or not Chi square test is performed. The test is performed based on the following hypotheses:

H0 Null hypothesis: weather conditions are independent of the severity of the accidents

H1 Alternative hypothesis: They are dependent on each other

To perform chi square test of independence first crosstab of the columns is calculated, which is later fed into the test to get results. Here the crosstab is:

accident_severity	Fatal	Serious	Slight
weather_conditions			
Fine	330943	76477	5726
Fine + high winds	4198	1107	122
Foggy	1819	462	80
Raining	47853	10198	686
Raining + high winds	4554	1227	116
Snowing	1802	330	25
Snowing + High winds	491	97	3

Now the crosstab data is used to perform the Chi-square test of independence. After performing the test, the following results are found:

```
The Chi square value is: 248.73666005702427
The p value is: 2.5093383165054407e-46
The value for degree of freedom: 12
Expected cell counts are: [[3.31368954e+05 7.60593532e+04 5.71769237e+03]
 [4.35279372e+03 9.99099857e+02 7.51064188e+01]
 [1.89366980e+03 4.34655383e+02 3.26748212e+01]
 [4.71107509e+04 1.08133644e+04 8.12884784e+02]
 [4.72976315e+03 1.08562592e+03 8.16109364e+01]
 [1.73004903e+03 3.97099391e+02 2.98515838e+01]
 [4.74018996e+02 1.08801919e+02 8.17908485e+00]]
```

As we can see the value of p is smaller than 0.05. So, we can discard the original hypothesis and we can say that the severity of the accidents is independent of the weather conditions.

Is there any relation among the accident's severity and the light conditions?

To test if the severity of the accidents is independent from the light conditions or not Chi square test is performed. The test is performed based on the following hypotheses:

H0 Null hypothesis: Light conditions are independent of the severity of the accidents

H1 Alternative hypothesis: They are dependent on each other

To perform chi square test of independence first crosstab of the columns is calculated, which is later fed into the test to get results. Here the crosstab is:

accident_severity light_conditions	Fatal	Serious	Slight
Darkness - lighting unknown	9000	1529	120
Darkness - lights lit	83902	19996	1426
Darkness - lights unlit	2897	736	77
Darkness - no lighting	18489	6615	1252
Daylight	300376	64611	4095

Now the crosstab data is used to perform the Chi-square test of independence. After performing the test, the following results are found:

```
The Chi square value is: 3764.268990542102
The p value is: 0.0
The value for degree of freedom: 8
Expected cell counts are: [[8.57227125e+03 1.93263925e+03 1.44089505e+02]
 [8.47841015e+04 1.91147804e+04 1.42511814e+03]
 [2.98648947e+03 6.73311261e+02 5.01992736e+01]
 [2.12161500e+04 4.78323223e+03 3.56617804e+02]
 [2.97104988e+05 6.69830369e+04 4.99397528e+03]]
```

As we can see the value of p is smaller than 0.05. So, we can discard the original hypothesis and we can say that the severity of the accidents is independent of the weather conditions.

Is there any connection among the accident's severity and the speed limit?

To test if the severity of the accidents is independent from the speed limits or not Chi square test is performed. The test is performed based on the following hypotheses:

H0 Null hypothesis: speed limits are independent of the severity of the accidents

H1 Alternative hypothesis: They are dependent on each other

To perform chi square test of independence first crosstab of the columns is calculated, which is later fed into the test to get results. Here the crosstab is:

accident_severity	Fatal	Serious	Slight
speed_limit			
-1.0	69	6	0
20.0	36105	6649	201
30.0	255624	52167	2338
40.0	34366	8379	690
50.0	16445	4190	550
60.0	46144	16639	2362
70.0	25911	5457	829

Now the crosstab data is used to perform the Chi-square test of independence. After performing the test, the following results are found:

```
The Chi square value is: 7697.734253513967
The p value is: 0.0
The value for degree of freedom: 12
Expected cell counts are: [[6.03737763e+01 1.36114136e+01 1.01481011e+00]
 [3.45780741e+04 7.79571030e+03 5.81215578e+02]
 [2.49648785e+05 5.62839213e+04 4.19629394e+03]
 [3.49644663e+04 7.88282335e+03 5.87710363e+02]
 [1.70535793e+04 3.84477064e+03 2.86650030e+02]
 [5.24406621e+04 1.18228739e+04 8.81464064e+02]
 [2.59180597e+04 5.84328913e+03 4.35651216e+02]]
```

As we can see the value of p is smaller than 0.05. So, we can discard the original hypothesis and we can say that the severity of the accidents is independent of the speed limits.

4.3 Machine Learning

In this part of the experiment, we will use different features of dataset to predict the severity of the accidents. This can help low enforcement officers and medical professionals to better prepare for these situations. Here, we will mainly run Logistic regression, Random Forest, Decision tree, and KNN to predict the severity of the accidents.

Before running those ML models, we first need to normalize the dataset, so that it does not negatively affect the ML algorithms. Here, age of the casualty column has values between 0 to

100 and it has the capability of skewing the performance of the machine learning algorithms. After normalizing the dataset, it is ready to perform machine learning models.

4.3.1 Decision tree

First algorithm we are going to implement is decision tree classifier. After splitting the dataset into training and testing part with 80:20 ratio we ran the algorithm. From Decision tree algorithm, we get the following results:

```
Accuracy 75.48
              precision    recall  f1-score   support

           1   0.035088  0.016688  0.022619      1558
           2   0.191719  0.083286  0.116125     21072
           3   0.808008  0.917350  0.859214     94108

    accuracy                       0.754776    116738
   macro avg   0.344938  0.339108  0.332653    116738
weighted avg   0.686448  0.754776  0.713916    116738
```

Here we can see we have obtained 75.48% accuracy using this algorithm. The value of weighted average precision, recall, and f-1 score is 0.68, 0.75, and 0.71. Actual and predicted results summary is given below:

Predicted	1	2	3	All
Actual				
1	26	166	1366	1558
2	170	1755	19147	21072
3	545	7233	86330	94108
All	741	9154	106843	116738

If we do some hyperparameters tuning on the previous model of decision tree. we get the following results:

```
Accuracy 80.57
              precision    recall  f1-score   support

           1   0.000000  0.000000  0.000000      1591
           2   0.481481  0.000613  0.001225     21206
           3   0.805745  0.999852  0.892365     94519

    accuracy                       0.805670    117316
   macro avg   0.429075  0.333488  0.297863    117316
weighted avg   0.736204  0.805670  0.719181    117316
```

As we can see from the results the overall accuracy score increased from the previous experiment, previously the accuracy score was 75.48%, now it has jumped to 80.57%. Here, actual and predicted scores table look like:

Predicted	2	3	All
Actual			
1	0	1591	1591
2	13	21193	21206
3	14	94505	94519
All	27	117289	117316

4.3.2 Random Forest

The random forest is a classifying a data classification strategy that employs a large number of decision trees. While generating every unique tree, it utilizes bagging and features randomization to construct a statistically independent forest of trees whose committee forecast is much more precise than any individual tree. After running the random forest classifier, we get the following results:

```
Accuracy 78.15
              precision    recall  f1-score   support

           1   0.061372  0.010911  0.018529      1558
           2   0.198225  0.043470  0.071303     21072
           3   0.807403  0.959536  0.876920     94108

    accuracy                       0.781519    116738
   macro avg   0.355667  0.337972  0.322251    116738
weighted avg   0.687486  0.781519  0.720045    116738
```

Here, we have obtained 78.15% accuracy by running random forest classifier. The weighted average of the precision score is .68, recall is .78 and f1 score is .72. Actual and predicted results summary is given below:


```
Accuracy 78.15
              precision    recall  f1-score   support

           1   0.061372  0.010911  0.018529      1558
           2   0.198225  0.043470  0.071303     21072
           3   0.807403  0.959536  0.876920     94108

    accuracy                       0.781519    116738
   macro avg   0.355667  0.337972  0.322251    116738
weighted avg   0.687486  0.781519  0.720045    116738
```


4.3.3 Logistic Regression

Logistic regression is a technique for estimating the probabilities of a finite output given an input element. Even though the name suggests otherwise it is mainly a classification algorithm.

Most basic form of logistic regression is used to classify binary problems. But here in this problem we will be using multiclass classification of logistic regression.

After running logistic regression, we get the following results:

```
Accuracy 80.61
              precision    recall  f1-score   support

           1   0.000000  0.000000  0.000000      1558
           2   0.000000  0.000000  0.000000     21072
           3   0.806147  1.000000  0.892670     94108

    accuracy                       0.806147    116738
   macro avg   0.268716  0.333333  0.297557    116738
weighted avg   0.649873  0.806147  0.719624    116738
```

Here we can see we have obtained 80.61% accuracy using this algorithm. The value of weighted average precision, recall, and f-1 score is 0.64, 0.80, and 0.71. Actual and predicted results summary is given below:

Predicted	3	All
Actual		
1	1558	1558
2	21072	21072
3	94108	94108
All	116738	116738

In terms of numbers, we can observe that Logistic regression performed admirably. We can see the confusion matrix if we look closely. The Decision Tree Algorithm performed significantly better. As a true positive, it indicated a higher number of fatal and severe injuries. The accuracy score is lower in comparison to another method since the other system predicted the majority of minor accidents, which are extremely high in the dataset overall. In terms of looking at all different predictions of each class, the confusion matrix aids us in determining which method performed better.

If we do some hyperparameters tuning, we get the following results for logistic regression:

```
Accuracy 80.57
              precision    recall  f1-score   support

           1   0.000000  0.000000  0.000000      1591
           2   0.000000  0.000000  0.000000     21206
           3   0.805679  1.000000  0.892383     94519

    accuracy                       0.805679    117316
   macro avg   0.268560  0.333333  0.297461    117316
weighted avg   0.649118  0.805679  0.718974    117316
```

As we can see the overall accuracy score actually decreased from the previous iteration. But the table below of actual and predicted scores we can see that it still only classified just for class 3.

Predicted	3	All
Actual		
1	1591	1591
2	21206	21206
3	94519	94519
All	117316	117316

4.3.4 KNN

KNN is a very simple algorithm, it basically assumes anything close to a datapoint is of a single class. After running the K nearest neighbour classifier, we get the following results:

```
Accuracy 77.17
              precision    recall  f1-score   support

           1   0.051020  0.006285  0.011192      1591
           2   0.200181  0.062671  0.095457     21206
           3   0.807315  0.943652  0.870176     94519

    accuracy                       0.771694    117316
   macro avg   0.352839  0.337536  0.325608    117316
weighted avg   0.687313  0.771694  0.718488    117316
```

Here, we have obtained 78.15% accuracy by running random forest classifier. The weighted average of the precision score is .68, recall is .78 and f1 score is .72. Actual and predicted results summary is given below:

Predicted	1	2	3	All
Actual				
1	10	117	1464	1591
2	53	1329	19824	21206
3	133	5193	89193	94519
All	196	6639	110481	117316

Now, if we compare all the results of machine learning models used in this experiment, we find that Decision tree with hyperparameter tuning performed the best. Here, even though logistic regression's score is high it is not comparable to others as even after tuning hyperparameters the algorithm was just classifying one of the classes of severity.

4.4 Deep Learning

At this level of research deep learning algorithms are applied on the dataset to predict accident severity. Here we will mainly use simple Neural Net, Neural net with different optimizers, LSTM and GRU model to predict.

4.4.1 Neural Net

First, we will implement a simple neural net to test the performance of it. Here, before implementing it we need to a few pre-processing on the data. First of all, we need to define a standard scaler, then we need to apply it on the X train and test data. Then we need to convert the class vectors to the target multiclass severity matrices. After that the Deep neural net model is made. The architecture of the model is given below:

```
Model: "sequential_21"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_26 (Dense)            (None, 9)                 90

 dense_27 (Dense)            (None, 64)                640

 dense_28 (Dense)            (None, 3)                 195

=================================================================
Total params: 925
Trainable params: 925
Non-trainable params: 0
```

Figure 53: Architecture of the Deep Neural network

After that the model was compiled using ‘categorical crossentropy’ loss, ‘Adam’ optimizer, and ’accuracy’ matrices. Then the model is iterated for 100 epochs. Score evaluation of the model is given below:

```
4584/4584 [==============================] - 4s 937us/step - loss: 0.5356 - a
ccuracy: 0.8051
Names of the score metrics in model evaluation are:
['loss', 'accuracy']
The score values are: Loss (Categorical Cross Entropy) and Accuracy (%)
[0.5355865359306335, 0.8050597906112671]
```

Plot of model’s loss with each epoch is illustrated below:

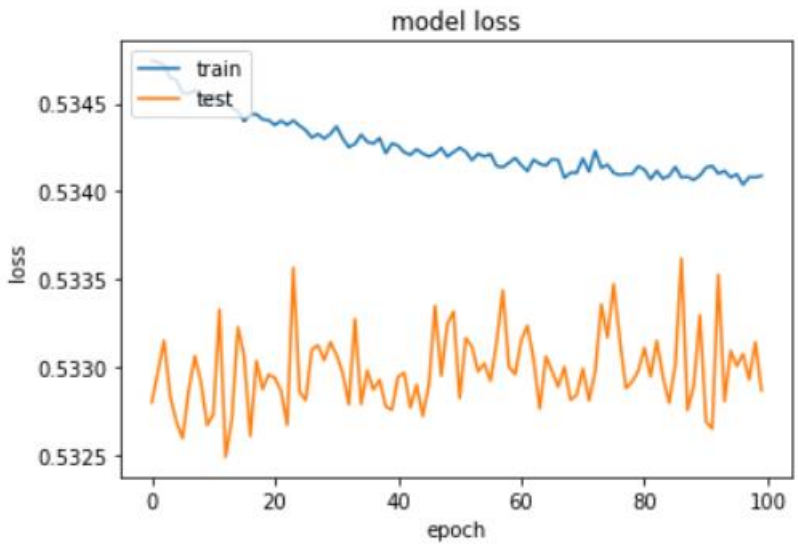


Figure 54: Loss with the epoch runs in Deep Neural Net

Here, as we can see, even the loss of the training data decreased over time, the loss of test data did not decrease well with time.

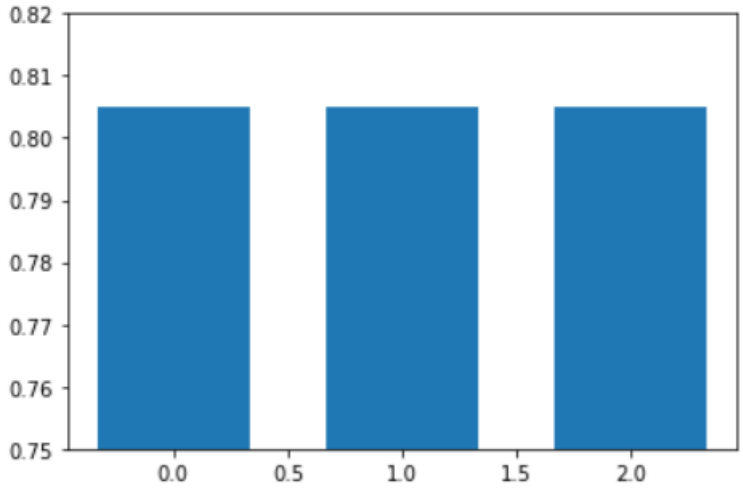
4.4.2 Neural Net with different optimizers

At this step of the experiment, another neural network was tested with different optimizers. Here, we primarily used SGD, RMSprop, and Adam optimizer. The architecture of the model used is given below:

```
 Layer (type)                 Output Shape              Param #
=================================================================
 dense_38 (Dense)             (None, 9)                 90

 activation_9 (Activation)    (None, 9)                 0

 dense_39 (Dense)             (None, 64)                640

 activation_10 (Activation)   (None, 64)                0

 dense_40 (Dense)             (None, 3)                 195

 activation_11 (Activation)   (None, 3)                 0

=================================================================
Total params: 925
Trainable params: 925
Non-trainable params: 0
```

Figure 55: Architecture of the NN to test performance of optimizers

Then the performance of the model is compared in the following figure:




```
Execution Time 1441.4669842720032 seconds:
```

Figure 56: Performance comparison of optimizers

Here, from the figure we can interpret that all the optimizers performance was same for this specific experiment.

4.4.3 LSTM

Now at this stage, LSTM is applied on the dataset. Here, Adam optimizer is used in the model. The loss function used in this case is 'mean_squared_error'. The model was iterated for 40 epochs with a batch size of 4, and the validation split used in this case is 0.20. The final accuracy of the model is given below:

```
Final Loss: 0.055057309567928314, Final Accuracy: 0.8399999737739563
```

The result of the model through RMSE is given below:

```
Trianing RMSE:  0.4870832400368943
Test RMSE:  0.46496676958919914
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

As we can see the training RMSE achieved in here is 0.487 and the score of tests RMSE is 0.464. Loss of the model with each epoch is illustrated in the following figure:

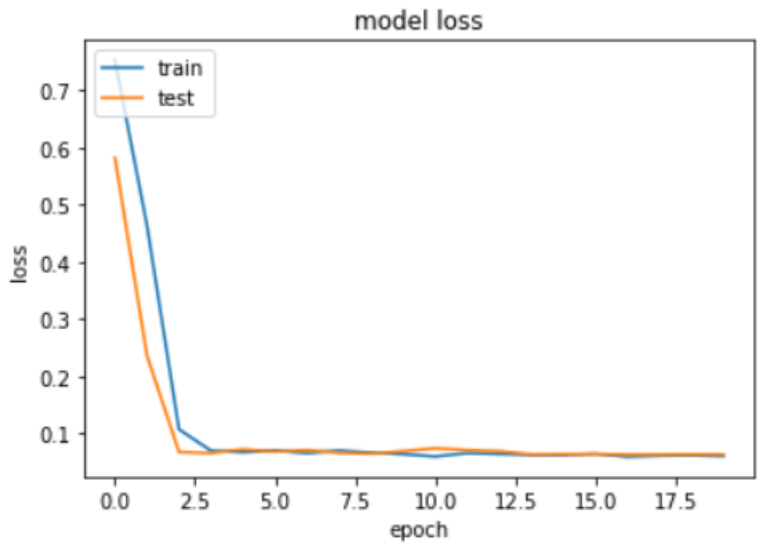


Figure 57: Model's loss with each epoch

4.4.4 GRU

At this stage GRU was used to estimate accidents severity. In the architecture of this model, we used the same Adam optimizer with a learning rate of 0.0001. Here, the loss function used is 'mean_squared_error'. The architecture of the model is given below:

```
Model: "sequential_20"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 gru (GRU)                   (None, 100)               30900

 dropout_8 (Dropout)         (None, 100)               0

 dense_25 (Dense)            (None, 1)                 101

=================================================================
Total params: 31,001
Trainable params: 31,001
Non-trainable params: 0
```

Figure 58: Architecture of GRU algorithm

The final accuracy score received in this model is 77.50%. Models' accuracy with each epoch is given below:

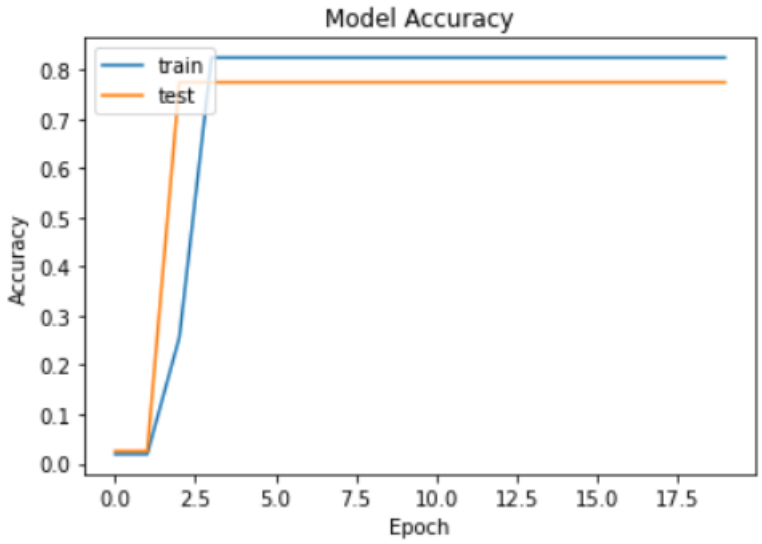


Figure 59: Accuracy with epoch of GRU

Model's loss with each epoch is illustrated below:

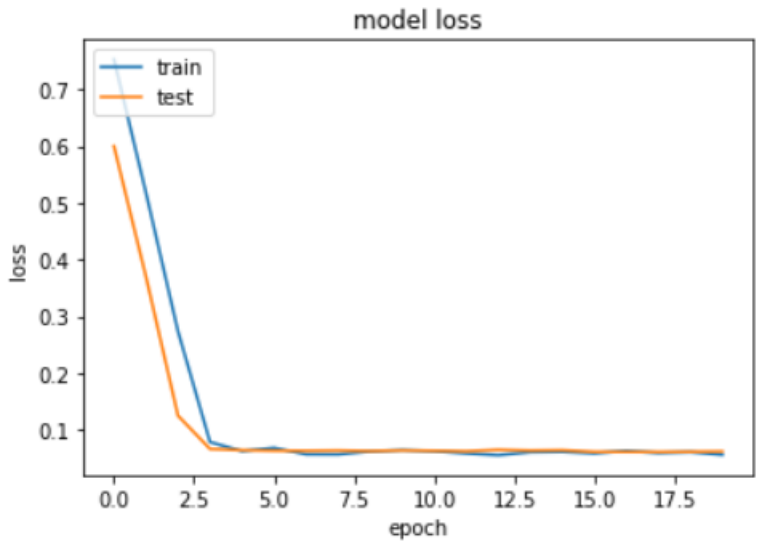


Figure 60: Models loss with each epoch for GRU

From the comparison of all the deep learning model's performance we see that LSTM performed the best among all these deep learning models. It is also observed there was no effect of using different optimizers in this particular experiment.

5. Conclusion and Recommendations

In this work we have done a thorough analysis of the road accidents data which is a combination of 2 different dataset collected from the UK govt site. In this experiment we have considered last 5 years data collected by the department of police of UK. Meticulous data pre-processing is done on the datasets, as the datasets are merged and trimmed that to better suit for this specific research. The pre-processing is done mainly in 4 steps, where data preparation is done by selecting, merging, cleaning and formatting data. Analysis and prediction of data is also done in 4 disparate stages. Firstly, descriptive analysis of the data is done to delineate the dataset and different visualisation tools and techniques is used to represent the data and make sense of different columns of the dataset. Secondly, different inferential statistical tools are used to make inferences on the dataset and to find the features that are mostly correlated to the accidents and accidents severity, which information is also essential for the third and fourth level of analysis. T-test, AVOVA (Analysis of variance) and Chi-square test of independence is done in addition to Pearson's correlation test to test various fulfil various research objectives that was set in the aims and objective section of the research. Here, using T-test we found that location and people's age a significant impact on the accidents, using one way ANOVA we found that road class, junction type and road surface conditions was also important, using Chi square test of independence we found that driver's gender, casualty home area type, road type, weather conditions was impacting the severity of the road accidents. The important features found at this stage is fed into the machine learning algorithms as features in the third stage of the experiment to predict the severity of the accidents. Here, in third level among all the machine learning algorithms that was tested Decision tree performed best. Even though simple logistic regression classifier had better accuracy score it was just classifying one of the three accidents severity class. In forth and final level of the analysis we constructed 4 neural network model to predict the severity of the accidents through the use of strongly correlated features of the accident. Of the 4 neural network models first one was a simple neural net with a Dense hidden layer, and input-output layer. This model achieved a final accuracy score of 80.51%. The second model was mainly designed to test the performances of deep learning optimizers. Here, we found that all the optimizers were having the same performance score. Thirdly a LSTM model was developed which scored a final accuracy score of 84%, which was the best among all the machine learning and deep learning modes tested in this experiment. Fourthly, gated recurrent unit (GRU) neural network model was constructed and this one achieved an overall score of 77.5%.

Further analysis can be done on the data by implementing more machine learning and deep learning algorithms. Like the case of Decision tree further tuning of the hyperparameters of the models might result in better results. Numerous combinations of deep learning models and several deep learning pre trained models can also be tested to find best overall algorithm to predict accidents. One thing to notice here is that our target variable in machine learning and deep learning models was mainly accidents severity prediction. Other features can also be tested to predict the road accidents. We can take away a few things from this research, most of the accidents is occurring in the urban type areas, and age of the driver is an important contributor to the road accidents.

References

- Azevedo, A. and Santos, M.F., 2008. KDD, SEMMA and CRISP-DM: a parallel overview. IADS-DM.
- Åberg, L., Afram, G. and Nilsson, M., 2005, October. Perception of other drivers' errors and violations and easiness of error detection. In 18th ICTCT Workshop. Helsinki: International Cooperation on Theories and Concepts in Traffic Safety. Retrieved from http://www. ictct. org/dlObject. php.
- Amin, S., 2020. Backpropagation–Artificial Neural Network (BP-ANN): Understanding gender characteristics of older driver accidents in West Midlands of United Kingdom. Safety science, 122, p.104539.
- Abdalla, I.M., Raeside, R., Barker, D. and McGuigan, D.R., 1997. An investigation into the relationships between area social characteristics and road accident casualties. Accident Analysis & Prevention, 29(5), pp.583-593.
- Al-Balbissi, A.H., 2003. Role of gender in road accidents. Traffic injury prevention, 4(1), pp.64-73.
- Brachman, R.J. and Anand, T., 1994, July. The Process of Knowledge Discovery in Databases: A First Sketch. In KDD workshop (Vol. 3, pp. 1-12).
- Brake: the road safety charity, 2020, UK road death and casualty statistics, Available at https://www.brake.org.uk/get-involved/take-action/mybrake/knowledge-centre/uk-road-safety. (Accessed: 18th April 2022)
- Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C. and Wirth, R., 2000. CRISP-DM 1.0: Step-by-step data mining guide. SPSS inc, 9, p.13.
- Clarke, D.D., Ward, P., Bartle, C. and Truman, W., 2006. Young driver accidents in the UK: The influence of age, experience, and time of day. Accident Analysis & Prevention, 38(5), pp.871-878.
- Chen, X., Huang, L., Dai, D., Zhu, M. and Jin, K., 2018. Hotspots of road traffic crashes in a redeveloping area of Shanghai. International journal of injury control and safety promotion, 25(3), pp.293-302.
- Casselgren, J. and Bodin, U., 2017. Reusable road condition information system for traffic safety and targeted maintenance. IET Intelligent Transport Systems, 11(4), pp.230-238.
- Department of Transport, 30 September 2020, Reported Road casualties in Great Britain: 2019 annual report. In Statistical release, p.1.
- Eboli, L., Mazzulla, G. and Pungillo, G., 2017. Measuring the driver's perception error in the traffic accident risk evaluation. IET intelligent transport systems, 11(10), pp.659-666.
- Fayyad, U., Piatetsky-Shapiro, G. and Smyth, P., 1996. From data mining to knowledge discovery in databases. AI magazine, 17(3), pp.37-37.
- Girotto, E., de Andrade, S.M., González, A.D. and Mesas, A.E., 2016. Professional experience and traffic accidents/near-miss accidents among truck drivers. Accident Analysis & Prevention, 95, pp.299-304.
- Hyodo, S., Yoshii, T., Satoshi, M. and Hirotoshi, S., 2017. An analysis of the impact of driving time on the driver's behavior using probe car data. Transportation research procedia, 21, pp.169-179.

- Huang, X., Zhang, M., Wang, J. and Chen, J., 2018, December. Study on Law of Scenario Evolution Based on Accident Statistics of Hazardous Chemicals Transportation. In 2018 IEEE International Conference of Safety Produce Informatization (IICSPI) (pp. 80-83). IEEE.
- Haddon Jr, W., 1980. Advances in the epidemiology of injuries as a basis for public policy. Public health reports, 95(5), p.411.
- Jian, T., Zhi-qiang, L., Jian-feng, X. and Hong-yu, G., 2019, July. Cause analysis and countermeasures of fatal traffic accidents on road passenger transportation based on typical cases. In 2019 5th International Conference on Transportation Information and Safety (ICTIS) (pp. 951-955). IEEE.
- Kumar, S. and Toshniwal, D., 2016. A data mining approach to characterize road accident locations. Journal of Modern Transportation, 24(1), pp.62-72.
- Kumar, S. and Toshniwal, D., 2015. A data mining framework to analyze road accident data. Journal of Big Data, 2(1), pp.1-18.
- Li, L., Shrestha, S. and Hu, G., 2017, June. Analysis of road traffic fatal accidents using data mining techniques. In 2017 IEEE 15th International Conference on Software Engineering Research, Management and Applications (SERA) (pp. 363-370). IEEE.
- Ljubič, P., Todorovski, L., Lavrač, N. and Bullas, J.C., 2002. Time-series analysis of UK traffic accident data. In Proceedings of the Fifth International Multi-Conference Information Society (pp. 131-134).
- Ming, H. and Xiucheng, G., 2009, October. The application of bp neural network principal component analysis in the forecasting the road traffic accident. In 2009 Second International Conference on Intelligent Computation Technology and Automation (Vol. 1, pp. 107-111). IEEE.
- National Statistics, 30 September 2021, Reported Road casualties Great Britain, annual report: 2020.
- National statistics, 30 September 2020, The impact of lockdown on reported road casualties Great Britain, final results: 2020.
- National statistics, 25 November 2021, Reported Road casualties in Great Britain, provisional estimates: year ending June 2021.
- Nowakowska, M., 2012. Road Traffic Accident Patterns: A Conceptual Grouping Approach to Evaluate Crash Clusters Wzorce wypadków drogowych: grupowanie pojęciowe w szacowaniu skupień wypadkowych. Archives of Transport, (1).
- Nandurge, P.A. and Dharwadkar, N.V., 2017, February. Analyzing road accident data using machine learning paradigms. In 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC) (pp. 604-610). IEEE.
- Ossenbruggen, P.J., Pendharkar, J. and Ivan, J., 2001. Roadway safety in rural and small urbanized areas. Accident Analysis & Prevention, 33(4), pp.485-498.
- Przepiorka, A.M., Hill, T., Blachnio, A.P., Sullman, M.J., Taylor, J.E. and Mamcarz, P., 2020. Psychometric properties of the Driving Behaviour Scale (DBS) among polish drivers. Transportation research part F: traffic psychology and behaviour, 73, pp.29-37.
- Petridou, E. and Moustaki, M., 2000. Human factors in the causation of road traffic crashes. European journal of epidemiology, 16(9), pp.819-826.
- Poulter, D.R. and Wann, J.P., 2013. Errors in motion processing amongst older drivers may increase accident risk. Accident Analysis & Prevention, 57, pp.150-156.

- Qiao, W. and Shang, B., 2011, June. Current situation analysis and safety countermeasure study on road traffic accidents in China. In 2011 International Conference on Remote Sensing, Environment and Transportation Engineering (pp. 3034-3037). IEEE.
- Reason, J., Manstead, A., Stradling, S., Baxter, J. and Campbell, K., 1990. Errors and violations on the roads: a real distinction?. Ergonomics, 33(10-11), pp.1315-1332.
- Reeja, S. and Jayaraj, V.S., 2017, April. An embedded based approaches for accident analysis using event data recorder. In 2017 IEEE International Conference on Electrical, Instrumentation and Communication Engineering (ICEICE) (pp. 1-5). IEEE.
- Sikdar, P., Rabbani, A., Dhapekar, N.K. and Bhatt, D.G., 2017. Hypothesis testing of road traffic accident data in india. International Journal of Civil Engineering and Technology (IJCIET), 8(6), pp.430-435.
- Steg, L. and van Brussel, A., 2009. Accidents, aberrant behaviours, and speeding of young moped riders. Transportation research part F: traffic psychology and behaviour, 12(6), pp.503-511.
- Sagberg, F., 1999. Road accidents caused by drivers falling asleep. Accident Analysis & Prevention, 31(6), pp.639-649.
- Taylor, M.C., Lynam, D.A. and Baruya, A., 2000. The effects of drivers' speed on the frequency of road accidents (p. 56). Crowthorne: Transport Research Laboratory.
- Tement, S., Plohl, N., Horvat, M., Musil, B., Jakus, G. and Sodnik, J., 2020. Driving demands, stress reactivity and driving behavior: An interactional approach. Transportation research part F: traffic psychology and behaviour, 69, pp.80-90.
- Vergura, S. et al. (2009) 'Descriptive and Inferential Statistics for Supervising and Monitoring the Operation of PV Plants', IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, p. 4456. Available at: http://search.ebscohost.com.ezproxy.leedsbeckett.ac.uk/login.aspx?direct=true &db=edsbl&AN=RN260171196&site=eds-live&scope=site (Accessed: 3 August 2020).
- WHO, 2018, Global status report on road safety. Available at: https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries. (Accessed: 18th April 2022)
- Xi, J., Zhao, Z., Li, W. and Wang, Q., 2016. A traffic accident causation analysis method based on AHP-Apriori. Procedia engineering, 137, pp.680-687.
- Yousefzadeh-Chabok, S., Ranjbar-Taklimie, F., Malekpouri, R. and Razzaghi, A., 2016. A time series model for assessing the trend and forecasting the road traffic accident mortality. Archives of trauma research, 5(3).
- Zhang, Z.K., 2010, March. Research on the taxi traffic accident and violation identification model. In 2010 2nd International Asia Conference on Informatics in Control, Automation and Robotics (CAR 2010) (Vol. 2, pp. 533-536). IEEE.

Appendix

In [133]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.impute import KNNImputer
import seaborn as sns
from scipy import stats
```

Understanding datasets

In [134]:
```
df=pd.read_csv('Dataset/dft-road-casualty-statistics-accident-last-5-years.csv',
df.head()
```

Out[134]:

	accident_index	accident_year	accident_reference	location_easting_osgr	location_northing_osgr
0	2.02E+12	2016	10000005	519310.0	188730.0
1	2.02E+12	2016	10000006	551920.0	174560.0
2	2.02E+12	2016	10000008	505930.0	183850.0
3	2.02E+12	2016	10000016	527770.0	168930.0
4	2.02E+12	2016	10000018	510740.0	177230.0

5 rows × 36 columns

In [135]:
```
df.tail()
```

Out[135]:

	accident_index	accident_year	accident_reference	location_easting_osgr	location_northing_
597968	2.02E+12	2020	991027064	343034.0	731
597969	2.02E+12	2020	991029573	257963.0	658
597970	2.02E+12	2020	991030297	383664.0	810
597971	2.02E+12	2020	991030900	277161.0	674
597972	2.02E+12	2020	991032575	240402.0	681

5 rows × 36 columns

In [136]:
```
df
```

Out[136]:

	accident_index	accident_year	accident_reference	location_easting_osgr	location_northing_
0	2.02E+12	2016	10000005	519310.0	188
1	2.02E+12	2016	10000006	551920.0	174
2	2.02E+12	2016	10000008	505930.0	183
3	2.02E+12	2016	10000016	527770.0	168
4	2.02E+12	2016	10000018	510740.0	177
...	...	...	...	...	
597968	2.02E+12	2020	991027064	343034.0	731
597969	2.02E+12	2020	991029573	257963.0	658
597970	2.02E+12	2020	991030297	383664.0	810
597971	2.02E+12	2020	991030900	277161.0	674
597972	2.02E+12	2020	991032575	240402.0	681

597973 rows × 36 columns

In [137]:
```
df.columns
```

Out[137]:
```
Index(['accident_index', 'accident_year', 'accident_reference',
       'location_easting_osgr', 'location_northing_osgr', 'longitude',
       'latitude', 'police_force', 'accident_severity', 'number_of_vehicles',
       'number_of_casualties', 'date', 'day_of_week', 'time',
       'local_authority_district', 'local_authority_ons_district',
       'local_authority_highway', 'first_road_class', 'first_road_number',
       'road_type', 'speed_limit', 'junction_detail', 'junction_control',
       'second_road_class', 'second_road_number',
       'pedestrian_crossing human_control',
       'pedestrian_crossing_physical_facilities', 'light_conditions',
       'weather_conditions', 'road_surface_conditions',
       'special_conditions_at_site', 'carriageway_hazards',
       'urban_or_rural_area', 'did_police_officer_attend_scene_of_accident',
       'trunk_road_flag', 'lsoa_of_accident_location'],
      dtype='object')
```

In [138]:
```
df.describe().T
```

Out[138]:

	count	mean	std	min
accident_year	597973.0	2017.827266	1.381319	2016.000000
location_easting_osgr	597850.0	451703.358935	94929.997358	64084.000000
location_northing_osgr	597850.0	282043.804647	152864.644216	10235.000000
longitude	597840.0	-1.260225	1.392031	-7.525273
latitude	597840.0	52.425656	1.376865	49.912362
police_force	597973.0	28.574257	24.915142	1.000000
accident_severity	597973.0	2.792099	0.437881	1.000000
number_of_vehicles	597973.0	1.842209	0.709933	1.000000
number_of_casualties	597973.0	1.307276	0.756494	1.000000
day_of_week	597973.0	4.109935	1.924954	1.000000
local_authority_district	597973.0	328.621888	257.290799	-1.000000
first_road_class	597973.0	4.197810	1.463992	1.000000
first_road_number	597973.0	778.149082	1580.905318	0.000000
road_type	597973.0	5.196603	1.689462	-1.000000
speed_limit	597936.0	37.137533	14.027131	-1.000000
junction_detail	597973.0	2.991710	8.650574	-1.000000
junction_control	597973.0	1.736229	2.495472	-1.000000
second_road_class	597973.0	2.685998	3.158066	-1.000000
second_road_number	597973.0	219.168503	926.539753	-1.000000
pedestrian_crossing human_control	597973.0	0.204884	1.289745	-1.000000
pedestrian_crossing_physical_facilities	597973.0	1.044415	2.260054	-1.000000
light_conditions	597973.0	2.024188	1.714039	-1.000000
weather_conditions	597973.0	1.627798	1.791257	-1.000000
road_surface_conditions	597973.0	1.363968	0.886029	-1.000000
special_conditions_at_site	597973.0	0.195765	1.163435	-1.000000
carriageway_hazards	597973.0	0.151627	1.049526	-1.000000
urban_or_rural_area	597973.0	1.331642	0.471219	-1.000000
did_police_officer_attend_scene_of_accident	597973.0	1.316391	0.537858	-1.000000
trunk_road_flag	597973.0	1.669672	0.863693	-1.000000

In [139]:
```
df.isnull().sum()
```

Out[139]:
```
accident_index                                  0
accident_year                                   0
accident_reference                              0
location_easting_osgr                         123
location_northing_osgr                        123
longitude                                     133
latitude                                      133
police_force                                    0
accident_severity                               0
number_of_vehicles                              0
number_of_casualties                            0
date                                            0
day_of_week                                     0
time                                            0
local_authority_district                        0
local_authority_ons_district                    0
local_authority_highway                         0
first_road_class                                0
first_road_number                               0
road_type                                       0
speed_limit                                    37
junction_detail                                 0
junction_control                                0
second_road_class                               0
second_road_number                              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
trunk_road_flag                                 0
lsoa_of_accident_location                       0
dtype: int64
```

In [140]:
```
df2=pd.read_csv('Dataset/dft-road-casualty-statistics-casualty-last-5-years.csv',
df2
```

Out[140]:

	accident_index	accident_year	accident_reference	vehicle_reference	casualty_reference	ca
0	2016010000005	2016	010000005	2	1	
1	2016010000006	2016	010000006	1	1	
2	2016010000008	2016	010000008	1	1	
3	2016010000016	2016	010000016	1	1	
4	2016010000018	2016	010000018	2	1	
...	...	...	...	...	...	
781711	2020991027064	2020	991027064	2	1	
781712	2020991029573	2020	991029573	1	1	
781713	2020991030297	2020	991030297	2	1	
781714	2020991030900	2020	991030900	2	1	
781715	2020991032575	2020	991032575	1	1	

781716 rows × 18 columns

In [141]:
```
df2.head()
```

Out[141]:

	accident_index	accident_year	accident_reference	vehicle_reference	casualty_reference	casualty
0	2016010000005	2016	010000005	2	1	
1	2016010000006	2016	010000006	1	1	
2	2016010000008	2016	010000008	1	1	
3	2016010000016	2016	010000016	1	1	
4	2016010000018	2016	010000018	2	1	

In [142]:
```
df2.tail()
```

Out[142]:

	accident_index	accident_year	accident_reference	vehicle_reference	casualty_reference	ca
781711	2020991027064	2020	991027064	2	1	
781712	2020991029573	2020	991029573	1	1	
781713	2020991030297	2020	991030297	2	1	
781714	2020991030900	2020	991030900	2	1	
781715	2020991032575	2020	991032575	1	1	

In [143]:
```
df2.describe().T
```

Out[143]:

	count	mean	std	min	25%	50%	7
accident_year	781716.0	2017.808837	1.377568	2016.0	2017.0	2018.0	201
vehicle_reference	781716.0	1.477553	1.711188	1.0	1.0	1.0	
casualty_reference	781716.0	1.393182	2.454991	1.0	1.0	1.0	
casualty_class	781716.0	1.485053	0.722330	1.0	1.0	1.0	
sex_of_casualty	781716.0	1.395642	0.502698	-1.0	1.0	1.0	
age_of_casualty	781716.0	36.660602	19.393478	-1.0	22.0	33.0	5
age_band_of_casualty	781716.0	6.306182	2.414841	-1.0	5.0	6.0	
casualty_severity	781716.0	2.825593	0.407362	1.0	3.0	3.0	
pedestrian_location	781716.0	0.730536	2.080773	-1.0	0.0	0.0	
pedestrian_movement	781716.0	0.570428	1.871116	-1.0	0.0	0.0	
car_passenger	781716.0	0.252173	0.626313	-1.0	0.0	0.0	
bus_or_coach_passenger	781716.0	0.068772	0.501691	-1.0	0.0	0.0	
pedestrian_road_maintenance_worker	781716.0	0.059004	0.340125	-1.0	0.0	0.0	
casualty_type	781716.0	7.270959	8.371928	-1.0	2.0	9.0	
casualty_home_area_type	781716.0	1.009290	0.977786	-1.0	1.0	1.0	
casualty_imd_decile	781716.0	4.232354	3.288848	-1.0	2.0	4.0	

In [144]:
```
df2.columns
```

Out[144]:
```
Index(['accident_index', 'accident_year', 'accident_reference',
       'vehicle_reference', 'casualty_reference', 'casualty_class',
       'sex_of_casualty', 'age_of_casualty', 'age_band_of_casualty',
       'casualty_severity', 'pedestrian_location', 'pedestrian_movement',
       'car_passenger', 'bus_or_coach_passenger',
       'pedestrian_road_maintenance_worker', 'casualty_type',
       'casualty_home_area_type', 'casualty_imd_decile'],
      dtype='object')
```

In [145]:
```
df2.isnull().sum()
```

Out[145]:
```
accident_index                       0
accident_year                        0
accident_reference                   0
vehicle_reference                    0
casualty_reference                   0
casualty_class                       0
sex_of_casualty                      0
age_of_casualty                      0
age_band_of_casualty                 0
casualty_severity                    0
pedestrian_location                  0
pedestrian_movement                  0
car_passenger                        0
bus_or_coach_passenger               0
pedestrian_road_maintenance_worker   0
casualty_type                        0
casualty_home_area_type              0
casualty_imd_decile                  0
dtype: int64
```

Combining dataFrames

In [146]:
```
f=[df,df2]
c=pd.concat(f,axis=1,join='inner')
c
```

Out[146]:

	accident_index	accident_year	accident_reference	location_easting_osgr	location_northing_
0	2.02E+12	2016	10000005	519310.0	188
1	2.02E+12	2016	10000006	551920.0	174
2	2.02E+12	2016	10000008	505930.0	183
3	2.02E+12	2016	10000016	527770.0	168
4	2.02E+12	2016	10000018	510740.0	177
...	...	...	...	...	
597968	2.02E+12	2020	991027064	343034.0	731
597969	2.02E+12	2020	991029573	257963.0	658
597970	2.02E+12	2020	991030297	383664.0	810
597971	2.02E+12	2020	991030900	277161.0	674
597972	2.02E+12	2020	991032575	240402.0	681

597973 rows × 54 columns

In [147]:
```
c.head().T
```

Out[147]:

	0	1	2	
accident_index	2.02E+12	2.02E+12	2.02E+12	2.
accident_year	2016	2016	2016	
accident_reference	10000005	10000006	10000008	1(
location_easting_osgr	519310.0	551920.0	505930.0	5
location_northing_osgr	188730.0	174560.0	183850.0	1
longitude	-0.279323	0.184928	-0.473837	-0
latitude	51.584754	51.449595	51.543563	51
police_force	1	1	1	
accident_severity	3	3	3	
number_of_vehicles	2	1	1	
number_of_casualties	1	1	1	
date	1/11/2016	1/11/2016	1/11/2016	1/
day_of_week	3	3	3	
time	2:30	0:37	1:25	
local_authority_district	28	18	26	
local_authority_ons_district	E09000005	E09000004	E09000017	E0
local_authority_highway	E09000005	E09000004	E09000017	E0
first_road_class	3	3	3	
first_road_number	4006	207	4020	
road_type	6	6	1	
speed_limit	30.0	30.0	30.0	
junction_detail	0	9	1	
junction_control	-1	4	4	
second_road_class	-1	6	3	
second_road_number	-1	0	4020	
pedestrian_crossing human_control	0	0	0	
pedestrian_crossing_physical_facilities	0	0	0	
light_conditions	5	4	4	
weather_conditions	1	1	1	
road_surface_conditions	1	1	1	
special_conditions_at_site	0	0	0	
carriageway_hazards	0	0	0	
urban_or_rural_area	1	1	1	
did_police_officer_attend_scene_of_accident	1	1	1	

	0	1	2	
trunk_road_flag	2	2	2	
lsoa_of_accident_location	E01000543	E01000375	E01033725	E01
accident_index	2016010000005	2016010000006	2016010000008	201601(
accident_year	2016	2016	2016	
accident_reference	010000005	010000006	010000008	01(
vehicle_reference	2	1	1	
casualty_reference	1	1	1	
casualty_class	1	1	1	
sex_of_casualty	1	2	1	
age_of_casualty	23	36	24	
age_band_of_casualty	5	7	5	
casualty_severity	3	3	3	
pedestrian_location	0	0	0	
pedestrian_movement	0	0	0	
car_passenger	0	0	0	
bus_or_coach_passenger	0	0	0	
pedestrian_road_maintenance_worker	0	0	0	
casualty_type	2	9	9	
casualty_home_area_type	1	1	1	
casualty_imd_decile	4	10	8	

In [148]:
```
c.describe().T
```

Out[148]:

	count	mean	std	m
accident_year	597973.0	2017.827266	1.381319	2016.0000
location_easting_osgr	597850.0	451703.358935	94929.997358	64084.0000
location_northing_osgr	597850.0	282043.804647	152864.644216	10235.0000
longitude	597840.0	-1.260225	1.392031	-7.5252
latitude	597840.0	52.425656	1.376865	49.9123
police_force	597973.0	28.574257	24.915142	1.0000
accident_severity	597973.0	2.792099	0.437881	1.0000
number_of_vehicles	597973.0	1.842209	0.709933	1.0000
number_of_casualties	597973.0	1.307276	0.756494	1.0000
day_of_week	597973.0	4.109935	1.924954	1.0000
local_authority_district	597973.0	328.621888	257.290799	-1.0000
first_road_class	597973.0	4.197810	1.463992	1.0000
first_road_number	597973.0	778.149082	1580.905318	0.0000
road_type	597973.0	5.196603	1.689462	-1.0000
speed_limit	597936.0	37.137533	14.027131	-1.0000
junction_detail	597973.0	2.991710	8.650574	-1.0000
junction_control	597973.0	1.736229	2.495472	-1.0000
second_road_class	597973.0	2.685998	3.158066	-1.0000
second_road_number	597973.0	219.168503	926.539753	-1.0000
pedestrian_crossing human_control	597973.0	0.204884	1.289745	-1.0000
pedestrian_crossing_physical_facilities	597973.0	1.044415	2.260054	-1.0000
light_conditions	597973.0	2.024188	1.714039	-1.0000
weather_conditions	597973.0	1.627798	1.791257	-1.0000
road_surface_conditions	597973.0	1.363968	0.886029	-1.0000
special_conditions_at_site	597973.0	0.195765	1.163435	-1.0000
carriageway_hazards	597973.0	0.151627	1.049526	-1.0000
urban_or_rural_area	597973.0	1.331642	0.471219	-1.0000
did_police_officer_attend_scene_of_accident	597973.0	1.316391	0.537858	-1.0000
trunk_road_flag	597973.0	1.669672	0.863693	-1.0000
accident_year	597973.0	2017.249528	1.038373	2016.0000
vehicle_reference	597973.0	1.478500	1.433171	1.0000
casualty_reference	597973.0	1.394946	1.671250	1.0000
casualty_class	597973.0	1.498211	0.727846	1.0000
sex_of_casualty	597973.0	1.401291	0.498781	-1.0000

	count	mean	std	m
age_of_casualty	597973.0	36.520527	19.375982	-1.0000
age_band_of_casualty	597973.0	6.290008	2.414589	-1.0000
casualty_severity	597973.0	2.834074	0.399097	1.0000
pedestrian_location	597973.0	0.747433	2.096948	-1.0000
pedestrian_movement	597973.0	0.580026	1.880009	-1.0000
car_passenger	597973.0	0.258145	0.634223	-1.0000
bus_or_coach_passenger	597973.0	0.077468	0.530973	-1.0000
pedestrian_road_maintenance_worker	597973.0	0.073752	0.378232	-1.0000
casualty_type	597973.0	7.211219	8.089992	-1.0000
casualty_home_area_type	597973.0	0.974205	0.982771	-1.0000
casualty_imd_decile	597973.0	4.117346	3.300961	-1.0000

In [149]:
```
cdf = c.loc[:,~c.columns.duplicated()]
print(cdf.columns)
print(cdf.shape)
```

```
Index(['accident_index', 'accident_year', 'accident_reference',
       'location_easting_osgr', 'location_northing_osgr', 'longitude',
       'latitude', 'police_force', 'accident_severity', 'number_of_vehicles',
       'number_of_casualties', 'date', 'day_of_week', 'time',
       'local_authority_district', 'local_authority_ons_district',
       'local_authority_highway', 'first_road_class', 'first_road_number',
       'road_type', 'speed_limit', 'junction_detail', 'junction_control',
       'second_road_class', 'second_road_number',
       'pedestrian_crossing human_control',
       'pedestrian_crossing_physical_facilities', 'light_conditions',
       'weather_conditions', 'road_surface_conditions',
       'special_conditions_at_site', 'carriageway_hazards',
       'urban_or_rural_area', 'did_police_officer_attend_scene_of_accident',
       'trunk_road_flag', 'lsoa_of_accident_location', 'vehicle_reference',
       'casualty_reference', 'casualty_class', 'sex_of_casualty',
       'age_of_casualty', 'age_band_of_casualty', 'casualty_severity',
       'pedestrian_location', 'pedestrian_movement', 'car_passenger',
       'bus_or_coach_passenger', 'pedestrian_road_maintenance_worker',
       'casualty_type', 'casualty_home_area_type', 'casualty_imd_decile'],
      dtype='object')
(597973, 51)
```

In [150]:
```
cdf.isnull().sum()
```

Out[150]:
```
accident_index                                  0
accident_year                                   0
accident_reference                              0
location_easting_osgr                         123
location_northing_osgr                        123
longitude                                     133
latitude                                      133
police_force                                    0
accident_severity                               0
number_of_vehicles                              0
number_of_casualties                            0
date                                            0
day_of_week                                     0
time                                            0
local_authority_district                        0
local_authority_ons_district                    0
local_authority_highway                         0
first_road_class                                0
first_road_number                               0
road_type                                       0
speed_limit                                    37
junction_detail                                 0
junction_control                                0
second_road_class                               0
second_road_number                              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
trunk_road_flag                                 0
lsoa_of_accident_location                       0
vehicle_reference                               0
casualty_reference                              0
casualty_class                                  0
sex_of_casualty                                 0
age_of_casualty                                 0
age_band_of_casualty                            0
casualty_severity                               0
pedestrian_location                             0
pedestrian_movement                             0
car_passenger                                   0
bus_or_coach_passenger                          0
pedestrian_road_maintenance_worker              0
casualty_type                                   0
casualty_home_area_type                         0
casualty_imd_decile                             0
dtype: int64
```

In [151]:
```
data=cdf[['accident_year', 'date', 'day_of_week', 'time',
       'longitude','latitude', 'local_authority_district','casualty_home_area_typ
        'sex_of_casualty','age_of_casualty', 'age_band_of_casualty',
        'accident_severity', 'casualty_type','number_of_vehicles',
        'first_road_class', 'road_type', 'speed_limit', 'junction_detail', 'junct
        'light_conditions','weather_conditions', 'road_surface_conditions','spec
print(data.columns)
print(data.shape)
```

```
Index(['accident_year', 'date', 'day_of_week', 'time', 'longitude', 'latitude',
       'local_authority_district', 'casualty_home_area_type',
       'sex_of_casualty', 'age_of_casualty', 'age_band_of_casualty',
       'accident_severity', 'casualty_type', 'number_of_vehicles',
       'first_road_class', 'road_type', 'speed_limit', 'junction_detail',
       'junction_control', 'light_conditions', 'weather_conditions',
       'road_surface_conditions', 'special_conditions_at_site',
       'carriageway_hazards'],
      dtype='object')
(597973, 24)
```

In [152]:
```
data.head()
```

Out[152]:

	accident_year	date	day_of_week	time	longitude	latitude	local_authority_district	casua
0	2016	1/11/2016	3	2:30	-0.279323	51.584754	28	
1	2016	1/11/2016	3	0:37	0.184928	51.449595	18	
2	2016	1/11/2016	3	1:25	-0.473837	51.543563	26	
3	2016	1/11/2016	3	9:15	-0.164442	51.404958	22	
4	2016	1/11/2016	3	7:53	-0.406580	51.483139	25	

5 rows × 24 columns

In [33]:
```
#### done after
```

In [34]:
```
print(data.shape)
data=data.dropna()
print(data.shape)
```

```
(597973, 24)
(597803, 24)
```

Descriptive analysis

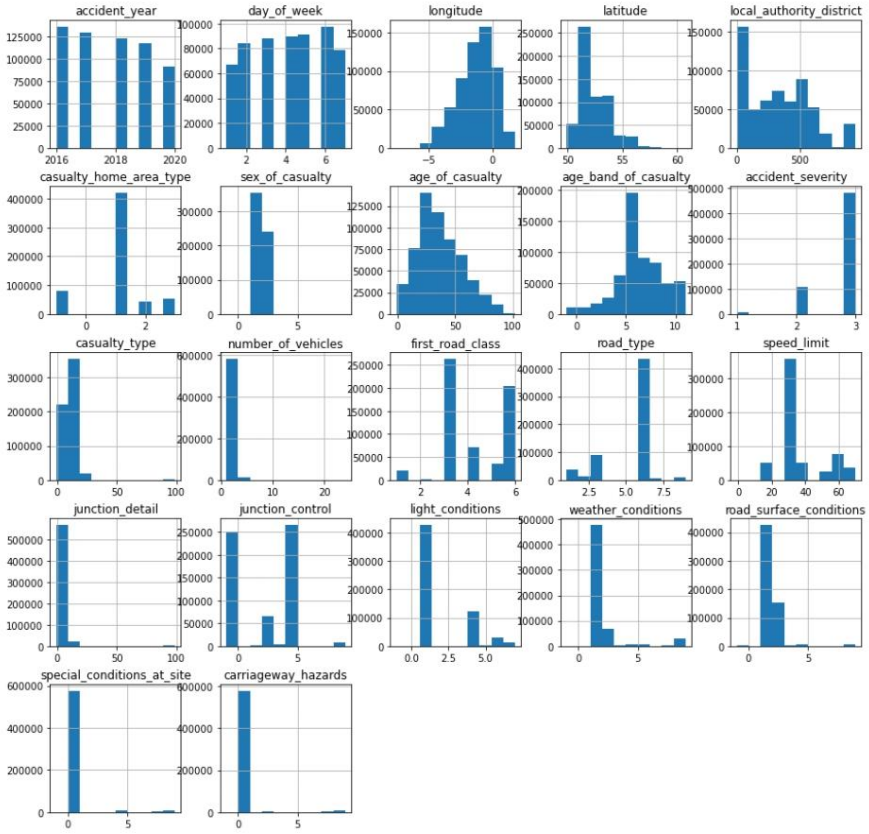
In [23]:
```
data.describe().T
```

Out[23]:

	count	mean	std	min	25%	50
accident_year	597803.0	2017.827319	1.381350	2016.000000	2017.000000	2018.0000
day_of_week	597803.0	4.109941	1.924952	1.000000	2.000000	4.0000
longitude	597803.0	-1.260248	1.392037	-7.525273	-2.180568	-1.1435
latitude	597803.0	52.425666	1.376893	49.912362	51.467411	51.8879
local_authority_district	597803.0	328.604994	257.279172	-1.000000	90.000000	305.0000
casualty_home_area_type	597803.0	0.974257	0.982734	-1.000000	1.000000	1.0000
sex_of_casualty	597803.0	1.401299	0.498785	-1.000000	1.000000	1.0000
age_of_casualty	597803.0	36.520902	19.375838	-1.000000	22.000000	33.0000
age_band_of_casualty	597803.0	6.290069	2.414547	-1.000000	5.000000	6.0000
accident_severity	597803.0	2.792090	0.437892	1.000000	3.000000	3.0000
casualty_type	597803.0	7.211362	8.090810	-1.000000	2.000000	9.0000
number_of_vehicles	597803.0	1.842221	0.709945	1.000000	1.000000	2.0000
first_road_class	597803.0	4.197831	1.463995	1.000000	3.000000	4.0000
road_type	597803.0	5.196553	1.689313	1.000000	6.000000	6.0000
speed_limit	597803.0	37.136077	14.026402	-1.000000	30.000000	30.0000
junction_detail	597803.0	2.992022	8.651654	-1.000000	0.000000	2.0000
junction_control	597803.0	1.736422	2.495475	-1.000000	-1.000000	2.0000
light_conditions	597803.0	2.024152	1.713950	-1.000000	1.000000	1.0000
weather_conditions	597803.0	1.627800	1.791210	-1.000000	1.000000	1.0000
road_surface_conditions	597803.0	1.364006	0.886070	-1.000000	1.000000	1.0000
special_conditions_at_site	597803.0	0.195784	1.163503	-1.000000	0.000000	0.0000
carriageway_hazards	597803.0	0.151650	1.049602	-1.000000	0.000000	0.0000

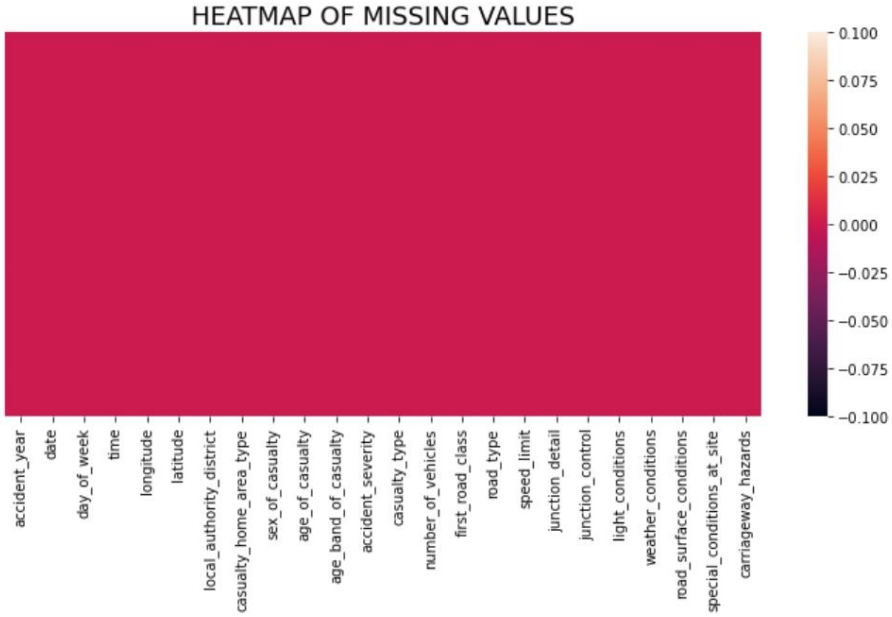
In [24]:
```
data.hist(figsize=(15,15))
```

Out[24]:
```
array([[<AxesSubplot:title={'center':'accident_year'}>,
        <AxesSubplot:title={'center':'day_of_week'}>,
        <AxesSubplot:title={'center':'longitude'}>,
        <AxesSubplot:title={'center':'latitude'}>,
        <AxesSubplot:title={'center':'local_authority_district'}>],
       [<AxesSubplot:title={'center':'casualty_home_area_type'}>,
        <AxesSubplot:title={'center':'sex_of_casualty'}>,
        <AxesSubplot:title={'center':'age_of_casualty'}>,
        <AxesSubplot:title={'center':'age_band_of_casualty'}>,
        <AxesSubplot:title={'center':'accident_severity'}>],
       [<AxesSubplot:title={'center':'casualty_type'}>,
        <AxesSubplot:title={'center':'number_of_vehicles'}>,
        <AxesSubplot:title={'center':'first_road_class'}>,
        <AxesSubplot:title={'center':'road_type'}>,
        <AxesSubplot:title={'center':'speed_limit'}>],
       [<AxesSubplot:title={'center':'junction_detail'}>,
        <AxesSubplot:title={'center':'junction_control'}>,
        <AxesSubplot:title={'center':'light_conditions'}>,
        <AxesSubplot:title={'center':'weather_conditions'}>,
        <AxesSubplot:title={'center':'road_surface_conditions'}>],
       [<AxesSubplot:title={'center':'special_conditions_at_site'}>,
        <AxesSubplot:title={'center':'carriageway_hazards'}>,
        <AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>]], dtype=object)
```


accident_year
day_of_week
longitude
latitude
local_authority_district
casualty_home_area_type
sex_of_casualty
age_of_casualty
age_band_of_casualty
accident_severity
casualty_type
number_of_vehicles
first_road_class
road_type
speed_limit
junction_detail
junction_control
light_conditions
weather_conditions
road_surface_conditions
special_conditions_at_site
carriageway_hazards

In [25]:
```
plt.figure(figsize=(12,5))
plt.title('HEATMAP OF MISSING VALUES', fontsize=18)
sns.heatmap(data.isnull(), yticklabels=False)
```

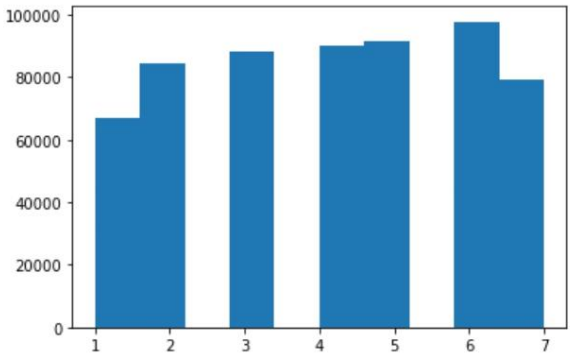
Out[25]: <AxesSubplot:title={'center':'HEATMAP OF MISSING VALUES'}>



Time

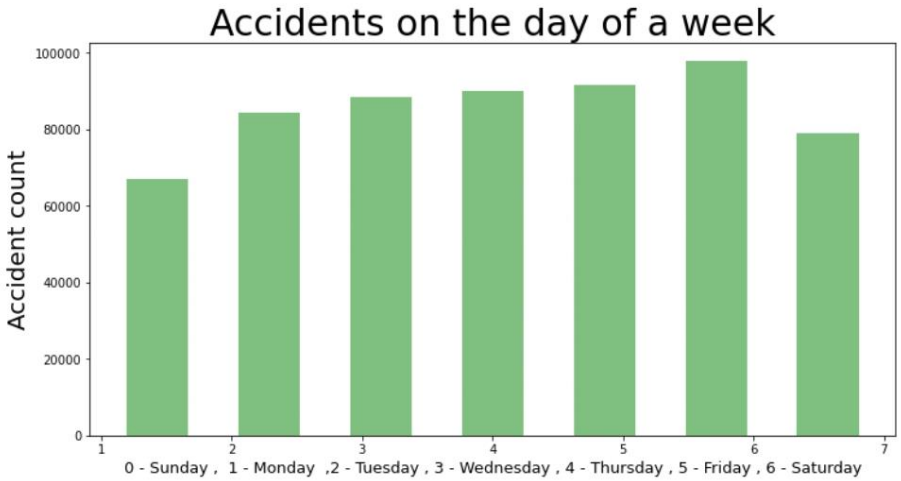
In [26]:
```
plt.hist(data['day_of_week'])
```

Out[26]:
```
(array([67036., 84336.,     0., 88376.,     0., 89902., 91412.,     0.,
        97756., 78985.]),
 array([1. , 1.6, 2.2, 2.8, 3.4, 4. , 4.6, 5.2, 5.8, 6.4, 7. ]),
 <BarContainer object of 10 artists>)
```



In [27]:
```
plt.figure(figsize=(12,6))
data['day_of_week'].hist(bins=7,rwidth=0.55,alpha=0.5, color= 'Green')
plt.title('Accidents on the day of a week' , fontsize= 30)
plt.grid(False)
plt.ylabel('Accident count' , fontsize = 20)
plt.xlabel('0 - Sunday ,  1 - Monday  ,2 - Tuesday , 3 - Wednesday , 4 - Thursday
```

Out[27]:
```
Text(0.5, 0, '0 - Sunday ,  1 - Monday  ,2 - Tuesday , 3 - Wednesday , 4 - Thur
sday , 5 - Friday , 6 - Saturday')
```



In [28]:
```
acc_time = data[['date','day_of_week','time','local_authority_district']]
```

In [29]:
```
acc_time.head()
```

Out[29]:

	date	day_of_week	time	local_authority_district
0	1/11/2016	3	2:30	28
1	1/11/2016	3	0:37	18
2	1/11/2016	3	1:25	26
3	1/11/2016	3	9:15	22
4	1/11/2016	3	7:53	25

In [30]:
```
acc_time.rename(columns={'date':'Date','day_of_week':'Day_of_Week','time':'Time'}
```

```
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\frame.py:5039: SettingWi
thCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
  return super().rename(
```

In [31]:
```
acc_time.dropna(axis=0,inplace=True)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\pandas\util\_decorators.py:311: Sett
ingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
  return func(*args, **kwargs)
```

In [32]:
```
acc_time.columns
```

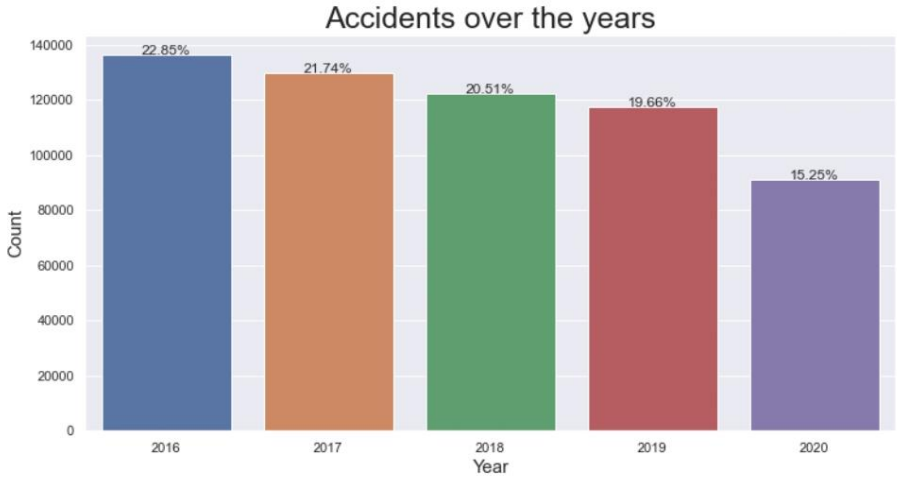
Out[32]:
```
Index(['Date', 'Day_of_Week', 'Time', 'local_authority_district'], dtype='objec
t')
```

In [33]:
```
def year(string):
    return int(string.split('/')[-1])
acc_time['Year']=acc_time['Date'].apply(lambda x: year(x))
#creating month column
def month(string):
    return int(string.split('/')[1])
acc_time['Month']=acc_time['Date'].apply(lambda x: month(x))
#creating day column
def day(string):
    return int(string.split('/')[0])
acc_time['Day']=acc_time['Date'].apply(lambda x: day(x))
#creating hour column
def hour(string):
    s=string.split(':')[0]
    return int(s)
acc_time['Hour']=acc_time['Time'].apply(lambda x: hour(x))
```

```
C:\Users\mdasi\AppData\Local\Temp/ipykernel_7032/2820701318.py:3: SettingWith
CopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (https://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy)
  acc_time['Year']=acc_time['Date'].apply(lambda x: year(x))
C:\Users\mdasi\AppData\Local\Temp/ipykernel_7032/2820701318.py:7: SettingWith
CopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (https://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy)
  acc_time['Month']=acc_time['Date'].apply(lambda x: month(x))
C:\Users\mdasi\AppData\Local\Temp/ipykernel_7032/2820701318.py:11: SettingWit
hCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (https://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy)
  acc_time['Day']=acc_time['Date'].apply(lambda x: day(x))
C:\Users\mdasi\AppData\Local\Temp/ipykernel_7032/2820701318.py:16: SettingWit
hCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (https://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
```

```
sus-a-copy)
  acc_time['Hour']=acc_time['Time'].apply(lambda x: hour(x))
```

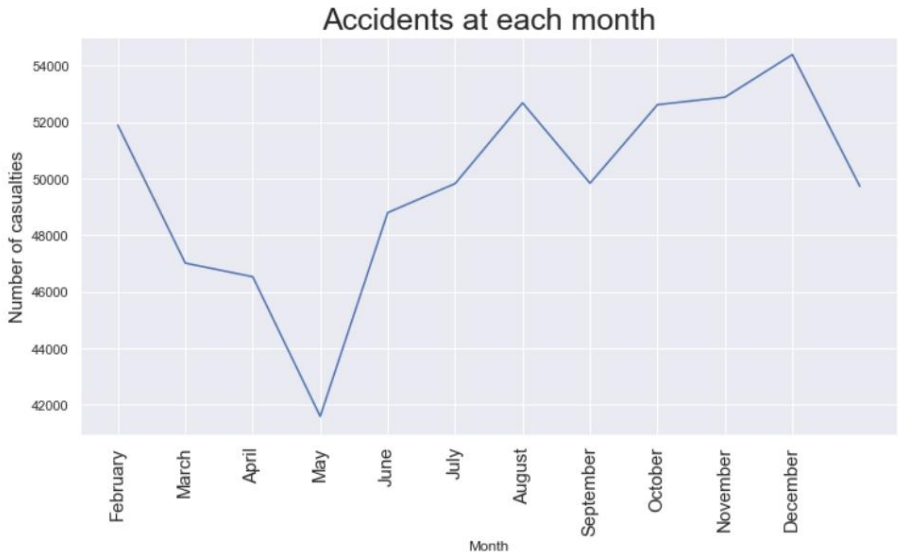
In [34]:
```
sns.set(style="darkgrid")
plt.figure(figsize=(12,6))
plt.title('Accidents over the years',fontsize=25)
plt.ylabel('Number of casualties' , fontsize = 15)
plt.xlabel('year', fontsize = 15)
yearplot = sns.countplot(x='Year',data=acc_time)
yearplot.set(xlabel='Year', ylabel='Count')
for p in yearplot.patches:
    height = p.get_height()
    yearplot.text(p.get_x()+p.get_width()/2.,
      height + 3,
      '{:1.2f}'.format((height/len(acc_time))*100)+'%',
      ha="center")
plt.show()
```



In [35]:
```
xx=acc_time.groupby('Month').size()
xx
```

Out[35]:
```
Month
1     51895
2     47013
3     46530
4     41588
5     48798
6     49827
7     52684
8     49840
9     52620
10    52885
11    54392
12    49731
dtype: int64
```

In [36]:
```
objects=['January','February','March','April','May','June','July','August','Septe
sns.set(style="darkgrid")
plt.figure(figsize=(12,6))
plt.title('Accidents at each month',fontsize=25)
plt.ylabel('Number of casualties' , fontsize = 15)
y_pos = np.arange(len(objects))
plt.xticks(y_pos , objects,fontsize=15, rotation=90)
monthh=sns.lineplot(data=xx, x=xx.index,y=acc_time.groupby('Month').size(),marker
plt.show()
```



In [37]:
```
objects=['January','February','March','April','May','June','July','August','Septe
sns.set(style="darkgrid")
plt.figure(figsize=(12,6))
plt.title('Accidents at each month',fontsize=25)
plt.ylabel('Number of casualties' , fontsize = 15)
monthh=sns.lineplot(data=acc_time, x=acc_time.index,y=acc_time.groupby('Date').si
plt.show()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_7032/4066601272.py in <module>
      4 plt.title('Accidents at each month',fontsize=25)
      5 plt.ylabel('Number of casualties' , fontsize = 15)
----> 6 monthh=sns.lineplot(data=acc_time, x=acc_time.index,y=acc_time.groupb
y('Date').size(),markers='*')
      7 plt.show()

C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py in inner_f
(*args, **kwargs)
     44             )
     45         kwargs.update({k: arg for k, arg in zip(sig.parameters, args)
})
---> 46         return f(**kwargs)
     47     return inner_f
     48

C:\ProgramData\Anaconda3\lib\site-packages\seaborn\relational.py in lineplot
```

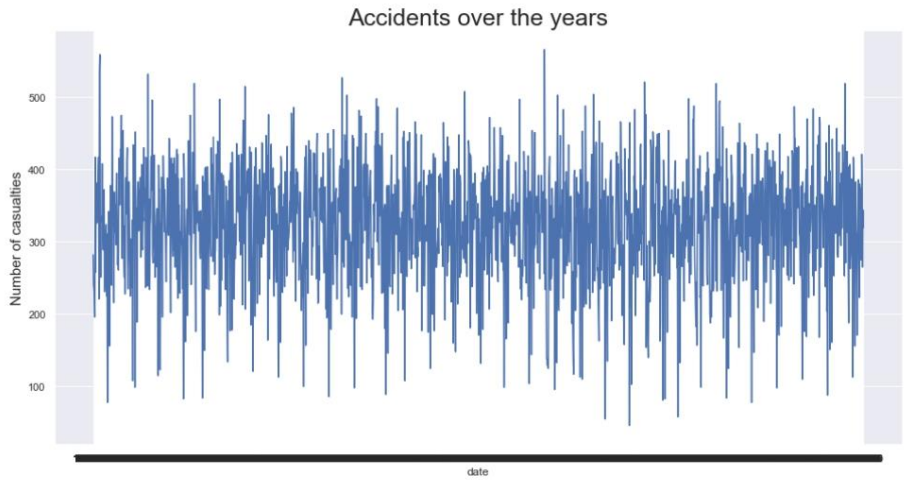
In [38]:
```
x=data.groupby('date').agg({'accident_severity':np.count_nonzero})
x
```

Out[38]:

date	accident_severity
1/1/2016	282
1/1/2017	241
1/1/2018	234
1/1/2019	231
1/1/2020	196
...	...
9/9/2016	421
9/9/2017	382
9/9/2018	265
9/9/2019	344
9/9/2020	320

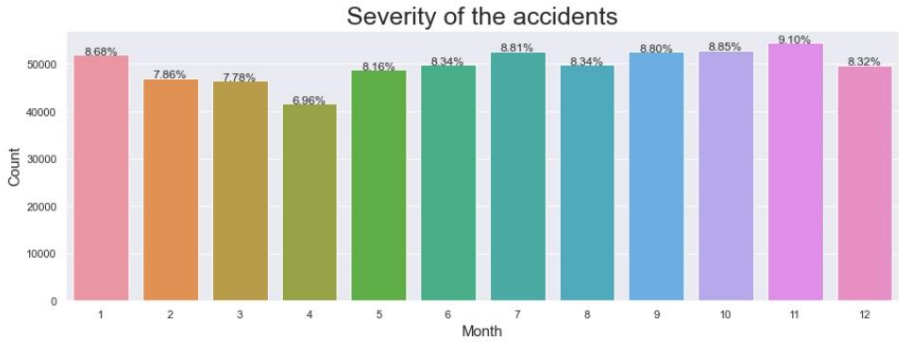
1827 rows × 1 columns

In [39]:
```
sns.set(style="darkgrid")
plt.figure(figsize=(16,8))
plt.title('Accidents over the years',fontsize=25)
plt.ylabel('Number of casualties' , fontsize = 15)
monthh=sns.lineplot(data=x, x=x.index,y=x['accident_severity'])
plt.show()
```

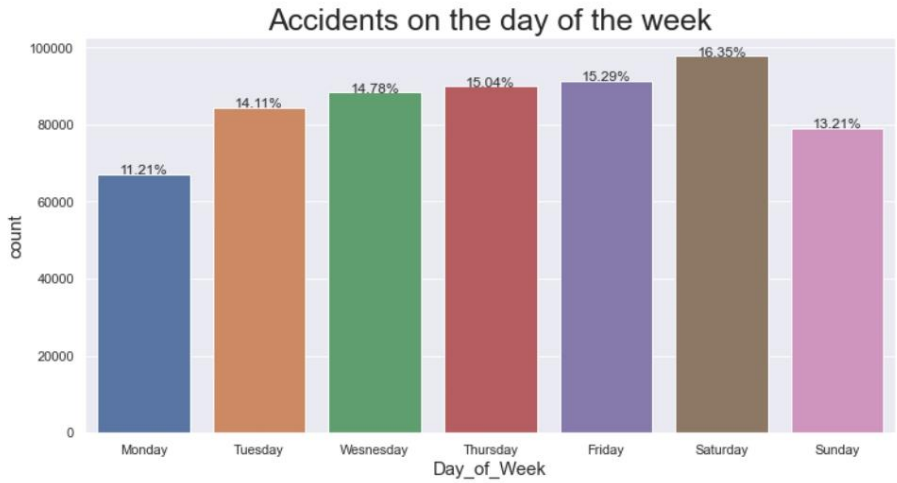


In [ ]:

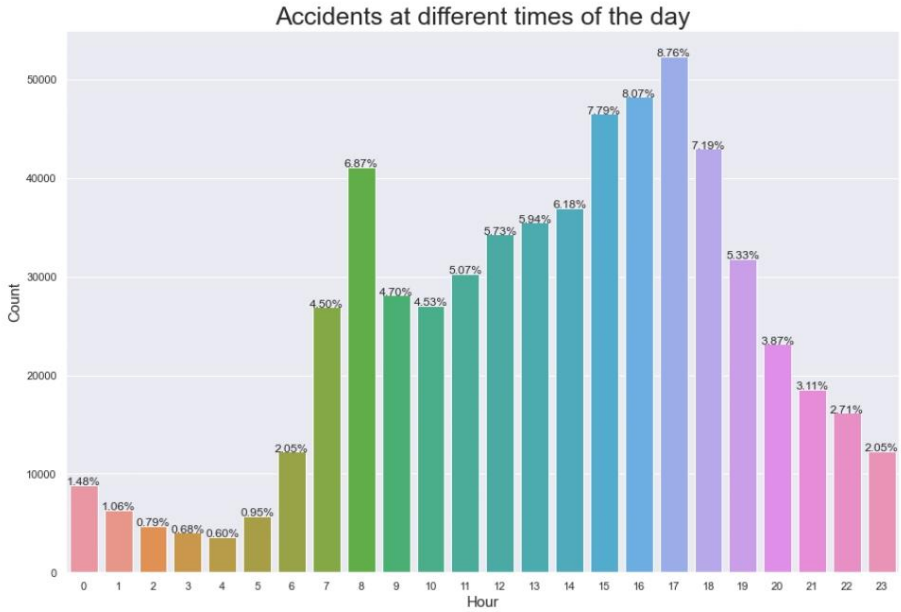
In [40]:
```
sns.set(style="darkgrid")
plt.figure(figsize=(15,5))
plt.title('Severity of the accidents',fontsize=25)
plt.ylabel('Number of casualties' , fontsize = 15)
plt.xlabel('Severity', fontsize = 15)
monthplot = sns.countplot(x='Month',data=acc_time)
monthplot.set(xlabel='Month', ylabel='Count')
for p in monthplot.patches:
    height = p.get_height()
    monthplot.text(p.get_x()+p.get_width()/2.,
      height + 3,
      '{:1.2f}'.format((height/len(acc_time))*100)+'%',
      ha="center")
plt.show()
```



In [41]:
```
sns.set(style="darkgrid")
plt.figure(figsize=(12,6))
plt.title('Accidents on the day of the week',fontsize=25)
plt.ylabel('Number of casualties' , fontsize = 15)
plt.xlabel('Weekdays', fontsize = 15)
weekplot = sns.countplot(x='Day_of_Week',data=acc_time)
for p in weekplot.patches:
    height = p.get_height()
    weekplot.text(p.get_x()+p.get_width()/2.,
      height + 3,
      '{:1.2f}'.format((height/len(acc_time))*100)+'%',
      ha="center")
weekplot.set(xticklabels=['Monday','Tuesday','Wesnesday','Thursday','Friday','Sat
plt.show()
```



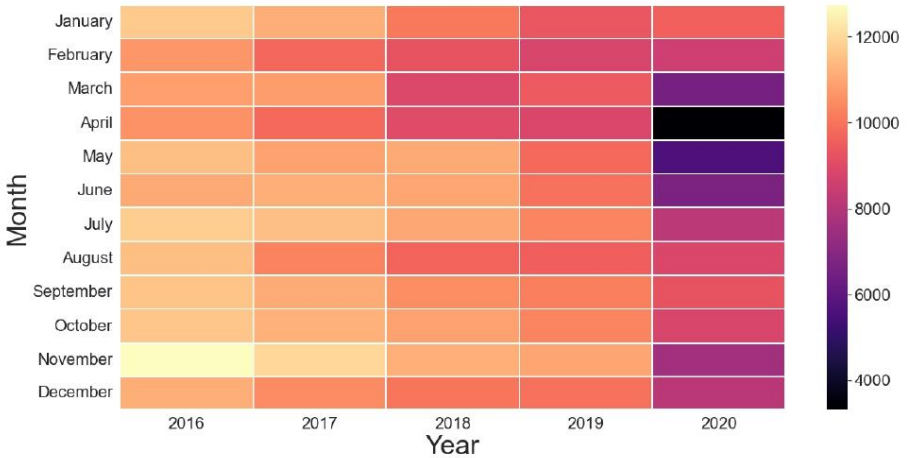
In [42]:
```
sns.set(style="darkgrid")
plt.figure(figsize=(15,10))
plt.title('Accidents at different times of the day',fontsize=25)
plt.ylabel('Number of casualties' , fontsize = 15)
plt.xlabel('Time of the day', fontsize = 15)
Hourplot = sns.countplot(x='Hour',data=acc_time)
Hourplot.set(xlabel='Hour', ylabel='Count')
for p in Hourplot.patches:
    height = p.get_height()
    Hourplot.text(p.get_x()+p.get_width()/2.,
      height + 3,
      '{:1.2f}'.format((height/len(acc_time))*100)+'%',
      ha="center")
plt.show()
```



In [43]:
```
acc_time['Ones']=1
table = pd.pivot_table(acc_time, values='Ones', index=['Month'],columns=['Year'],
plt.figure(figsize=(20,10))
yticks = np.array(['January','February','March','April','May','June','July','Augu
sns.set(rc={"axes.labelsize":36},font_scale=2)
sns.heatmap(table, yticklabels=yticks,linewidths=.1,annot=False,cmap='magma')
```

```
C:\Users\mdasi\AppData\Local\Temp/ipykernel_7032/7994104.py:1: SettingWithCopyW
arning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
  acc_time['Ones']=1
```

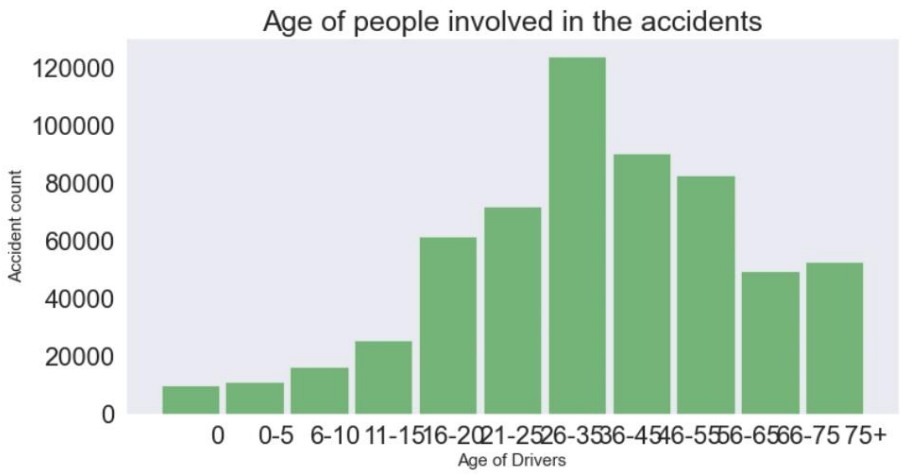
Out[43]: <AxesSubplot:xlabel='Year', ylabel='Month'>



People

In [44]:
```
objects = ['0','0-5','6-10','11-15','16-20','21-25','26-35',
           '36-45', '46-55','56-65','66-75','75+']

plt.figure(figsize=(12,6))
data.age_band_of_casualty.hist(bins = 11,alpha=0.5,rwidth=0.90, color= 'green',)
plt.title('Age of people involved in the accidents', fontsize = 25)
plt.grid(False)
y_pos = np.arange(len(objects))
plt.xticks(y_pos , objects)
plt.ylabel('Accident count' , fontsize = 15)
plt.xlabel('Age of Drivers', fontsize = 15,)
```

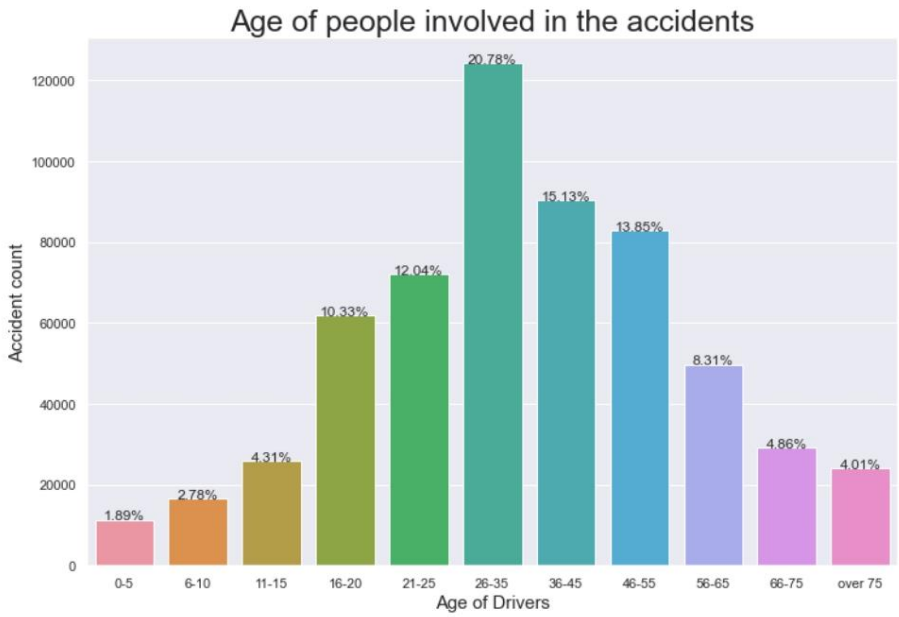
Out[44]: Text(0.5, 0, 'Age of Drivers')



In [45]:
```
#Turning the variable into a categorical one
def map_age(age):
    if age == 1:
        return '0-5'
    elif age == 2:
        return '6-10'
    elif age == 3:
        return '11-15'
    elif age == 4:
        return '16-20'
    elif age == 5:
        return '21-25'
    elif age == 6:
        return '26-35'
    elif age == 7:
        return '36-45'
    elif age == 8:
        return '46-55'
    elif age == 9:
        return '56-65'
    elif age == 10:
        return '66-75'
    elif age == 11:
        return 'over 75'
    elif age == -1:
        return "Don't know"


data['age_band_of_casualty'] = data['age_band_of_casualty'].apply(map_age)
```

In [46]:
```
sns.set(style="darkgrid")
plt.figure(figsize=(12,8))
plt.title('Age of people involved in the accidents', fontsize = 25)
ageplot=sns.countplot(x='age_band_of_casualty',data=data,order=['0-5','6-10','11-
for p in ageplot.patches:
    height = p.get_height()
    ageplot.text(p.get_x()+p.get_width()/2.,
      height + 3,
      '{:1.2f}'.format((height/len(data))*100)+'%',
      ha="center")
plt.ylabel('Accident count' , fontsize = 15)
plt.xlabel('Age of Drivers', fontsize = 15,)
```

Out[46]: Text(0.5, 0, 'Age of Drivers')



In [47]:
```
data['sex_of_casualty'].unique()
```

Out[47]: array([1, 2, -1, 9], dtype=int64)

In [48]:
```
data=data[(data['sex_of_casualty']!=-1) & (data['sex_of_casualty']!=9)]
data['sex_of_casualty'].unique()
```

Out[48]: array([1, 2], dtype=int64)

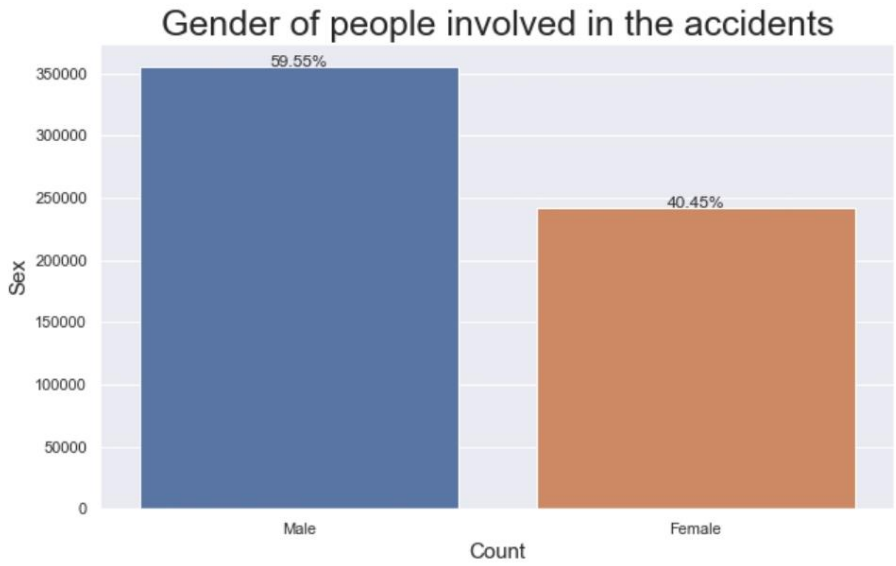
In [49]:
```
def map_sex(sex):
    if sex==1:
        return 'Male'
    else:
        return 'Female'
data['sex_of_casualty']=data['sex_of_casualty'].apply(map_sex)
```

In [50]:
```
data['sex_of_casualty'].unique()
```

Out[50]:
```
array(['Male', 'Female'], dtype=object)
```

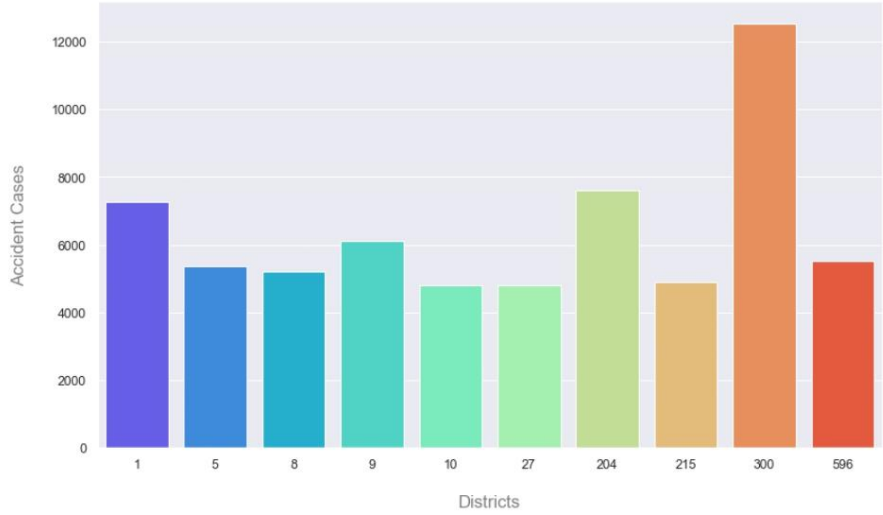
In [51]:
```
sns.set(style='darkgrid')
plt.figure(figsize=(10,6))
plt.title('Gender of people involved in the accidents', fontsize = 25)
genderplot=sns.countplot(x='sex_of_casualty',data=data)
for p in genderplot.patches:
    h=p.get_height()
    genderplot.text(p.get_x()+p.get_width()/2.,h+3,
                    '{:1.2f}'.format((h/len(data))*100)+'%',ha='center')
plt.ylabel('Sex' , fontsize = 15)
plt.xlabel('Count', fontsize = 15,)
```

Out[51]:
```
Text(0.5, 0, 'Count')
```



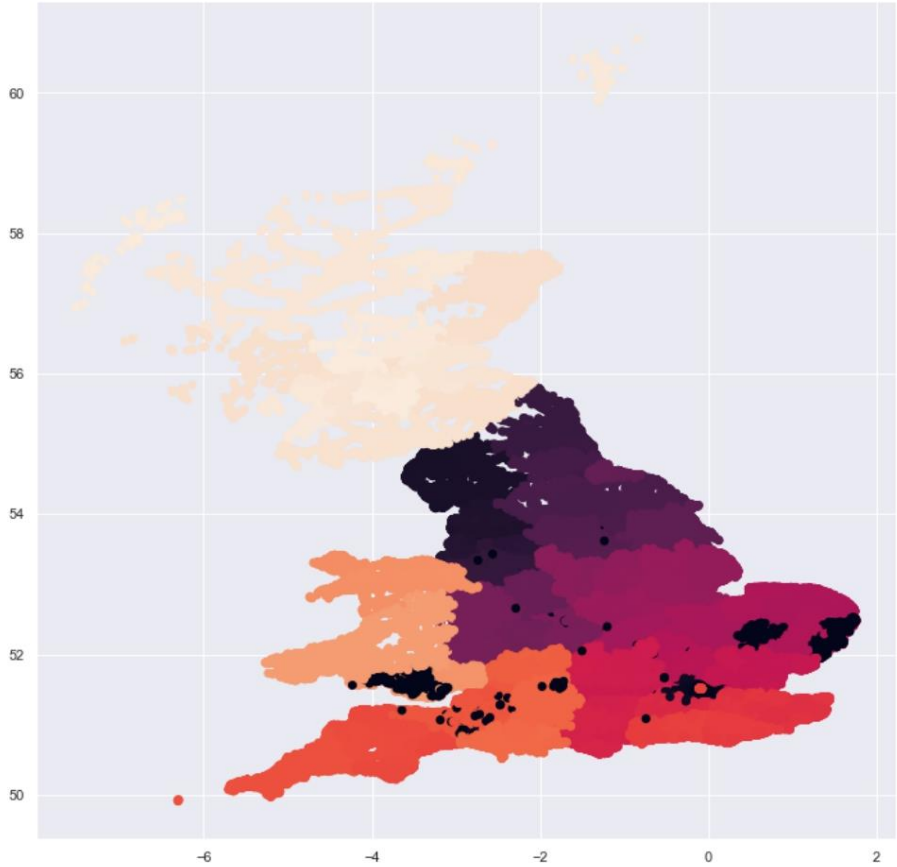
Location

In [52]:
```
district=pd.DataFrame(data['local_authority_district'].value_counts()).reset_inde
top_10_dis=pd.DataFrame(district.head(10))

fig, ax = plt.subplots(figsize = (12,7), dpi = 80)

ax=sns.barplot(y=top_10_dis['Cases'], x=top_10_dis['local_authority_district'], p

ax.set_xlabel('\nDistricts\n', fontsize=15, color='grey')
ax.set_ylabel('\nAccident Cases\n', fontsize=15, color='grey')

plt.show()
```

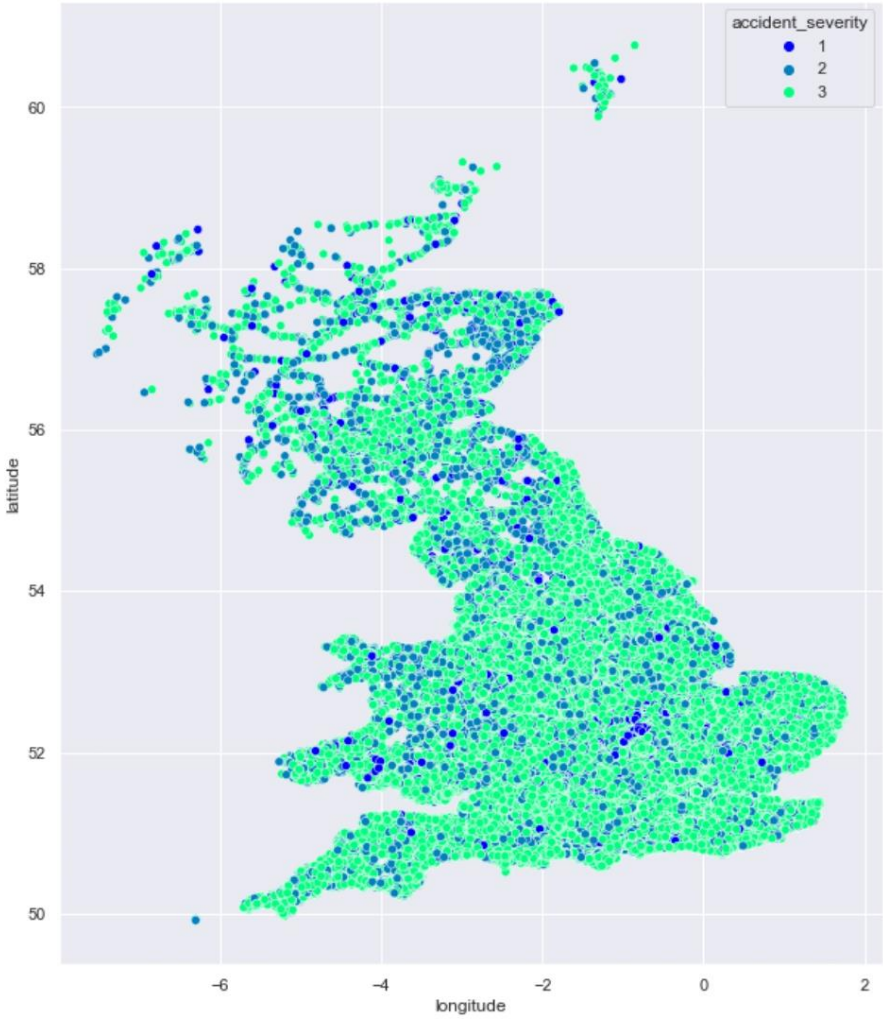


In [53]:
```
plt.figure(figsize=(12,12))
plt.scatter(data['longitude'],data['latitude'],c=data['local_authority_district']
```

Out[53]:
```
<matplotlib.collections.PathCollection at 0x2690e90be50>
```



In [54]:
```
plt.figure(figsize=(10,12))
sns.scatterplot(data=data, x = "longitude", y = "latitude", hue = "accident_sever
plt.show()
```

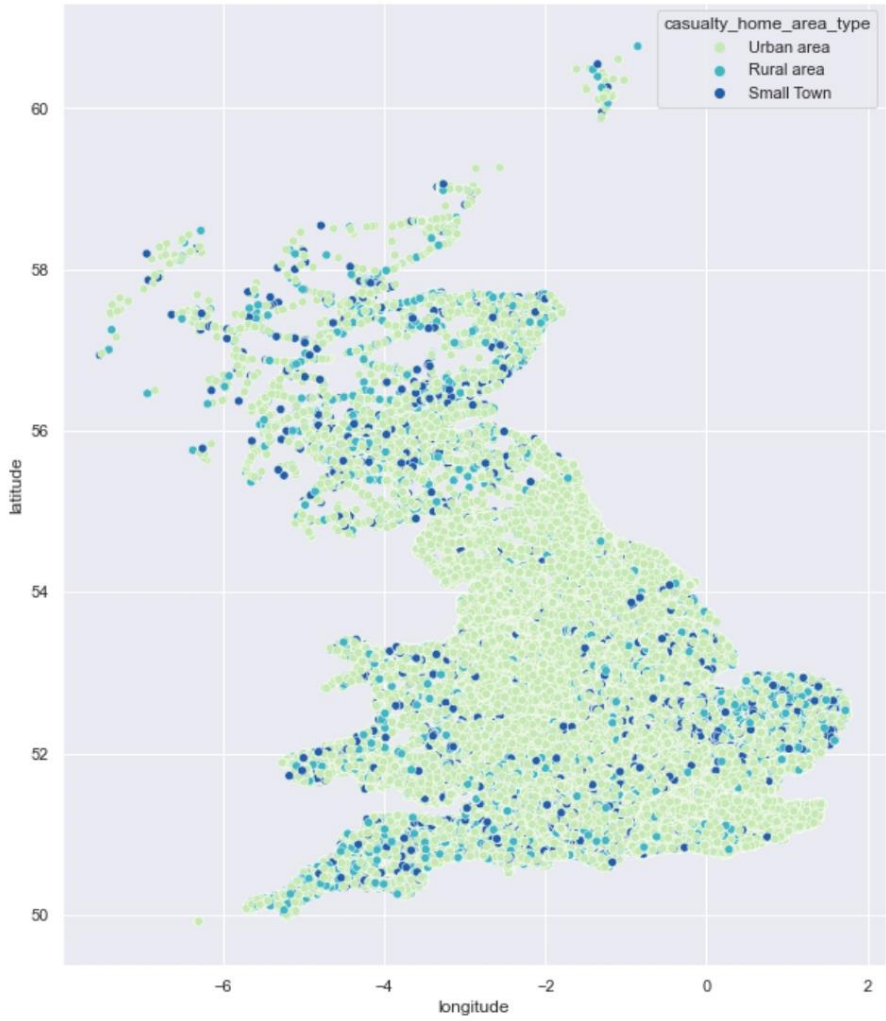


In [55]:
```
data=data[(data['casualty_home_area_type']!=-1)]
data['casualty_home_area_type'].unique()
```

Out[55]:
```
array([1, 3, 2], dtype=int64)
```

In [56]:
```
def map_area(area):
    if area==1:
        return 'Urban area'
    elif area==2:
        return 'Small Town'
    elif area==3:
        return 'Rural area'
data['casualty_home_area_type']=data['casualty_home_area_type'].apply(map_area)
```

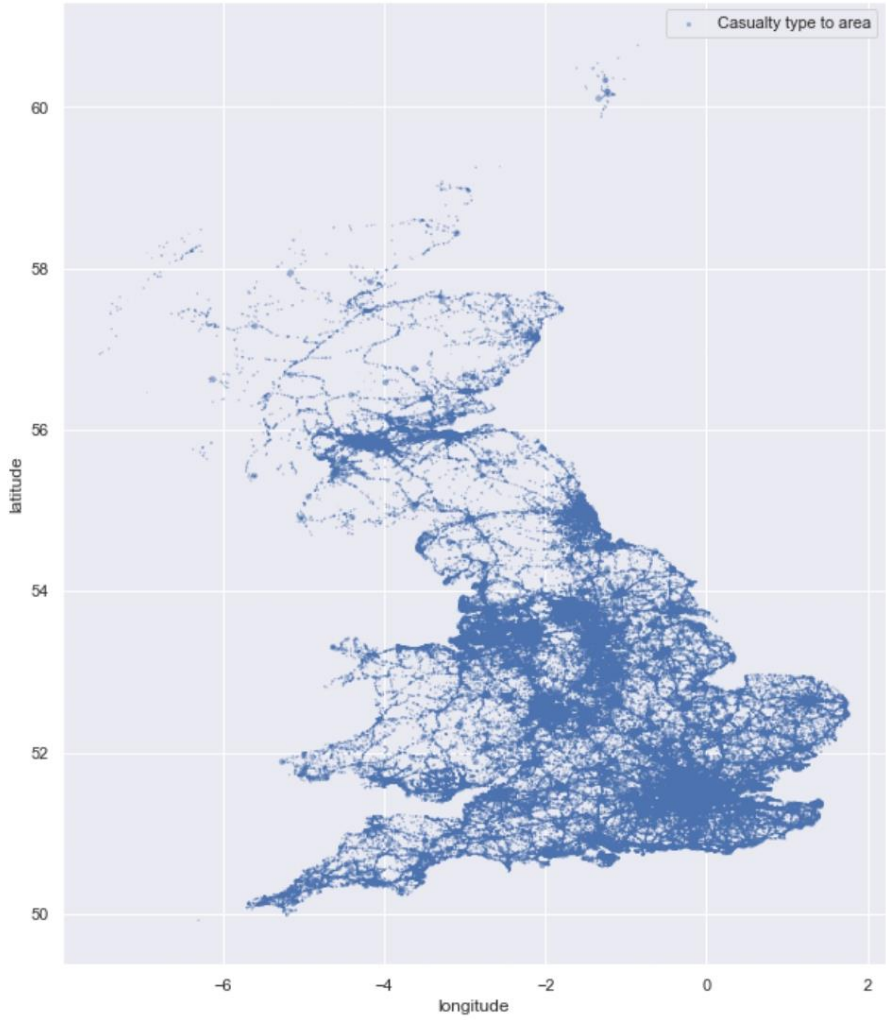
In [57]:
```
plt.figure(figsize=(10,12))
sns.scatterplot(data=data, x = "longitude", y = "latitude", hue = 'casualty_home_
plt.show()
```



In [58]:
```
plt.figure(figsize=(10,12))
sns.scatterplot(data=data,x = "longitude", y = "latitude", alpha = 0.5,
             s = data['casualty_type']/5,label='Casualty type to area', cmap = pl
```

```
C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\collections.py:1003: Runt
imeWarning: invalid value encountered in sqrt
  scale = np.sqrt(self._sizes) * dpi / 72.0 * self._factor
```

Out[58]:
```
<AxesSubplot:xlabel='longitude', ylabel='latitude'>
```



In [ ]:

In [ ]:

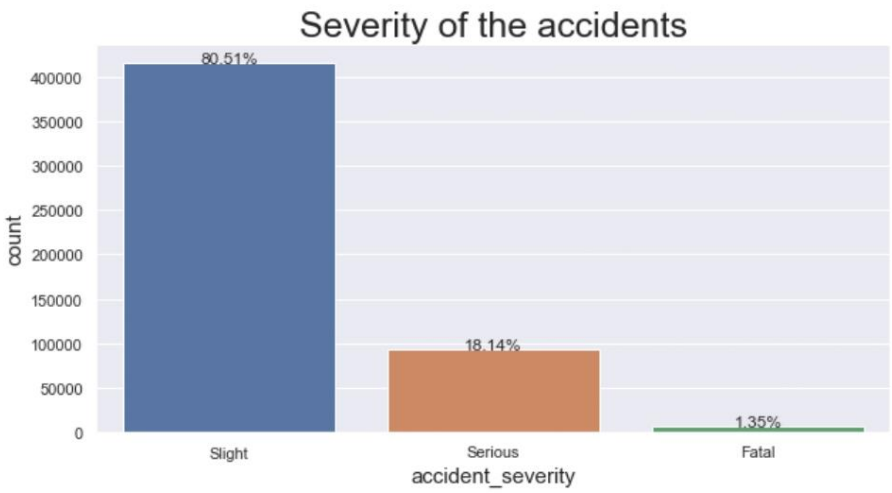
Severity

In [59]:
```
data['accident_severity'].unique()
```

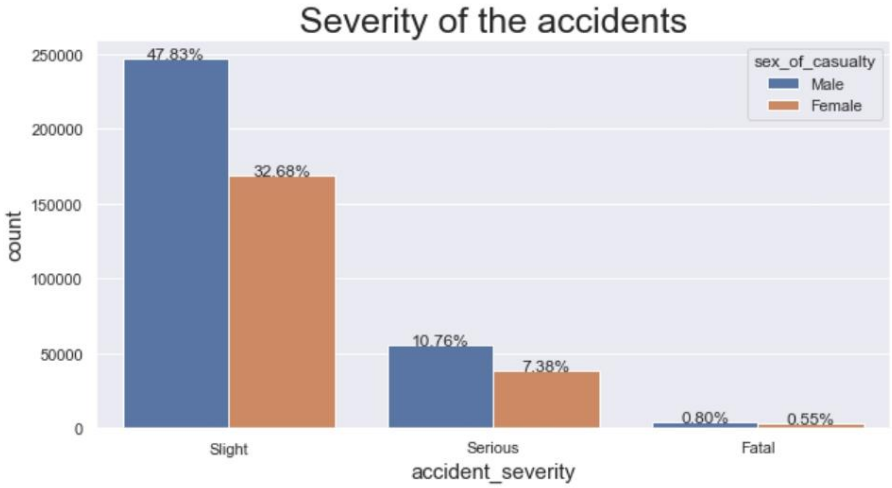
Out[59]:
```
array([3, 2, 1], dtype=int64)
```

In [60]:
```
def map_severity(severity):
    if severity == 1:
        return 'Fatal'
    elif severity == 2:
        return 'Serious'
    elif severity == 3:
        return 'Slight'

data['accident_severity'] = data['accident_severity'].apply(map_severity)
```

In [61]:
```
sns.set(style="darkgrid")
plt.figure(figsize=(10,5))
plt.title('Severity of the accidents',fontsize=25)
plt.ylabel('Number of casualties' , fontsize = 15)
plt.xlabel('Severity', fontsize = 15,)
severityplot = sns.countplot(x='accident_severity',data=data,order=['Slight','Ser
for p in severityplot.patches:
    height = p.get_height()
    severityplot.text(p.get_x()+p.get_width()/2.,
      height + 3,
      '{:1.2f}'.format((height/len(data))*100)+'%',
      ha="center")
```

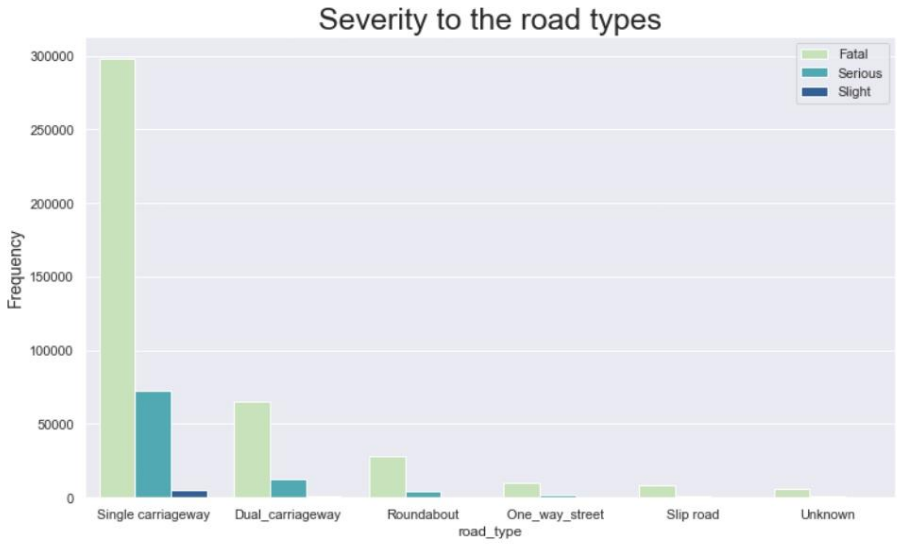


In [62]:
```
sns.set(style="darkgrid")
plt.figure(figsize=(10,5))
plt.title('Severity of the accidents',fontsize=25)
plt.ylabel('Number of casualties' , fontsize = 15)
plt.xlabel('Severity', fontsize = 15)
severityplot = sns.countplot(x='accident_severity',hue='sex_of_casualty',data=dat
for p in severityplot.patches:
    height = p.get_height()
    severityplot.text(p.get_x()+p.get_width()/2.,
      height + 3,
      '{:1.2f}'.format((height/len(data))*100)+'%',
      ha="center")
```



In [63]:
```
sns.set(style="darkgrid")
plt.figure(figsize=(12,7))
plt.title('Severity to the road types',fontsize=25)
ax = sns.countplot('road_type',hue='accident_severity',data=data,
                   order = data["road_type"].value_counts().index, palette = "Yl
ax.set_xticklabels(['Single carriageway','Dual_carriageway','Roundabout',
                    'One_way_street ',
                    'Slip road','Unknown'])
plt.legend(['Fatal','Serious','Slight'])
plt.ylabel("Frequency", fontsize = 14)
plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWar
ning: Pass the following variable as a keyword arg: x. From version 0.12, the o
nly valid positional argument will be `data`, and passing other arguments witho
ut an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

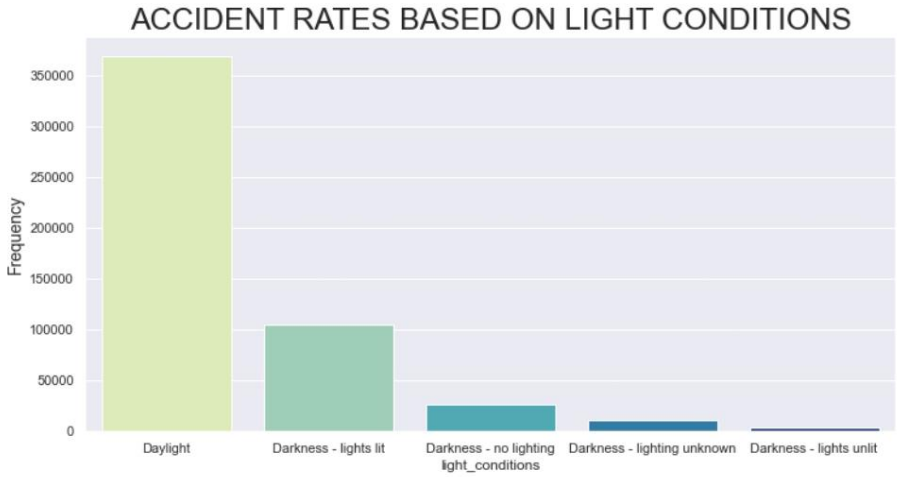


Weather conditions

In [64]:
```
data=data[data['light_conditions']!=-1]
```

In [65]:
```
plt.figure(figsize=(12,6))
ax=sns.countplot('light_conditions',data=data,
                 order = data["light_conditions"].value_counts().index,
                 palette = "YlGnBu")
ax.set_xticklabels(['Daylight','Darkness - lights lit',
                    'Darkness - no lighting',
                    'Darkness - lighting unknown',
                    'Darkness - lights unlit'])
plt.title('ACCIDENT RATES BASED ON LIGHT CONDITIONS',fontsize=25)
plt.ylabel("Frequency", fontsize = 14)
plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWar
ning: Pass the following variable as a keyword arg: x. From version 0.12, the o
nly valid positional argument will be `data`, and passing other arguments witho
ut an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



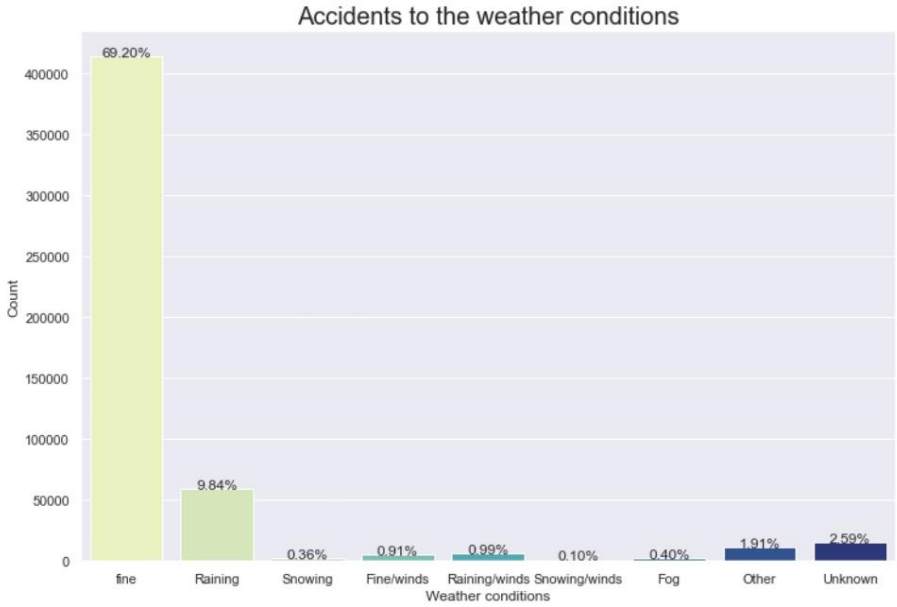
In [66]:
```
df_conditions = data[['light_conditions','weather_conditions','road_surface_cond
```

In [67]:
```
df_conditions['weather_conditions'].unique()
```

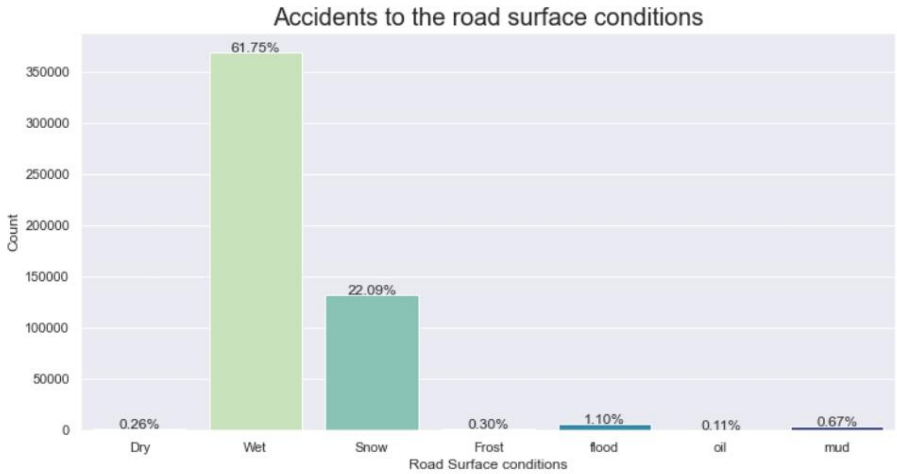
Out[67]:
```
array([ 1,  2,  7,  8,  9,  4,  5,  3,  6, -1], dtype=int64)
```

In [68]:
```
df_conditions=df_conditions[df_conditions['weather_conditions']!=-1]
```

In [69]:
```
sns.set(style="darkgrid")
plt.figure(figsize=(12,8))
weatherplot = sns.countplot(x='weather_conditions',data=df_conditions,palette = '
weatherplot.set(xlabel='Weather conditions', ylabel='Count',xticklabels=['fine','
                                                                          'Raining

for p in weatherplot.patches:
    height = p.get_height()
    weatherplot.text(p.get_x()+p.get_width()/2.,
      height + 3,
      '{:1.2f}'.format((height/len(acc_time))*100)+'%',
      ha="center")
plt.title('Accidents to the weather conditions',fontsize=20)
plt.show()
```

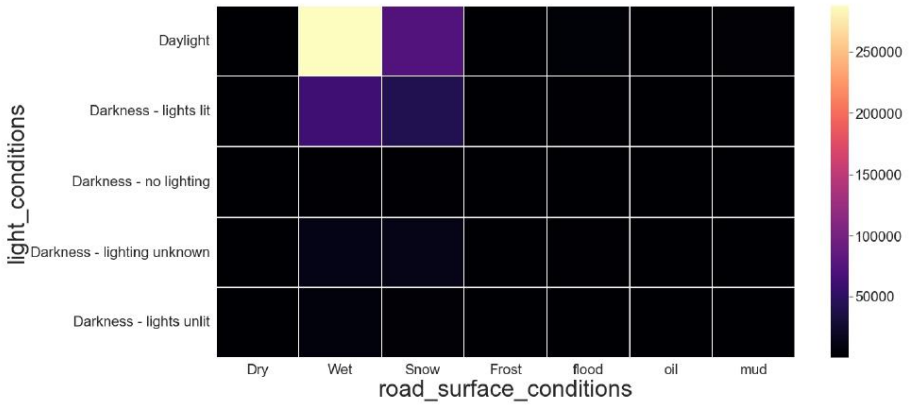


In [70]:
```
sns.set(style="darkgrid")
plt.figure(figsize=(12,6))
roadplot = sns.countplot(x='road_surface_conditions',data=df_conditions,palette =
roadplot.set(xlabel='Road Surface conditions', ylabel='Count',xticklabels=['Dry',
for p in roadplot.patches:
    height = p.get_height()
    roadplot.text(p.get_x()+p.get_width()/2.,
      height + 3,
      '{:1.2f}'.format((height/len(acc_time))*100)+'%',
      ha="center")
plt.title('Accidents to the road surface conditions',fontsize=20)
plt.show()
```



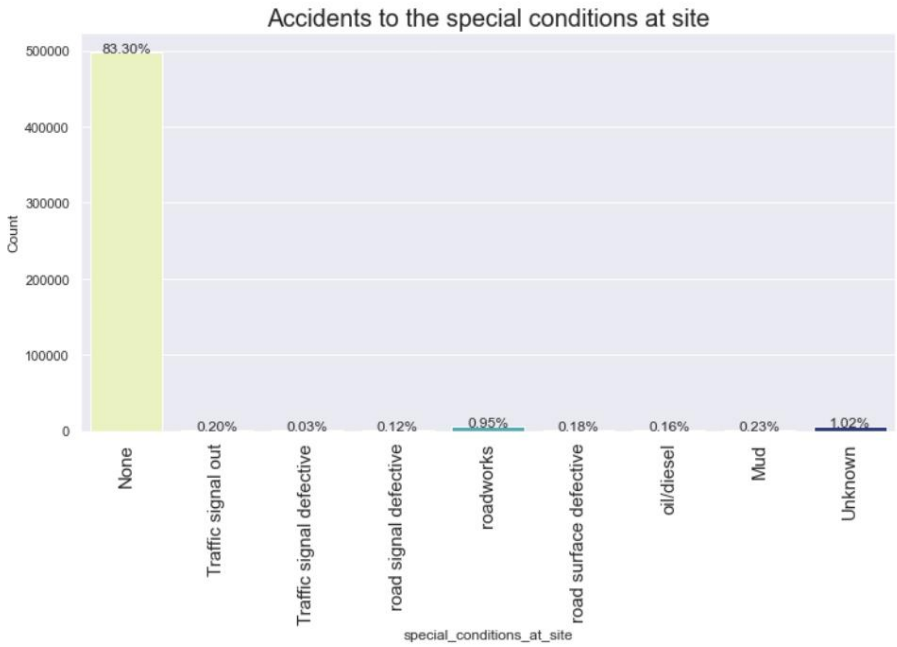
In [71]:
```
df_conditions['Ones']=1
table = pd.pivot_table(df_conditions, values='Ones', index=['light_conditions'],c
plt.figure(figsize=(20,10))
yticks = np.array(['Daylight','Darkness - lights lit','Darkness - no lighting','D
xticks = np.array(['Dry','Wet','Snow','Frost','flood','oil','mud'])
sns.set(rc={"axes.labelsize":36},font_scale=2)
sns.heatmap(table, yticklabels=yticks,xticklabels=xticks,linewidths=.1,annot=Fals
```

Out[71]: <AxesSubplot:xlabel='road_surface_conditions', ylabel='light_conditions'>



In [72]:
```
data=data[data['special_conditions_at_site']!=-1]
```

In [73]:
```
sns.set(style="darkgrid")
plt.figure(figsize=(12,6))
roadplot = sns.countplot(x='special_conditions_at_site',data=data,palette = "YlGr
roadplot.set(xlabel='special_conditions_at_site', ylabel='Count',xticklabels=['No
for p in roadplot.patches:
    height = p.get_height()
    roadplot.text(p.get_x()+p.get_width()/2.,
      height + 3,
      '{:1.2f}'.format((height/len(acc_time))*100)+'%',
      ha="center")
plt.title('Accidents to the special conditions at site',fontsize=20)
plt.xticks(rotation=90,fontsize=15)
plt.show()
```

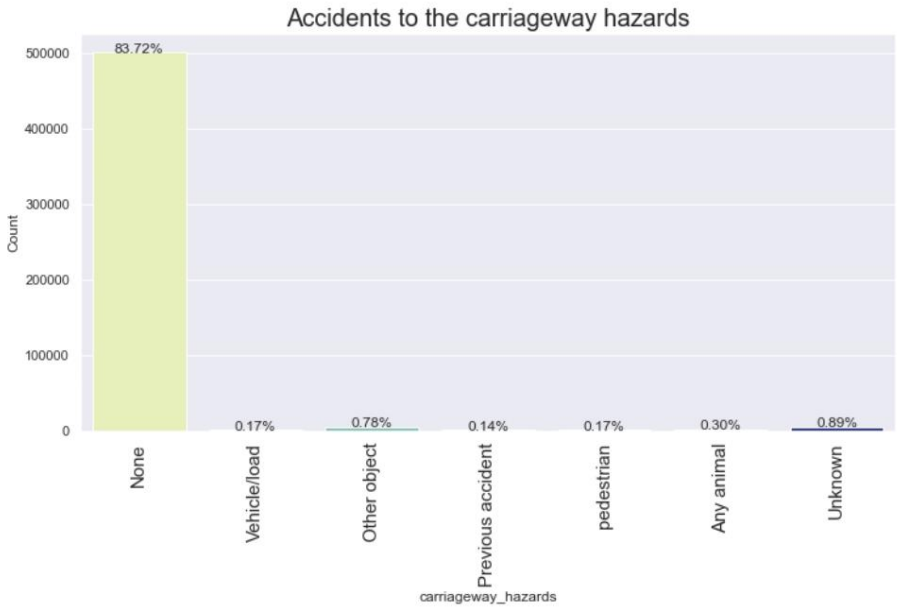


In [74]:
```
data=data[data['carriageway_hazards']!=-1]
```

In [75]:
```
data['carriageway_hazards'].unique()
```

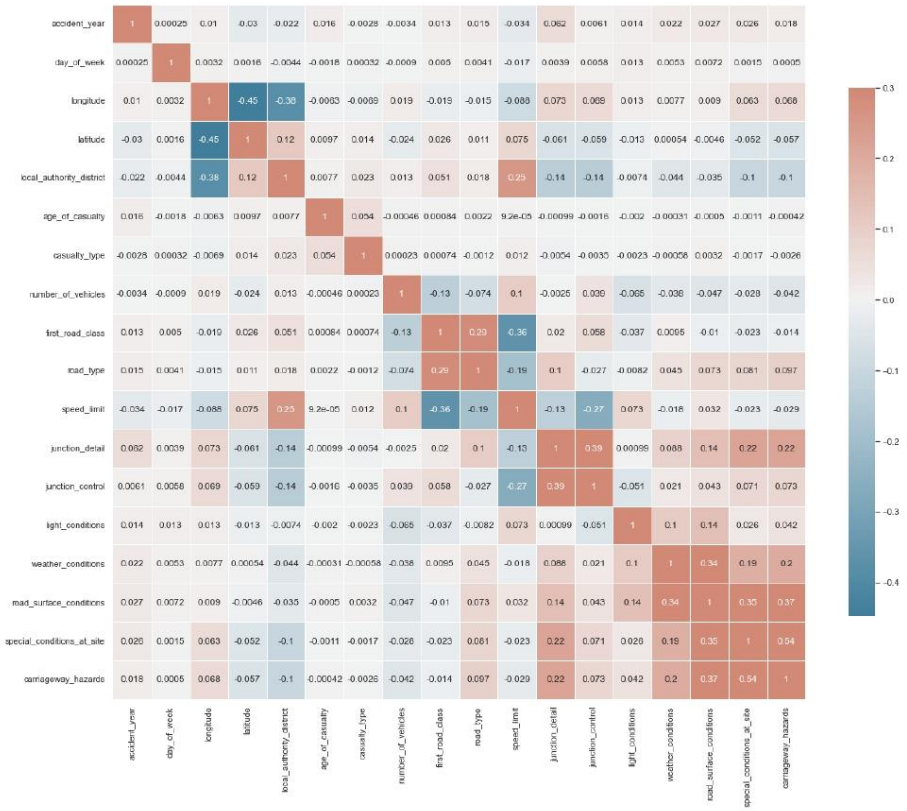
Out[75]:
```
array([0, 2, 6, 1, 9, 7, 3], dtype=int64)
```

In [76]:
```
sns.set(style="darkgrid")
plt.figure(figsize=(12,6))
roadplot = sns.countplot(x='carriageway_hazards',data=data,palette = "YlGnBu")
roadplot.set(xlabel='carriageway_hazards', ylabel='Count',xticklabels=['None','Ve
for p in roadplot.patches:
    height = p.get_height()
    roadplot.text(p.get_x()+p.get_width()/2.,
      height + 3,
      '{:1.2f}'.format((height/len(acc_time))*100)+'%',
      ha="center")
plt.title('Accidents to the carriageway hazards',fontsize=20)
plt.xticks(rotation=90,fontsize=15)
plt.show()
```

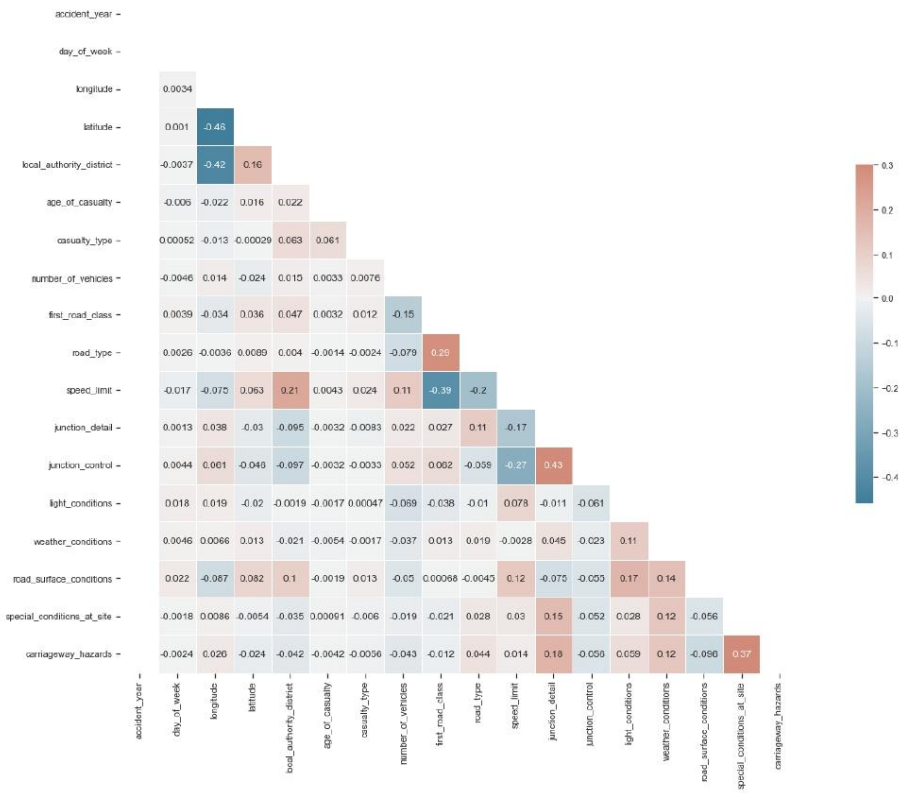


Inferential Statistics

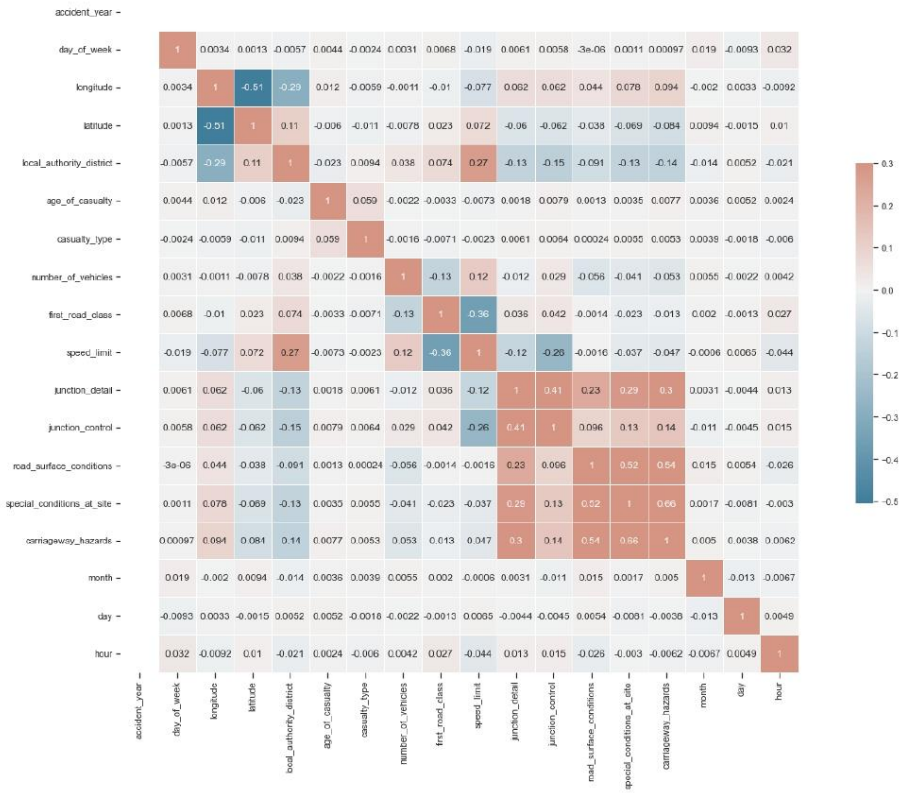
In [78]:
```
d1=data[['accident_year','date','day_of_week']]
correlation=data.corr()

mask=np.triu(np.ones_like(correlation, dtype=bool))

f,ax=plt.subplots(figsize=(20,25))
sns.set_style("white")
cmap=sns.diverging_palette(230,20,as_cmap=True)
heatmap=sns.heatmap(correlation,cmap=cmap,vmax=.3,center=0,
                    square=True,linewidth=.5,cbar_kws={"shrink":.5},
                    annot=True,annot_kws={"size":12})
sns.set_style({'xtick.bottom':True},{'ytick.left':True})
```



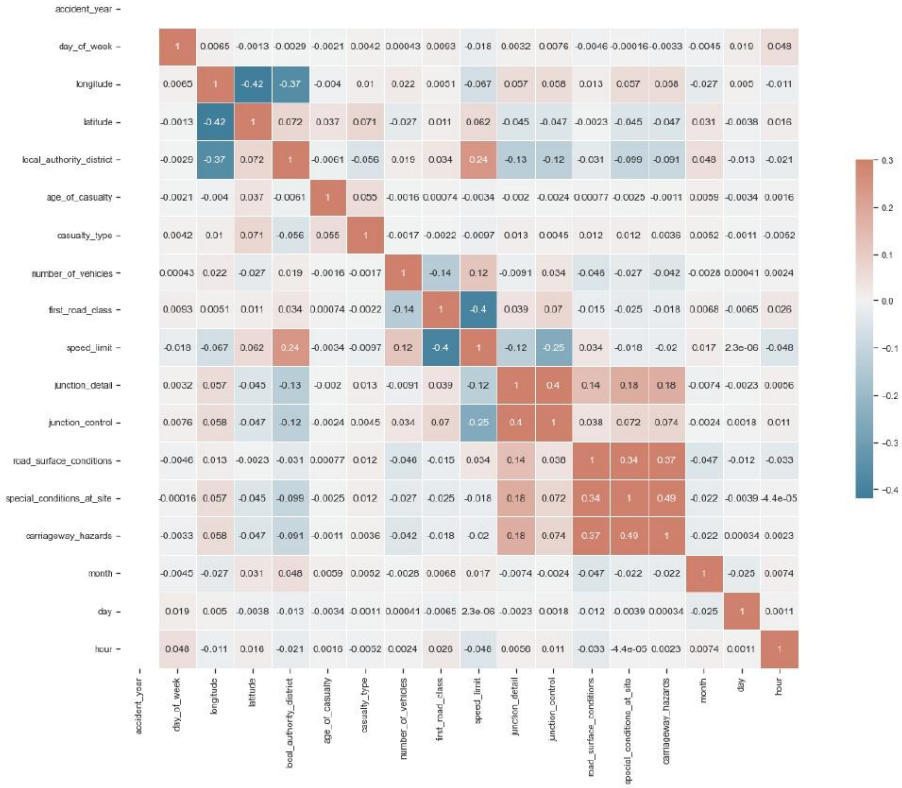
In [79]:
```
d1=data[data['accident_year']==2016]
correlation=d1.corr()

mask=np.triu(np.ones_like(correlation, dtype=bool))

f,ax=plt.subplots(figsize=(25,15))
sns.set_style("white")
cmap=sns.diverging_palette(230,20,as_cmap=True)
heatmap=sns.heatmap(correlation,mask=mask,cmap=cmap,vmax=.3,center=0,
                    square=True,linewidth=.5,cbar_kws={"shrink":.5},
                    annot=True,annot_kws={"size":12})
sns.set_style({'xtick.bottom':True},{'ytick.left':True})
heatmap.get_figure().savefig('/t_y_2016.png',bbox_inches='tight')
```



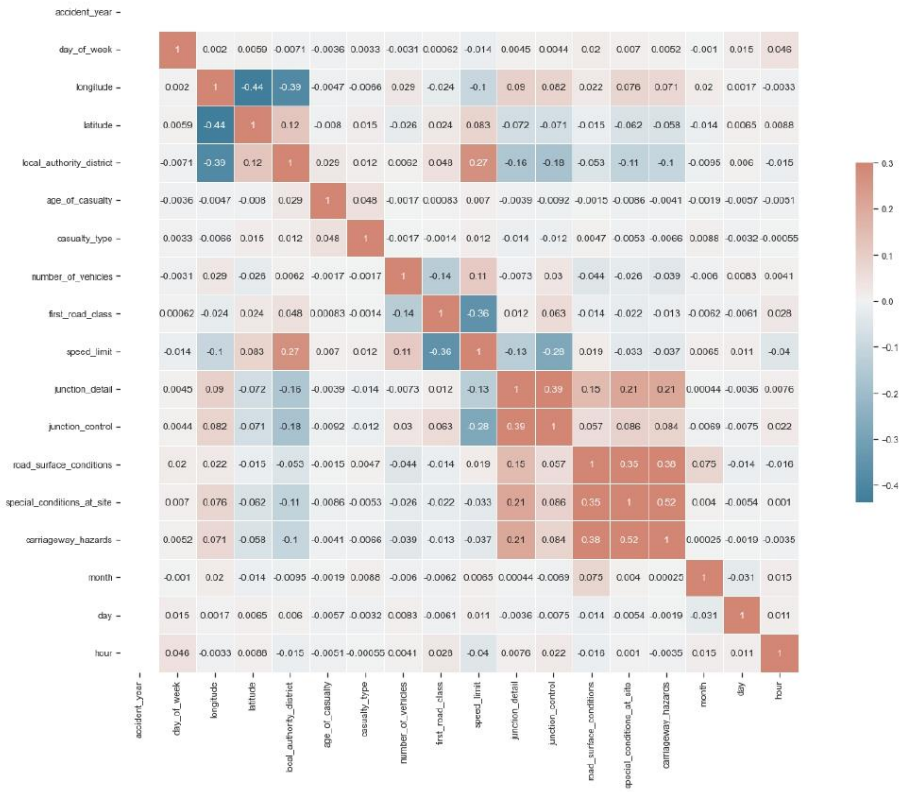
In [194]:
```
d1=data[data['accident_year']==2017]
correlation=d1.corr()

mask=np.triu(np.ones_like(correlation, dtype=bool))

f,ax=plt.subplots(figsize=(25,15))
sns.set_style("white")
cmap=sns.diverging_palette(230,20,as_cmap=True)
heatmap=sns.heatmap(correlation,cmap=cmap,vmax=.3,center=0,
                    square=True,linewidth=.5,cbar_kws={"shrink":.5},
                    annot=True,annot_kws={"size":12})
sns.set_style({'xtick.bottom':True},{'ytick.left':True})
heatmap.get_figure().savefig('/t_y_2016.png',bbox_inches='tight')
```



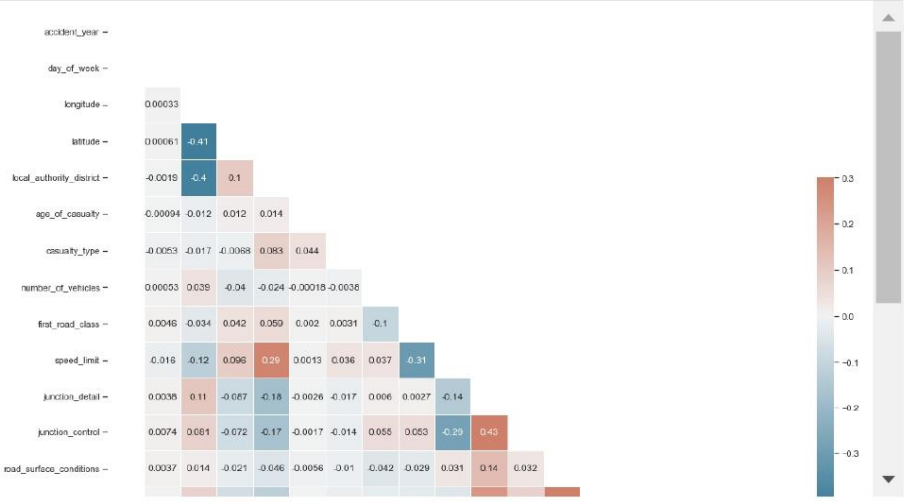
In [195]:
```
d1=data[data['accident_year']==2018]
correlation=d1.corr()

mask=np.triu(np.ones_like(correlation, dtype=bool))

f,ax=plt.subplots(figsize=(25,15))
sns.set_style("white")
cmap=sns.diverging_palette(230,20,as_cmap=True)
heatmap=sns.heatmap(correlation,cmap=cmap,vmax=.3,center=0,
                    square=True,linewidth=.5,cbar_kws={"shrink":.5},
                    annot=True,annot_kws={"size":12})
sns.set_style({'xtick.bottom':True},{'ytick.left':True})
heatmap.get_figure().savefig('/t_y_2016.png',bbox_inches='tight')
```



In [196]:
```
d1=data[data['accident_year']==2019]
correlation=d1.corr()

mask=np.triu(np.ones_like(correlation, dtype=bool))

f,ax=plt.subplots(figsize=(25,15))
sns.set_style("white")
cmap=sns.diverging_palette(230,20,as_cmap=True)
heatmap=sns.heatmap(correlation,cmap=cmap,vmax=.3,center=0,
                    square=True,linewidth=.5,cbar_kws={"shrink":.5},
                    annot=True,annot_kws={"size":12})
sns.set_style({'xtick.bottom':True},{'ytick.left':True})
heatmap.get_figure().savefig('/t_y_2016.png',bbox_inches='tight')
```



In [199]:
```
d1=data[data['accident_year']==2020]
correlation=d1.corr()

mask=np.triu(np.ones_like(correlation, dtype=bool))

f,ax=plt.subplots(figsize=(25,15))
sns.set_style("white")
cmap=sns.diverging_palette(230,20,as_cmap=True)
heatmap=sns.heatmap(correlation,mask=mask,cmap=cmap,vmax=.3,center=0,
                    square=True,linewidth=.5,cbar_kws={"shrink":.5},
                    annot=True,annot_kws={"size":12})
sns.set_style({'xtick.bottom':True},{'ytick.left':True})
heatmap.get_figure().savefig('/t_y_2016.png',bbox_inches='tight')
```



T-test

In [85]:
```
def year(string):
    return int(string.split('/')[-1])
data['Year']=data['date'].apply(lambda x: year(x))
#creating month column
```

In [101]:
```
data=data.drop(columns=['month'])
```

In [103]:
```
def monthh(string):
    return int(string.split('/')[1])
data['month']=data['date'].apply(lambda x: monthh(x))

def dayy(string):
    return int(string.split('/')[0])
data['day']=data['date'].apply(lambda x: dayy(x))

def hourr(string):
    return int(string.split(':')[0])
data['hour']=data['time'].apply(lambda x: hourr(x))
```

In [104]:
```
data.head()
```

Out[104]:

	accident_year	date	day_of_week	time	longitude	latitude	local_authority_district	casua
0	2016	1/11/2016	3	2:30	-0.279323	51.584754	28	
1	2016	1/11/2016	3	0:37	0.184928	51.449595	18	
2	2016	1/11/2016	3	1:25	-0.473837	51.543563	26	
3	2016	1/11/2016	3	9:15	-0.164442	51.404958	22	
4	2016	1/11/2016	3	7:53	-0.406580	51.483139	25	

5 rows × 27 columns

In [111]:
```
data['accident_severity'].unique()
```

Out[111]:
```
array([1, 2, 3], dtype=int64)
```

In [110]:
```
def acci(row):
    if row=='Slight':
        return 1
    elif row=='Serious':
        return 2
    else:
        return 3
data['accident_severity']=data['accident_severity'].apply(acci)
```

In [162]:
```
stats.levene(data[data['sex_of_casualty']=='Male']['accident_severity'],
                  data[data['sex_of_casualty']=='Female']['accident_severity'], c
```

Out[162]:
```
LeveneResult(statistic=0.7883350893860946, pvalue=0.3746044477196969)
```

In [135]:
```
s,p=stats.ttest_ind(data[data['sex_of_casualty']=='Male']['accident_severity'],
                    data[data['sex_of_casualty']=='Female']['accident_severity'], e
print(f'The value of P: {p}')
print(f'The statistic value: {s}')
if(p<0.05):
    print('The null hypothesis is rejected')
else:
    print('The null hypothesis is accepted')
```

```
The value of P: 0.6400610385731413
The statistic value: -0.46761373523828986
The null hypothesis is accepted
```

In [117]:
```
data['accident_year'].unique()
```

Out[117]:
```
array([2016, 2017, 2018, 2019, 2020], dtype=int64)
```

In [161]:
```
stats.levene(data[data['accident_year']==2016]['accident_severity'],
                    data[data['accident_year']==2017]['accident_severity'], center=
```

Out[161]:
```
LeveneResult(statistic=226.62063364437307, pvalue=3.443963820410245e-51)
```

In [118]:
```
stats.ttest_ind(data[data['accident_year']==2016]['accident_severity'],
                    data[data['accident_year']==2017]['accident_severity'], equal_v
```

Out[118]:
```
Ttest_indResult(statistic=-8.038239456798951, pvalue=9.15703115288173e-16)
```

In [119]:
```
stats.ttest_ind(data[data['accident_year']==2017]['accident_severity'],
                    data[data['accident_year']==2018]['accident_severity'], equal_v
```

Out[119]:
```
Ttest_indResult(statistic=-10.36268026717804, pvalue=3.7174962851662714e-25)
```

In [120]:
```
stats.ttest_ind(data[data['accident_year']==2018]['accident_severity'],
                    data[data['accident_year']==2019]['accident_severity'], equal_v
```

Out[120]:
```
Ttest_indResult(statistic=-4.838846637047749, pvalue=1.3068888238017492e-06)
```

In [122]:
```
stats.ttest_ind(data[data['accident_year']==2019]['accident_severity'],
                    data[data['accident_year']==2020]['accident_severity'], equal_v
```

Out[122]:
```
Ttest_indResult(statistic=-2.6144650622042755, pvalue=0.00893744476732243)
```

In [126]:
```
data['casualty_home_area_type'].unique()
```

Out[126]:
```
array(['Urban area', 'Rural area', 'Small Town'], dtype=object)
```

In [127]:
```
def xx(row):
    if row=='Urban area':
        return 1
    elif row=='Small Town':
        return 2
    else:
        return 3
data['casualty_home_area_type']=data['casualty_home_area_type'].apply(xx)
```

In [128]:
```
data['casualty_home_area_type'].unique()
```

Out[128]:
```
array([1, 3, 2], dtype=int64)
```

In [163]:
```
stats.levene(data[data['casualty_home_area_type']==1]['accident_severity'],
                  data[data['casualty_home_area_type']!=1]['accident_severity'],
```

Out[163]:
```
LeveneResult(statistic=22.653383166317866, pvalue=1.940708195826756e-06)
```

In [136]:
```
s,p=stats.ttest_ind(data[data['casualty_home_area_type']==1]['accident_severity']
                  data[data['casualty_home_area_type']!=1]['accident_severity'],
print(f'The value of P: {p}')
print(f'The statistic value: {s}')
if(p<0.05):
    print('The null hypothesis is rejected')
else:
    print('The null hypothesis is accepted')
```

```
The value of P: 0.006738855128995241
The statistic value: -2.709492034892519
The null hypothesis is rejected
```

In [130]:
```
data['age_of_casualty'].unique()
```

Out[130]:
```
array([ 23,  36,  24,  59,  28,  30,  33,  21,  53,  29,  11,  62,  34,
        20,  38,  32,  52,  79,  37,  61,  80,  74,  77,  58,  22,   2,
        51,  12,  83,  75,  40,  46,  56,  18,  39,  15,  45,  73,  57,
        48,  50,  81,  64,  16,  42,   0,  31,  88,  14,  43,  27,  25,
        68,  84,  35,  26,  76,  65,  69,  19,  44,  13,  49,  41,  89,
        71,  55,  17,  54,   9,  86,   3,  47,  60,  90,  66,  72,   5,
        91,   4,  10,   8,  82,  70,  67,   7,  63,  95,   6,  78,  85,
         1,  -1,  87,  93,  92,  96,  94,  97,  98, 101,  99, 100, 102],
      dtype=int64)
```

In [164]:
```
stats.levene(data[data['age_of_casualty']<=35]['accident_severity'],
                  data[data['age_of_casualty']>35]['accident_severity'], center='
```

Out[164]:
```
LeveneResult(statistic=25.25369513663393, pvalue=5.027995456612775e-07)
```

In [140]:
```
s,p=stats.ttest_ind(data[data['age_of_casualty']<=35]['accident_severity'],
                    data[data['age_of_casualty']>35]['accident_severity'], equal_va
print(f'The value of P: {p}')
print(f'The statistic value: {s}')
if(p<0.05):
    print('The null hypothesis is rejected')
else:
    print('The null hypothesis is accepted')
```

```
The value of P: 0.008774165212417218
The statistic value: -2.620740278507992
The null hypothesis is rejected
```

In [165]:
```
stats.levene(data[data['light_conditions']==1]['accident_severity'],
                    data[data['light_conditions']!=1]['accident_severity'], center=
```

Out[165]:
```
LeveneResult(statistic=3321.430172802838, pvalue=0.0)
```

In [141]:
```
s,p=stats.ttest_ind(data[data['light_conditions']==1]['accident_severity'],
                    data[data['light_conditions']!=1]['accident_severity'], equal_v
print(f'The value of P: {p}')
print(f'The statistic value: {s}')
if(p<0.05):
    print('The null hypothesis is rejected')
else:
    print('The null hypothesis is accepted')
```

```
The value of P: 2.7606889541413785e-190
The statistic value: -29.435737759427383
The null hypothesis is rejected
```

In [ ]:
```
stats.ttest_ind(data[data['road_surface_conditions']==1]['accident_severity'],
                    data[data['road_surface_conditions']!=1]['accident_severity'],
```

ANOVA

In [142]:
```
data['day_of_week'].unique()
```

Out[142]:
```
array([3, 4, 5, 6, 7, 1, 2], dtype=int64)
```

In [143]:
```
data['casualty_type'].unique()
```

Out[143]:
```
array([ 2,  9, 11,  1,  0,  4,  3, 19,  5,  8, 90, 21, 10, 20, 16, 97, 22,
       17, 98, 18, 23, 99, -1], dtype=int64)
```

In [145]:
```
f,p=stats.f_oneway(data[data['day_of_week']==1]['casualty_type'],data[data['day_c
              data[data['day_of_week']==3]['casualty_type'],data[data['day_of_wee
              data[data['day_of_week']==5]['casualty_type'],data[data['day_of_wee
              data[data['day_of_week']==7]['casualty_type'])
print(f'The value of P: {p}')
print(f'The statistic value: {f}')
if(p<0.05):
    print('The null hypothesis is rejected')
else:
    print('The null hypothesis is accepted')
```

```
The value of P: 0.6281527164942428
The statistic value: 0.7265783613118897
The null hypothesis is accepted
```

In [148]:
```
f,p=stats.f_oneway(data[data['accident_year']==2016]['accident_severity'],data[da
              data[data['accident_year']==2018]['accident_severity'],data[data['a
                  data[data['accident_year']==2020]['accident_severity'])
print(f'The value of P: {p}')
print(f'The statistic value: {f}')
if(p<0.05):
    print('The null hypothesis is rejected')
else:
    print('The null hypothesis is accepted')
```

```
The value of P: 4.6907876277626996e-198
The statistic value: 230.45587881291263
The null hypothesis is rejected
```

In [149]:
```
data['first_road_class'].unique()
```

Out[149]:
```
array([3, 5, 4, 6, 1, 2], dtype=int64)
```

In [151]:
```
f,p=stats.f_oneway(data[data['first_road_class']==1]['accident_severity'],data[da
              data[data['first_road_class']==3]['accident_severity'],data[data['f
                  data[data['first_road_class']==5]['accident_severity'],data[dat
print(f'The value of P: {p}')
print(f'The statistic value: {f}')
if(p<0.05):
    print('The null hypothesis is rejected')
else:
    print('The null hypothesis is accepted')
```

```
The value of P: 3.5696354847308164e-76
The statistic value: 72.52425586501744
The null hypothesis is rejected
```

In [152]:
```
data['junction_detail'].unique()
```

Out[152]:
```
array([ 0,  9,  1,  3,  7,  6,  2,  5,  8, 99, -1], dtype=int64)
```

In [156]:
```
f,p=stats.f_oneway(data[data['junction_detail']==1]['casualty_type'],data[data['j
              data[data['junction_detail']==3]['casualty_type'],
              data[data['junction_detail']==5]['casualty_type'],data[data['junct
              data[data['junction_detail']==7]['casualty_type'],data[data['junct
                  data[data['junction_detail']==9]['casualty_type'])
print(f'The value of P: {p}')
print(f'The statistic value: {f}')
if(p<0.05):
    print('The null hypothesis is rejected')
else:
    print('The null hypothesis is accepted')
```

```
The value of P: 0.03482884494257662
The statistic value: 2.156142083387309
The null hypothesis is rejected
```

In [157]:
```
data['road_surface_conditions'].unique()
```

Out[157]:
```
array([ 1,  2, -1,  5,  4,  3,  9], dtype=int64)
```

In [160]:
```
f,p=stats.f_oneway(data[data['road_surface_conditions']==1]['accident_severity'],
              data[data['road_surface_conditions']==3]['accident_severity'],data[
                  data[data['road_surface_conditions']==5]['accident_severity'],d
print(f'The value of P: {p}')
print(f'The statistic value: {f}')
if(p<0.05):
    print('The null hypothesis is rejected')
else:
    print('The null hypothesis is accepted')
```

```
The value of P: 1.2431654732826588e-95
The statistic value: 90.59292097244156
The null hypothesis is rejected
```

Chi-square test

In [170]:
```
def map_severity(severity):
    if severity == 1:
        return 'Fatal'
    elif severity == 2:
        return 'Serious'
    elif severity == 3:
        return 'Slight'

data['accident_severity'] = data['accident_severity'].apply(map_severity)
```

In [171]:
```
crosstab=pd.crosstab(data['sex_of_casualty'],data['accident_severity'])
crosstab
```

Out[171]:

accident_severity	Fatal	Serious	Slight
sex_of_casualty			
Female	168307	38023	2835
Male	246357	55464	4135

In [173]:
```
w,x,y,z=stats.chi2_contingency(crosstab)
print(f'The Chi square value is: {w}')
print(f'The p value is: {x}')
print(f'The value for degree of freedom: {y}')
print(f'Expected cell counts are: {z}')
```

```
The Chi square value is: 0.23302758311377192
The p value is: 0.8900178221606094
The value for degree of freedom: 2
Expected cell counts are: [[168374.41214783  37960.41775622   2830.17009596]
 [246289.58785217  55526.58224378   4139.82990404]]
```

In [176]:
```
def map_area(row):
    if row == 1:
        return 'Urban Area'
    elif row == 2:
        return 'Small Town'
    elif row == 3:
        return 'Rural area'

data['casualty_home_area_type'] = data['casualty_home_area_type'].apply(map_area)
```

In [177]:
```
crosstab=pd.crosstab(data['casualty_home_area_type'],data['accident_severity'])
crosstab
```

Out[177]:

accident_severity	Fatal	Serious	Slight
casualty_home_area_type			
Rural area	41915	9717	668
Small Town	34073	7921	570
Urban Area	338676	75849	5732

In [178]:
```
w,x,y,z=stats.chi2_contingency(crosstab)
print(f'The Chi square value is: {w}')
print(f'The p value is: {x}')
print(f'The value for degree of freedom: {y}')
print(f'Expected cell counts are: {z}')
```

```
The Chi square value is: 17.60692688482232
The p value is: 0.0014725970936065745
The value for degree of freedom: 4
Expected cell counts are: [[ 42100.64664419   9491.69243731    707.6609185 ]
 [ 34263.32550216   7724.74946275    575.92503509]
 [338300.02785365  76270.55809994   5686.41404641]]
```

In [180]:
```
def map_roadt(row):
    if row == 1:
        return 'Roundabout'
    elif row == 2:
        return 'One way street'
    elif row == 3:
        return 'Duel carriageway'
    elif row == 6:
        return 'Single carriageway'
    elif row == 7:
        return 'Slip road'
    elif row == 12:
        return 'One way street/slip road'

data['road_type'] = data['road_type'].apply(map_roadt)
```

In [181]:
```
crosstab=pd.crosstab(data['road_type'],data['accident_severity'])
crosstab
```

Out[181]:

accident_severity	Fatal	Serious	Slight
road_type			
Duel carriageway	64770	12897	1382
One way street	10374	1911	74
Roundabout	27799	4201	116
Single carriageway	297728	72703	5295
Slip road	5872	916	74

In [182]:
```
w,x,y,z=stats.chi2_contingency(crosstab)
print(f'The Chi square value is: {w}')
print(f'The p value is: {x}')
print(f'The value for degree of freedom: {y}')
print(f'Expected cell counts are: {z}')
```

```
The Chi square value is: 1683.1758699305292
The p value is: 0.0
The value for degree of freedom: 8
Expected cell counts are: [[6.34974425e+04 1.44674514e+04 1.08410610e+03]
 [9.92757519e+03 2.26192908e+03 1.69495722e+02]
 [2.57977187e+04 5.87783109e+03 4.40450248e+02]
 [3.01808246e+05 6.87649136e+04 5.15284002e+03]
 [5.51201723e+03 1.25587486e+03 9.41079089e+01]]
```

In [184]:
```
def map_weather(row):
    if row == 1:
        return 'Fine'
    elif row == 2:
        return 'Raining'
    elif row == 3:
        return 'Snowing'
    elif row == 4:
        return 'Fine + high winds'
    elif row == 5:
        return 'Raining + high winds'
    elif row == 6:
        return 'Snowing + High winds'
    elif row == 7:
        return 'Foggy'


data['weather_conditions'] = data['weather_conditions'].apply(map_weather)
```

In [185]:
```
crosstab=pd.crosstab(data['weather_conditions'],data['accident_severity'])
crosstab
```

Out[185]:

accident_severity	Fatal	Serious	Slight
weather_conditions			
Fine	330943	76477	5726
Fine + high winds	4198	1107	122
Foggy	1819	462	80
Raining	47853	10198	686
Raining + high winds	4554	1227	116
Snowing	1802	330	25
Snowing + High winds	491	97	3

In [186]:
```
w,x,y,z=stats.chi2_contingency(crosstab)
print(f'The Chi square value is: {w}')
print(f'The p value is: {x}')
print(f'The value for degree of freedom: {y}')
print(f'Expected cell counts are: {z}')
```

```
The Chi square value is: 248.73666005702427
The p value is: 2.5093383165054407e-46
The value for degree of freedom: 12
Expected cell counts are: [[3.31368954e+05 7.60593532e+04 5.71769237e+03]
 [4.35279372e+03 9.99099857e+02 7.51064188e+01]
 [1.89366980e+03 4.34655383e+02 3.26748212e+01]
 [4.71107509e+04 1.08133644e+04 8.12884784e+02]
 [4.72976315e+03 1.08562592e+03 8.16109364e+01]
 [1.73004903e+03 3.97099391e+02 2.98515838e+01]
 [4.74018996e+02 1.08801919e+02 8.17908485e+00]]
```

In [188]:
```
def map_light(row):
    if row == 1:
        return 'Daylight'
    elif row == 4:
        return 'Darkness - lights lit'
    elif row == 5:
        return 'Darkness - lights unlit'
    elif row == 6:
        return 'Darkness - no lighting'
    elif row == 7:
        return 'Darkness - lighting unknown'


data['light_conditions'] = data['light_conditions'].apply(map_light)
```

In [189]:
```
crosstab=pd.crosstab(data['light_conditions'],data['accident_severity'])
crosstab
```

Out[189]:

accident_severity	Fatal	Serious	Slight
light_conditions			
Darkness - lighting unknown	9000	1529	120
Darkness - lights lit	83902	19996	1426
Darkness - lights unlit	2897	736	77
Darkness - no lighting	18489	6615	1252
Daylight	300376	64611	4095

In [190]:
```
w,x,y,z=stats.chi2_contingency(crosstab)
print(f'The Chi square value is: {w}')
print(f'The p value is: {x}')
print(f'The value for degree of freedom: {y}')
print(f'Expected cell counts are: {z}')
```

```
The Chi square value is: 3764.268990542102
The p value is: 0.0
The value for degree of freedom: 8
Expected cell counts are: [[8.57227125e+03 1.93263925e+03 1.44089505e+02]
 [8.47841015e+04 1.91147804e+04 1.42511814e+03]
 [2.98648947e+03 6.73311261e+02 5.01992736e+01]
 [2.12161500e+04 4.78323223e+03 3.56617804e+02]
 [2.97104988e+05 6.69830369e+04 4.99397528e+03]]
```

In [192]:
```
crosstab=pd.crosstab(data['speed_limit'],data['accident_severity'])
crosstab
```

Out[192]:

accident_severity	Fatal	Serious	Slight
speed_limit			
-1.0	69	6	0
20.0	36105	6649	201
30.0	255624	52167	2338
40.0	34366	8379	690
50.0	16445	4190	550
60.0	46144	16639	2362
70.0	25911	5457	829

In [193]:
```
w,x,y,z=stats.chi2_contingency(crosstab)
print(f'The Chi square value is: {w}')
print(f'The p value is: {x}')
print(f'The value for degree of freedom: {y}')
print(f'Expected cell counts are: {z}')
```

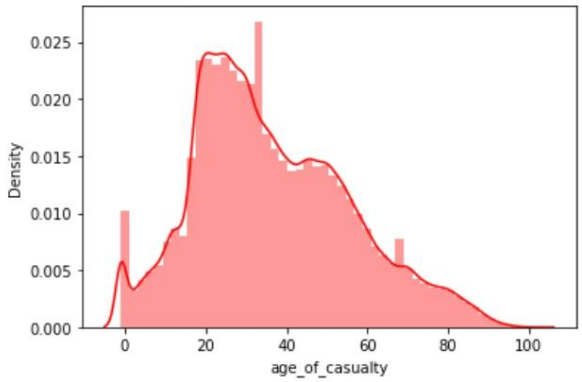
```
The Chi square value is: 7697.734253513967
The p value is: 0.0
The value for degree of freedom: 12
Expected cell counts are: [[6.03737763e+01 1.36114136e+01 1.01481011e+00]
 [3.45780741e+04 7.79571030e+03 5.81215578e+02]
 [2.49648785e+05 5.62839213e+04 4.19629394e+03]
 [3.49644663e+04 7.88282335e+03 5.87710363e+02]
 [1.70535793e+04 3.84477064e+03 2.86650030e+02]
 [5.24406621e+04 1.18228739e+04 8.81464064e+02]
 [2.59180597e+04 5.84328913e+03 4.35651216e+02]]
```

Machine learning

In [118]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC, LinearSVC
from sklearn.metrics import log_loss
from sklearn import svm
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense,LSTM, Dropout,Flatten
import keras
from keras.layers import GRU
from tensorflow.keras.optimizers import Adam
from keras.models import Model, Sequential, Input, load_model
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from keras.layers.convolutional import Conv1D
from keras.layers.convolutional import MaxPooling1D
```

In [153]:
```
sns.distplot(data['age_of_casualty'], color='red');
fig = plt.figure()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Futur
eWarning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level function
with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
```




```
<Figure size 432x288 with 0 Axes>
```

In [154]:
```
data['age_of_casualty'].unique()
```

Out[154]:
```
array([ 23,  36,  24,  59,  28,  30,  33,  21,   7,  53,  29,  11,  62,
        34,  20,  38,  32,  52,  79,  37,  61,  80,  74,  77,  58,  22,
         2,  31,  51,  12,  83,  75,  89,  40,  46,   3,  56,  18,  39,
        15,  45,  73,  57,  48,  50,  81,  64,  16,  42,   0,  88,  26,
        14,  43,  27,  25,  68,  84,  35,  76,  65,  69,  19,  44,  13,
        49,  41,  71,  55,  17,  54,   9,  86,  47,  60,  90,  66,  72,
         5,  91,   4,  67,  10,   8,  82,  70,  -1,  63,  95,   6,  78,
        85,   1,  87,  93,  92,  96,  94,  97,  98, 101,  99, 100, 102],
      dtype=int64)
```

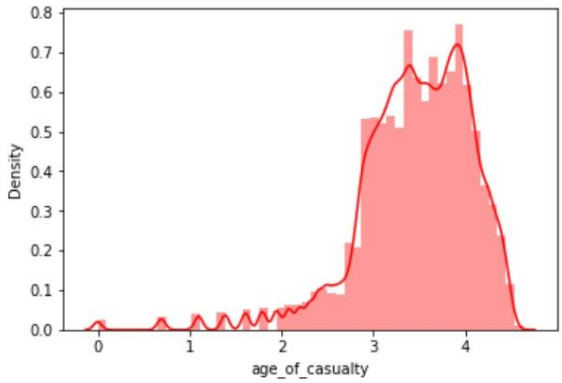
In [155]:
```
data['age_of_casualty'] = np.log(data['age_of_casualty'])
data=data[data['age_of_casualty']>=0]
data['age_of_casualty'].unique()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\arraylike.py:364: Runtim
eWarning: divide by zero encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\arraylike.py:364: Runtim
eWarning: invalid value encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
C:\Users\mdasi\AppData\Local\Temp/ipykernel_4124/469062920.py:1: SettingWithCop
yWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
  data['age_of_casualty'] = np.log(data['age_of_casualty'])
```

Out[155]:
```
array([3.13549422, 3.58351894, 3.17805383, 4.07753744, 3.33220451,
       3.40119738, 3.49650756, 3.04452244, 1.94591015, 3.97029191,
       3.36729583, 2.39789527, 4.12713439, 3.52636052, 2.99573227,
       3.63758616, 3.4657359 , 3.95124372, 4.36944785, 3.61091791,
       4.11087386, 4.38202663, 4.30406509, 4.34380542, 4.06044301,
       3.09104245, 0.69314718, 3.4339872 , 3.93182563, 2.48490665,
       4.41884061, 4.31748811, 4.48863637, 3.68887945, 3.8286414 ,
       1.09861229, 4.02535169, 2.89037176, 3.66356165, 2.7080502 ,
       3.80666249, 4.29045944, 4.04305127, 3.87120101, 3.91202301,
       4.39444915, 4.15888308, 2.77258872, 3.73766962, 4.47733681,
       3.25809654, 2.63905733, 3.76120012, 3.29583687, 3.21887582,
       4.21950771, 4.4308168 , 3.55534806, 4.33073334, 4.17438727,
       4.2341065 , 2.94443898, 3.78418963, 2.56494936, 3.8918203 ,
       3.71357207, 4.26267988, 4.00733319, 2.83321334, 3.98898405,
       2.19722458, 4.4543473 , 3.8501476 , 4.09434456, 4.49980967,
       4.18965474, 4.27666612, 1.60943791, 4.51085951, 1.38629436,
       4.20469262, 2.30258509, 2.07944154, 4.40671925, 4.24849524,
       4.14313473, 4.55387689, 1.79175947, 4.35670883, 4.44265126,
       0.        , 4.46590812, 4.53259949, 4.52178858, 4.56434819,
       4.54329478, 4.57471098, 4.58496748, 4.61512052, 4.59511985,
       4.60517019, 4.62497281])
```

In [156]:
```
sns.distplot(data['age_of_casualty'],color='red')
fig = plt.figure()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Futur
eWarning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level function
with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
```




```
<Figure size 432x288 with 0 Axes>
```

In [27]:
```
data.isnull().sum()
```

Out[27]:
```
accident_year                  0
date                           0
day_of_week                    0
time                           0
longitude                    130
latitude                     130
local_authority_district       0
casualty_home_area_type        0
sex_of_casualty                0
age_of_casualty                0
age_band_of_casualty           0
accident_severity              0
casualty_type                  0
number_of_vehicles             0
first_road_class               0
road_type                      0
speed_limit                   34
junction_detail                0
junction_control               0
light_conditions               0
weather_conditions             0
road_surface_conditions        0
special_conditions_at_site     0
carriageway_hazards            0
dtype: int64
```

In [212]:
```
accident_ml = data.drop('accident_severity' ,axis=1)
accident_ml = accident_ml[['day_of_week','casualty_home_area_type',
        'sex_of_casualty','age_of_casualty', 'junction_control',
        'light_conditions','weather_conditions', 'road_surface_conditions', 'carr

# Split the data into a training and test set.
X_train, X_test, y_train, y_test = train_test_split(accident_ml.values,
                                                    data['accident_severity'].values,te
```

Random forest

In [129]:
```
random_forest = RandomForestClassifier(n_estimators=200,verbose=1)
random_forest.fit(X_train,y_train)
Y_pred = random_forest.predict(X_test)
random_forest.score(X_test, y_test)
acc_random_forest1 = round(random_forest.score(X_test, y_test) * 100, 2)

sk_report = classification_report(
    digits=6,
    y_true=y_test,
    y_pred=Y_pred)
print("Accuracy" , acc_random_forest1)
print(sk_report)
pd.crosstab(y_test, Y_pred, rownames=['Actual'], colnames=['Predicted'], margins=
```

```
C:\Users\mdasi\AppData\Local\Temp/ipykernel_4124/1546269827.py:2: DataConversio
nWarning: A column-vector y was passed when a 1d array was expected. Please cha
nge the shape of y to (n_samples,), for example using ravel().
  random_forest.fit(X_train,y_train)
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent worker
s.
[Parallel(n_jobs=1)]: Done 200 out of 200 | elapsed:  9.9min finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent worker
s.
[Parallel(n_jobs=1)]: Done 200 out of 200 | elapsed:    6.6s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent worker
s.
[Parallel(n_jobs=1)]: Done 200 out of 200 | elapsed:    6.7s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent worker
s.
[Parallel(n_jobs=1)]: Done 200 out of 200 | elapsed:    6.7s finished


Accuracy 80.08
              precision    recall  f1-score   support

           1   0.043478  0.001257  0.002443      1591
           2   0.203390  0.009054  0.017336     21206
           3   0.805942  0.991885  0.889298     94519

    accuracy                       0.800794    117316
   macro avg   0.350937  0.334065  0.303026    117316
weighted avg   0.686685  0.800794  0.719655    117316
```

Out[129]:

Predicted	1	2	3	All
Actual				
1	2	21	1568	1591
2	8	192	21006	21206
3	36	731	93752	94519
All	46	944	116326	117316

Logistic regression

In [299]:
```
lr = LogisticRegression()
# Fit the model on the trainng data.
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)
sk_report = classification_report(
    digits=6,
    y_true=y_test,
    y_pred=y_pred)
print("Accuracy", round(accuracy_score(y_pred, y_test)*100,2))
print(sk_report)
pd.crosstab(y_test, y_pred, rownames=['Actual'], colnames=['Predicted'], margins=
```

```
Accuracy 80.61
              precision    recall  f1-score   support

           1   0.000000  0.000000  0.000000      1558
           2   0.000000  0.000000  0.000000     21072
           3   0.806147  1.000000  0.892670     94108

    accuracy                       0.806147    116738
   macro avg   0.268716  0.333333  0.297557    116738
weighted avg   0.649873  0.806147  0.719624    116738


C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:76
3: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-
learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on)
  n_iter_i = _check_optimize_result(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1
248: UndefinedMetricWarning: Precision and F-score are ill-defined and being se
t to 0.0 in labels with no predicted samples. Use `zero_division` parameter to
 control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1
248: UndefinedMetricWarning: Precision and F-score are ill-defined and being se
t to 0.0 in labels with no predicted samples. Use `zero_division` parameter to
 control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1
248: UndefinedMetricWarning: Precision and F-score are ill-defined and being se
t to 0.0 in labels with no predicted samples. Use `zero_division` parameter to
 control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Out[299]:

Predicted	3	All
Actual		
1	1558	1558

Predicted	3	All
Actual		
2	21072	21072
3	94108	94108
All	116738	116738

In [125]:
```
from sklearn.linear_model import LogisticRegressionCV
lr = LogisticRegressionCV(cv=3, random_state=0,multi_class='multinomial')
# Fit the model on the trainng data.
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)
sk_report = classification_report(
    digits=6,
    y_true=y_test,
    y_pred=y_pred)
print("Accuracy", round(accuracy_score(y_pred, y_test)*100,2))
print(sk_report)
pd.crosstab(y_test, y_pred, rownames=['Actual'], colnames=['Predicted'], margins=
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:63: Data
ConversionWarning: A column-vector y was passed when a 1d array was expected. P
lease change the shape of y to (n_samples, ), for example using ravel().
  return f(*args, **kwargs)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1
248: UndefinedMetricWarning: Precision and F-score are ill-defined and being se
t to 0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1
248: UndefinedMetricWarning: Precision and F-score are ill-defined and being se
t to 0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1
248: UndefinedMetricWarning: Precision and F-score are ill-defined and being se
t to 0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))

Accuracy 80.57
              precision    recall  f1-score   support

           1   0.000000  0.000000  0.000000      1591
           2   0.000000  0.000000  0.000000     21206
           3   0.805679  1.000000  0.892383     94519

    accuracy                       0.805679    117316
   macro avg   0.268560  0.333333  0.297461    117316
weighted avg   0.649118  0.805679  0.718974    117316
```

Out[125]:

Predicted	3	All
Actual		
1	1591	1591
2	21206	21206
3	94519	94519
All	117316	117316

In [300]:
```
decision_tree = DecisionTreeClassifier()
decision_tree.fit(X_train, y_train)
Y_pred = decision_tree.predict(X_test)
acc_decision_tree1 = round(decision_tree.score(X_test, y_test) * 100, 2)
sk_report = classification_report(
    digits=6,
    y_true=y_test,
    y_pred=Y_pred)
print("Accuracy", acc_decision_tree1)
print(sk_report)
### Confusion Matrix
pd.crosstab(y_test, Y_pred, rownames=['Actual'], colnames=['Predicted'], margins=
```

```
Accuracy 75.48
              precision    recall  f1-score   support

           1   0.035088  0.016688  0.022619      1558
           2   0.191719  0.083286  0.116125     21072
           3   0.808008  0.917350  0.859214     94108

    accuracy                       0.754776    116738
   macro avg   0.344938  0.339108  0.332653    116738
weighted avg   0.686448  0.754776  0.713916    116738
```

Out[300]:

Predicted	1	2	3	All
Actual				
1	26	166	1366	1558
2	170	1755	19147	21072
3	545	7233	86330	94108
All	741	9154	106843	116738

In [126]:
```
decision_tree = DecisionTreeClassifier(min_samples_leaf=12, max_features=4)
decision_tree.fit(X_train, y_train)
Y_pred = decision_tree.predict(X_test)
acc_decision_tree1 = round(decision_tree.score(X_test, y_test) * 100, 2)
sk_report = classification_report(
    digits=6,
    y_true=y_test,
    y_pred=Y_pred)
print("Accuracy", acc_decision_tree1)
print(sk_report)
### Confusion Matrix
pd.crosstab(y_test, Y_pred, rownames=['Actual'], colnames=['Predicted'], margins=
```

```
Accuracy 80.57
              precision    recall  f1-score   support

           1   0.000000  0.000000  0.000000      1591
           2   0.481481  0.000613  0.001225     21206
           3   0.805745  0.999852  0.892365     94519

    accuracy                       0.805670    117316
   macro avg   0.429075  0.333488  0.297863    117316
weighted avg   0.736204  0.805670  0.719181    117316


C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1
248: UndefinedMetricWarning: Precision and F-score are ill-defined and being se
t to 0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1
248: UndefinedMetricWarning: Precision and F-score are ill-defined and being se
t to 0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1
248: UndefinedMetricWarning: Precision and F-score are ill-defined and being se
t to 0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Out[126]:

Predicted	2	3	All
Actual			
1	0	1591	1591
2	13	21193	21206
3	14	94505	94519
All	27	117289	117316

In [33]:
```
knn_clf=KNeighborsClassifier()
knn_clf.fit(X_train, y_train)
Y_pred = knn_clf.predict(X_test)
acc_decision_tree1 = round(knn_clf.score(X_test, y_test) * 100, 2)
sk_report = classification_report(
    digits=6,
    y_true=y_test,
    y_pred=Y_pred)
print("Accuracy", acc_decision_tree1)
print(sk_report)
### Confusion Matrix
pd.crosstab(y_test, Y_pred, rownames=['Actual'], colnames=['Predicted'], margins=
```

```
Accuracy 77.17
              precision    recall  f1-score   support

           1   0.051020  0.006285  0.011192      1591
           2   0.200181  0.062671  0.095457     21206
           3   0.807315  0.943652  0.870176     94519

    accuracy                       0.771694    117316
   macro avg   0.352839  0.337536  0.325608    117316
weighted avg   0.687313  0.771694  0.718488    117316
```

Out[33]:

Predicted	1	2	3	All
Actual				
1	10	117	1464	1591
2	53	1329	19824	21206
3	133	5193	89193	94519
All	196	6639	110481	117316

Deep learning

In [215]:
```
#LSTM


xs=MinMaxScaler()
ys=MinMaxScaler()
y_train=y_train.reshape(-1, 1)
s_xtr=xs.fit_transform(X_train)
s_ytr=ys.fit_transform(y_train)
s_xte=xs.fit_transform(X_test)

s_xtr, s_ytr=np.array(s_xtr), np.array(s_ytr)

sx=np.reshape(s_xtr,(s_xtr.shape[0],s_xtr.shape[1],1))
sy=np.reshape(s_ytr,(s_ytr.shape[0],s_ytr.shape[1],1))
sx=sx[:200]
sy=sy[:200]
```

In [92]:
```
model_lstm=Sequential()

model_lstm.add(LSTM(units=100, return_sequences=False,input_shape=(sx.shape[1],1)
model_lstm.add(Dropout(0.2))
model_lstm.add(Dense(units=1))

optimizer=Adam(lr=0.0001)
model_lstm.compile(optimizer=optimizer, loss='mean_squared_error', metrics=['accu
lstm_history=model_lstm.fit(sx,sy,epochs=20, batch_size=4, shuffle=True,validatio
```

```
Epoch 1/20

C:\ProgramData\Anaconda3\lib\site-packages\keras\optimizer_v2\adam.py:105: User
Warning: The `lr` argument is deprecated, use `learning_rate` instead.
  super(Adam, self).__init__(name, **kwargs)

40/40 [==============================] - 6s 117ms/step - loss: 0.7535 - accurac
y: 0.0188 - val_loss: 0.5825 - val_accuracy: 0.0250
Epoch 2/20
40/40 [==============================] - 5s 113ms/step - loss: 0.4643 - accurac
y: 0.0250 - val_loss: 0.2363 - val_accuracy: 0.1000
Epoch 3/20
40/40 [==============================] - 4s 112ms/step - loss: 0.1071 - accurac
y: 0.7625 - val_loss: 0.0674 - val_accuracy: 0.7750
Epoch 4/20
40/40 [==============================] - 4s 110ms/step - loss: 0.0697 - accurac
y: 0.8250 - val_loss: 0.0654 - val_accuracy: 0.7750
Epoch 5/20
40/40 [==============================] - 4s 111ms/step - loss: 0.0672 - accurac
y: 0.8250 - val_loss: 0.0722 - val_accuracy: 0.7750
Epoch 6/20
40/40 [==============================] - 4s 112ms/step - loss: 0.0699 - accurac
y: 0.8250 - val_loss: 0.0680 - val_accuracy: 0.7750
Epoch 7/20
40/40 [==============================] - 4s 110ms/step - loss: 0.0654 - accurac
y: 0.8250 - val_loss: 0.0708 - val_accuracy: 0.7750
Epoch 8/20
40/40 [==============================] - 5s 114ms/step - loss: 0.0699 - accurac
y: 0.8250 - val_loss: 0.0660 - val_accuracy: 0.7750
Epoch 9/20
40/40 [==============================] - 4s 110ms/step - loss: 0.0663 - accurac
y: 0.8250 - val_loss: 0.0644 - val_accuracy: 0.7750
Epoch 10/20
40/40 [==============================] - 4s 109ms/step - loss: 0.0637 - accurac
y: 0.8250 - val_loss: 0.0690 - val_accuracy: 0.7750
Epoch 11/20
40/40 [==============================] - 4s 110ms/step - loss: 0.0596 - accurac
y: 0.8250 - val_loss: 0.0739 - val_accuracy: 0.7750
Epoch 12/20
40/40 [==============================] - 4s 109ms/step - loss: 0.0655 - accurac
y: 0.8250 - val_loss: 0.0708 - val_accuracy: 0.7750
Epoch 13/20
40/40 [==============================] - 4s 111ms/step - loss: 0.0640 - accurac
y: 0.8250 - val_loss: 0.0690 - val_accuracy: 0.7750
Epoch 14/20
40/40 [==============================] - 5s 115ms/step - loss: 0.0625 - accurac
```

```
y: 0.8250 - val_loss: 0.0634 - val_accuracy: 0.7750
Epoch 15/20
40/40 [==============================] - 5s 113ms/step - loss: 0.0623 - accurac
y: 0.8250 - val_loss: 0.0637 - val_accuracy: 0.7750
Epoch 16/20
40/40 [==============================] - 4s 109ms/step - loss: 0.0643 - accurac
y: 0.8250 - val_loss: 0.0638 - val_accuracy: 0.7750
Epoch 17/20
40/40 [==============================] - 5s 113ms/step - loss: 0.0592 - accurac
y: 0.8250 - val_loss: 0.0629 - val_accuracy: 0.7750
Epoch 18/20
40/40 [==============================] - 4s 112ms/step - loss: 0.0611 - accurac
y: 0.8250 - val_loss: 0.0627 - val_accuracy: 0.7750
Epoch 19/20
40/40 [==============================] - 4s 111ms/step - loss: 0.0617 - accurac
y: 0.8250 - val_loss: 0.0634 - val_accuracy: 0.7750
Epoch 20/20
40/40 [==============================] - 4s 111ms/step - loss: 0.0603 - accurac
y: 0.8250 - val_loss: 0.0629 - val_accuracy: 0.7750
```

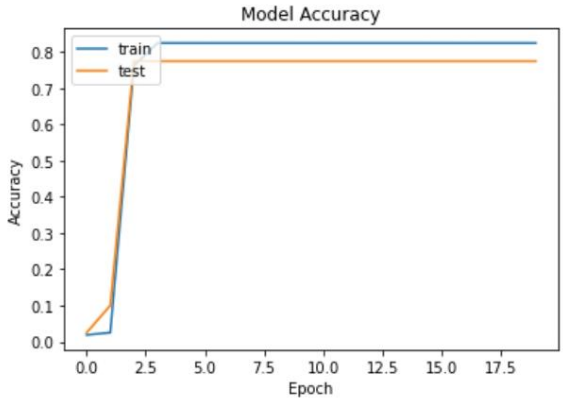
In [210]:
```
print(model_lstm.summary())
```

```
Model: "sequential_9"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_8 (LSTM)               (None, 100)               40800

 dropout_7 (Dropout)         (None, 100)               0

 dense_8 (Dense)             (None, 1)                 101

=================================================================
Total params: 40,901
Trainable params: 40,901
Non-trainable params: 0
_________________________________________________________________
None
```

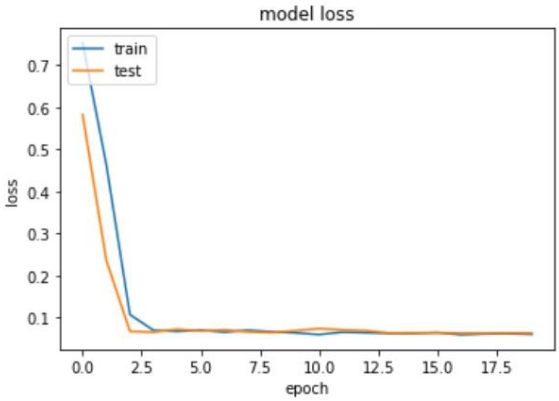
In [95]:
```
xte=np.array(s_xte)

xte=np.reshape(xte,(xte.shape[0],xte.shape[1],1))

xte=xte[:200]
pred=model_lstm.predict(xte)
xpred=model_lstm.predict(sx)

pred=ys.inverse_transform(pred)
xpred=ys.inverse_transform(xpred)

rmse_te=np.sqrt(np.mean((pred-y_test[:200])**2))
rmse_tr=np.sqrt(np.mean((xpred-y_train[:200])**2))
print('Trianing RMSE: ',rmse_tr)
print('Test RMSE: ',rmse_te)
print(lstm_history.history.keys())
print('\n')
```

```
Trianing RMSE:  0.4870832400368943
Test RMSE:  0.46496676958919914
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

In [96]:
```
plt.plot(lstm_history.history['accuracy'])
plt.plot(lstm_history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['train','test'],loc='upper left')
plt.show()
```



In [97]:
```
plt.plot(lstm_history.history['loss'])
plt.plot(lstm_history.hiastory['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



In [124]:
```
final_loss, final_accuracy = model_lstm.evaluate(xte, yte[:200])
print('Final Loss: {}, Final Accuracy: {}'.format(final_loss, final_accuracy))
```

```
7/7 [==============================] - 1s 87ms/step - loss: 0.0551 - accuracy:
0.8400
Final Loss: 0.055057309567928314, Final Accuracy: 0.8399999737739563
```

In [ ]:
```
### CNN
```

In [216]:
```
xtrain,ytrain=np.array(X_train),np.array(y_train)
sx2=np.reshape(xtrain,(xtrain.shape[0],xtrain.shape[1],1))
sy2=np.reshape(ytrain,(ytrain.shape[0],ytrain.shape[1],1))
```

In [217]:
```
modelc=Sequential()
modelc.add(Conv1D(filters=64,kernel_size=2, activation='relu',input_shape=(sx2.sh
modelc.add(MaxPooling1D(pool_size=2))
modelc.add(Flatten())
modelc.add(Dense(100,activation='relu'))
modelc.add(Dense(3))
modelc.compile(optimizer=optimizer,loss='mse',metrics=['accuracy'])
cnn_history=modelc.fit(sx2[:200],sy2[:200],epochs=20,verbose=1, validation_split=
```

```
Epoch 1/20
5/5 [==============================] - 0s 35ms/step - loss: 7.9971 - accurac
y: 0.2375 - val_loss: 6.8647 - val_accuracy: 0.1500
Epoch 2/20
5/5 [==============================] - 0s 9ms/step - loss: 4.7821 - accuracy:
0.1875 - val_loss: 3.9240 - val_accuracy: 0.0250
Epoch 3/20
5/5 [==============================] - 0s 7ms/step - loss: 2.4095 - accuracy:
0.0500 - val_loss: 1.9726 - val_accuracy: 0.0000e+00
Epoch 4/20
5/5 [==============================] - 0s 8ms/step - loss: 1.2079 - accuracy:
0.0000e+00 - val_loss: 0.9541 - val_accuracy: 0.0000e+00
Epoch 5/20
5/5 [==============================] - 0s 7ms/step - loss: 0.8031 - accuracy:
0.0000e+00 - val_loss: 0.5944 - val_accuracy: 0.0000e+00
Epoch 6/20
5/5 [==============================] - 0s 10ms/step - loss: 0.7945 - accurac
y: 0.0437 - val_loss: 0.5170 - val_accuracy: 0.0500
Epoch 7/20
5/5 [==============================] - 0s 8ms/step - loss: 0.7972 - accuracy:
0.0562 - val_loss: 0.5129 - val_accuracy: 0.0750
Epoch 8/20
5/5 [==============================] - 0s 8ms/step - loss: 0.7372 - accuracy:
0.1063 - val_loss: 0.5291 - val_accuracy: 0.1000
Epoch 9/20
5/5 [==============================] - 0s 7ms/step - loss: 0.6691 - accuracy:
0.1187 - val_loss: 0.5636 - val_accuracy: 0.1000
Epoch 10/20
5/5 [==============================] - 0s 7ms/step - loss: 0.6146 - accuracy:
0.1250 - val_loss: 0.5948 - val_accuracy: 0.1000
Epoch 11/20
5/5 [==============================] - 0s 7ms/step - loss: 0.5835 - accuracy:
0.1312 - val_loss: 0.6145 - val_accuracy: 0.0750
Epoch 12/20
5/5 [==============================] - 0s 7ms/step - loss: 0.5672 - accuracy:
0.0938 - val_loss: 0.6197 - val_accuracy: 0.0750
Epoch 13/20
5/5 [==============================] - 0s 7ms/step - loss: 0.5488 - accuracy:
0.0875 - val_loss: 0.6024 - val_accuracy: 0.0500
Epoch 14/20
5/5 [==============================] - 0s 7ms/step - loss: 0.5330 - accuracy:
0.0688 - val_loss: 0.5808 - val_accuracy: 0.0500
Epoch 15/20
5/5 [==============================] - 0s 7ms/step - loss: 0.5165 - accuracy:
0.0562 - val_loss: 0.5553 - val_accuracy: 0.0500
Epoch 16/20
```

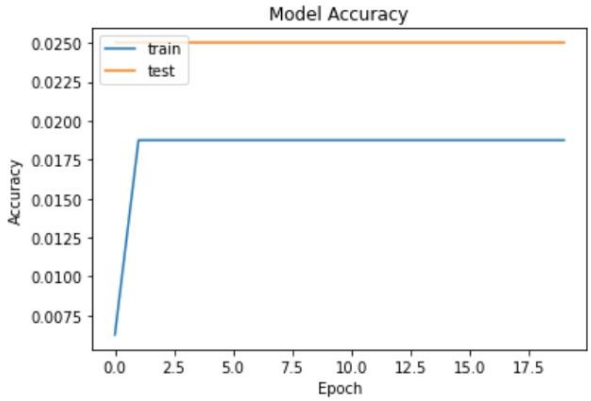
```
5/5 [==============================] - 0s 7ms/step - loss: 0.5018 - accuracy:
0.0688 - val_loss: 0.5355 - val_accuracy: 0.0500
Epoch 17/20
5/5 [==============================] - 0s 7ms/step - loss: 0.4890 - accuracy:
0.0875 - val_loss: 0.5202 - val_accuracy: 0.0750
Epoch 18/20
5/5 [==============================] - 0s 7ms/step - loss: 0.4788 - accuracy:
0.0875 - val_loss: 0.5105 - val_accuracy: 0.0750
Epoch 19/20
5/5 [==============================] - 0s 7ms/step - loss: 0.4706 - accuracy:
0.0938 - val_loss: 0.5068 - val_accuracy: 0.0750
Epoch 20/20
5/5 [==============================] - 0s 7ms/step - loss: 0.4603 - accuracy:
0.0938 - val_loss: 0.4993 - val_accuracy: 0.0750
```

In [218]:

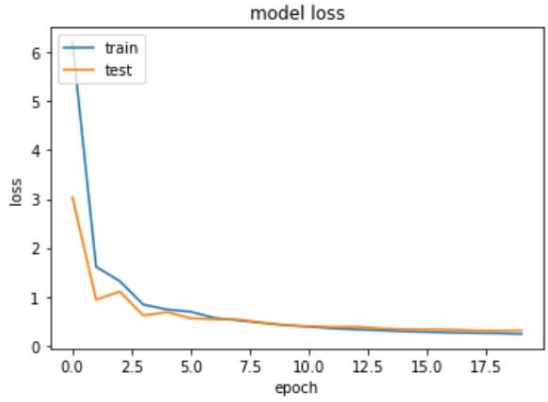
```
xtest=np.array(X_test)
xtest=np.reshape(xtest,(xtest.shape[0],xtest.shape[1],1))
pred=modelc.predict(xtest,verbose=0)
xpred=modelc.predict(sx2,verbose=0)
rmse_test=np.sqrt(np.mean(pred[:200]-y_test[:200])**2)
rmse_train=np.sqrt(np.mean(xpred[:200]-ytrain[:200])**2)
print('Training RMSE: ',rmse_train)
print('Test RMSE: ',rmse_test)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_4124/4139984054.py in <module>
      3 pred=modelc.predict(xtest,verbose=0)
      4 xpred=modelc.predict(sx2,verbose=0)
----> 5 rmse_test=np.sqrt(np.mean(pred[:200]-y_test[:200])**2)
      6 rmse_train=np.sqrt(np.mean(xpred[:200]-ytrain[:200])**2)
      7 print('Training RMSE: ',rmse_train)

ValueError: operands could not be broadcast together with shapes (200,3) (20
0,)
```

In [113]:
```
plt.plot(cnn_history.history['accuracy'])
plt.plot(cnn_history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['train','test'],loc='upper left')
plt.show()
```



In [116]:
```
plt.plot(cnn_history.history['loss'])
plt.plot(cnn_history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



In [ ]:
```
### GRU
```

In [119]:
```
modelg=Sequential()
modelg.add(GRU(100, return_sequences=False, input_shape=(sx.shape[1],1)))
modelg.add(Dropout(0.2))
modelg.add(Dense(units=1))
optimizer=Adam(lr=0.0001)
modelg.compile(optimizer=optimizer, loss='mean_squared_error',metrics=['accuracy'
gru_history=modelg.fit(sx,sy,epochs=20,batch_size=4,shuffle=True,validation_split
```

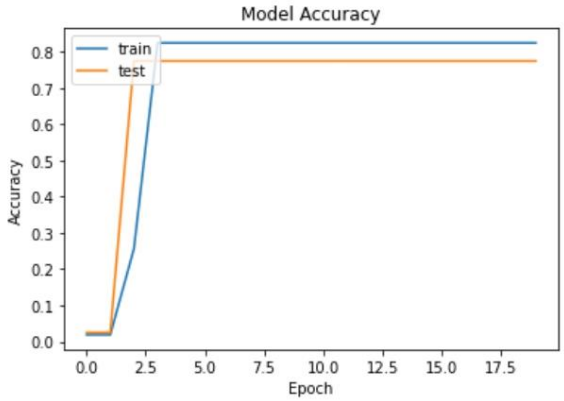
```
Epoch 1/20

C:\ProgramData\Anaconda3\lib\site-packages\keras\optimizer_v2\adam.py:105: User
Warning: The `lr` argument is deprecated, use `learning_rate` instead.
  super(Adam, self).__init__(name, **kwargs)

40/40 [==============================] - 7s 137ms/step - loss: 0.7531 - accurac
y: 0.0188 - val_loss: 0.6001 - val_accuracy: 0.0250
Epoch 2/20
40/40 [==============================] - 5s 130ms/step - loss: 0.5178 - accurac
y: 0.0188 - val_loss: 0.3699 - val_accuracy: 0.0250
Epoch 3/20
40/40 [==============================] - 5s 129ms/step - loss: 0.2736 - accurac
y: 0.2562 - val_loss: 0.1249 - val_accuracy: 0.7750
Epoch 4/20
40/40 [==============================] - 5s 131ms/step - loss: 0.0785 - accurac
y: 0.8250 - val_loss: 0.0664 - val_accuracy: 0.7750
Epoch 5/20
40/40 [==============================] - 5s 134ms/step - loss: 0.0626 - accurac
y: 0.8250 - val_loss: 0.0642 - val_accuracy: 0.7750
Epoch 6/20
40/40 [==============================] - 5s 131ms/step - loss: 0.0682 - accurac
y: 0.8250 - val_loss: 0.0632 - val_accuracy: 0.7750
Epoch 7/20
40/40 [==============================] - 5s 132ms/step - loss: 0.0569 - accurac
y: 0.8250 - val_loss: 0.0630 - val_accuracy: 0.7750
Epoch 8/20
40/40 [==============================] - 5s 132ms/step - loss: 0.0568 - accurac
y: 0.8250 - val_loss: 0.0636 - val_accuracy: 0.7750
Epoch 9/20
40/40 [==============================] - 5s 134ms/step - loss: 0.0620 - accurac
y: 0.8250 - val_loss: 0.0626 - val_accuracy: 0.7750
Epoch 10/20
40/40 [==============================] - 5s 130ms/step - loss: 0.0643 - accurac
y: 0.8250 - val_loss: 0.0638 - val_accuracy: 0.7750
Epoch 11/20
40/40 [==============================] - 5s 136ms/step - loss: 0.0624 - accurac
y: 0.8250 - val_loss: 0.0630 - val_accuracy: 0.7750
Epoch 12/20
40/40 [==============================] - 5s 134ms/step - loss: 0.0585 - accurac
y: 0.8250 - val_loss: 0.0622 - val_accuracy: 0.7750
Epoch 13/20
40/40 [==============================] - 5s 133ms/step - loss: 0.0554 - accurac
y: 0.8250 - val_loss: 0.0648 - val_accuracy: 0.7750
Epoch 14/20
40/40 [==============================] - 5s 135ms/step - loss: 0.0607 - accurac
y: 0.8250 - val_loss: 0.0635 - val_accuracy: 0.7750
Epoch 15/20
40/40 [==============================] - 5s 132ms/step - loss: 0.0612 - accurac
```

```
y: 0.8250 - val_loss: 0.0641 - val_accuracy: 0.7750
Epoch 16/20
40/40 [==============================] - 5s 133ms/step - loss: 0.0586 - accurac
y: 0.8250 - val_loss: 0.0612 - val_accuracy: 0.7750
Epoch 17/20
40/40 [==============================] - 5s 135ms/step - loss: 0.0631 - accurac
y: 0.8250 - val_loss: 0.0611 - val_accuracy: 0.7750
Epoch 18/20
40/40 [==============================] - 5s 132ms/step - loss: 0.0594 - accurac
y: 0.8250 - val_loss: 0.0610 - val_accuracy: 0.7750
Epoch 19/20
40/40 [==============================] - 5s 132ms/step - loss: 0.0608 - accurac
y: 0.8250 - val_loss: 0.0613 - val_accuracy: 0.7750
Epoch 20/20
40/40 [==============================] - 5s 129ms/step - loss: 0.0561 - accurac
y: 0.8250 - val_loss: 0.0619 - val_accuracy: 0.7750
```

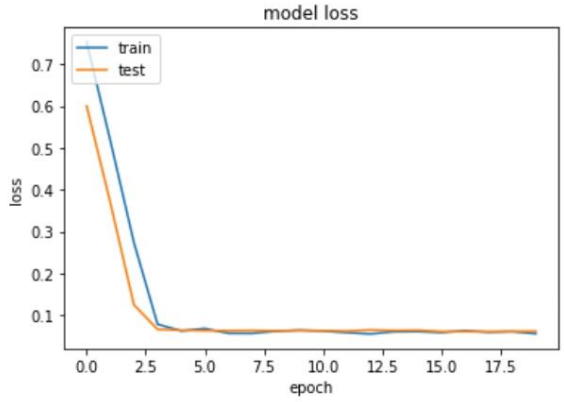
In [220]:
```
print(modelg.summary())
```

```
Model: "sequential_20"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 gru (GRU)                   (None, 100)               30900

 dropout_8 (Dropout)         (None, 100)               0

 dense_25 (Dense)            (None, 1)                 101

=================================================================
Total params: 31,001
Trainable params: 31,001
Non-trainable params: 0
_________________________________________________________________
None
```

In [120]:
```
plt.plot(gru_history.history['accuracy'])
plt.plot(gru_history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['train','test'],loc='upper left')
plt.show()
```



In [121]:
```
plt.plot(gru_history.history['loss'])
plt.plot(gru_history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



In [161]:
```
### From tutorial
### 1
```

In [199]:
```
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Dense
from tensorflow.keras import layers
from tensorflow.keras.layers.experimental import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from keras.utils.np_utils import to_categorical
from sklearn.datasets import load_wine
from tensorflow.keras.optimizers import SGD, RMSprop, Adam
from keras.regularizers import l2
from keras.layers import Dense, Activation
import time
start_time = time.time()
```

In [175]:
```
X = data[['day_of_week','casualty_home_area_type',
        'sex_of_casualty','age_of_casualty', 'junction_control',
        'light_conditions','weather_conditions', 'road_surface_conditions', 'carr
y = data['accident_severity']
```

In [176]:
```
X.head()
```

Out[176]:

	day_of_week	casualty_home_area_type	sex_of_casualty	age_of_casualty	junction_control	light_
0	3	1	1	3.135494	-1	
1	3	1	2	3.583519	4	
2	3	1	1	3.178054	4	
3	3	1	2	4.077537	2	
4	3	1	1	3.332205	-1	

In [182]:
```
y.unique()
```

Out[182]:
```
array([3, 2, 1], dtype=int64)
```

In [183]:
```
def ff(row):
    if row==1:
        return 0
    elif row==2:
        return 1
    elif row==3:
        return 2
y=y.apply(ff)
```

In [184]:
```
y.unique()
```

Out[184]:
```
array([2, 1, 0], dtype=int64)
```

In [185]:
```
xtr,xte,ytr,yte=train_test_split(X,y,test_size=0.25, random_state=42)
```

In [186]:
```
sc=StandardScaler(with_mean=True, with_std=True)
sc.fit(xtr)
```

Out[186]:
```
StandardScaler()
```

In [187]:
```
xtrs=sc.transform(xtr)
xtes=sc.transform(xte)
```

In [188]:
```
no_classes=3
ytr=keras.utils.to_categorical(ytr,no_classes)
yte=keras.utils.to_categorical(yte,no_classes)
```

In [189]:
```
model=Sequential()
model.add(Dense(9,input_dim=9, activation='relu'))
model.add(Dense(64,activation='relu'))
model.add(Dense(3,activation='sigmoid'))
print(model.summary())
```

```
Model: "sequential_21"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_26 (Dense)            (None, 9)                 90

 dense_27 (Dense)            (None, 64)                640

 dense_28 (Dense)            (None, 3)                 195

=================================================================
Total params: 925
Trainable params: 925
Non-trainable params: 0
_________________________________________________________________
None
```

In [190]:
```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accura
```

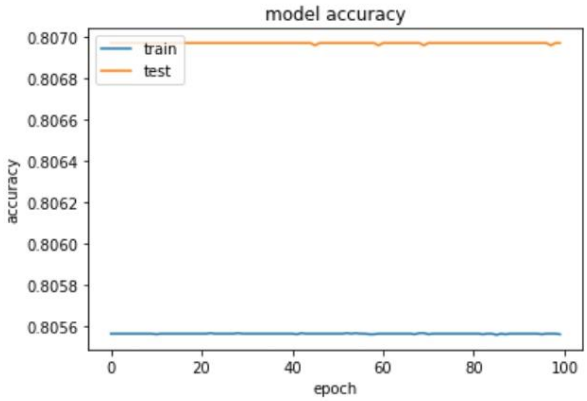
In [196]:
```
history=model.fit(xtrs,ytr, batch_size=100, epochs=100, verbose=1, validation_spl
```

```
Epoch 1/100
3521/3521 [==============================] - 5s 1ms/step - loss: 0.5347 - acc
uracy: 0.8056 - val_loss: 0.5328 - val_accuracy: 0.8070
Epoch 2/100
3521/3521 [==============================] - 5s 1ms/step - loss: 0.5347 - acc
uracy: 0.8056 - val_loss: 0.5330 - val_accuracy: 0.8070
Epoch 3/100
3521/3521 [==============================] - 5s 1ms/step - loss: 0.5347 - acc
uracy: 0.8056 - val_loss: 0.5332 - val_accuracy: 0.8070
Epoch 4/100
3521/3521 [==============================] - 5s 1ms/step - loss: 0.5346 - acc
uracy: 0.8056 - val_loss: 0.5328 - val_accuracy: 0.8070
Epoch 5/100
3521/3521 [==============================] - 5s 1ms/step - loss: 0.5346 - acc
uracy: 0.8056 - val_loss: 0.5327 - val_accuracy: 0.8070
Epoch 6/100
3521/3521 [==============================] - 5s 1ms/step - loss: 0.5346 - acc
uracy: 0.8056 - val_loss: 0.5326 - val_accuracy: 0.8070
Epoch 7/100
```

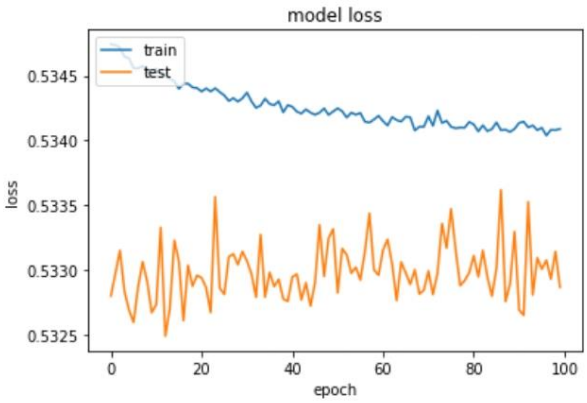
In [207]:
```
score=model.evaluate(xtes, yte, verbose=1)
print("Names of the score metrics in model evaluation are: ")
print(model.metrics_names)
print("The score values are: Loss (Categorical Cross Entropy) and Accuracy (%)")
print(score)
print("\n")
```

```
4584/4584 [==============================] - 4s 963us/step - loss: 0.5389 - acc
uracy: 0.8051
Names of the score metrics in model evaluation are:
['loss', 'accuracy']
The score values are: Loss (Categorical Cross Entropy) and Accuracy (%)
[0.5388577580451965, 0.8050597906112671]
```

In [208]:
```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



In [209]:
```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



In [197]:
```
### 2
```

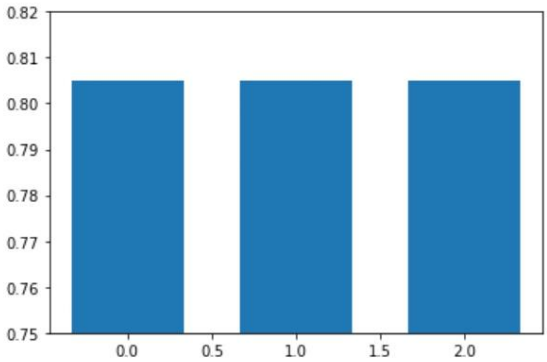
In [202]:
```
no_classes=3
test_split=0.25
batchsize=128
```

In [203]:
```
accuracy=[]
for OPTIMIZER in [SGD(),RMSprop(),Adam()]:
    for epoch_no in [100]:
        for neurons_no in [64]:
            model=Sequential()
            #input
            model.add(Dense(units=9,input_shape=(xtr.shape[1],),kernel_regularize
            model.add(Activation('relu'))
            #hidden
            model.add(Dense(units=neurons_no, kernel_regularizer=l2()))
            model.add(Activation('relu'))
            #output
            model.add(Dense(units=no_classes))
            model.add(Activation('softmax'))
            model.summary()
            #compile and fit
            model.compile(loss='categorical_crossentropy',optimizer=OPTIMIZER,met
            model.fit(xtrs,ytr,batch_size=batchsize,epochs=epoch_no,verbose=1, va
            #evaluate
            score=model.evaluate(xtes,yte,verbose=1)

            print()
            print('Optimizers: ', OPTIMIZER)
            print('Epoch Sizes: ', epoch_no)
            print('Neurons or Units: ', neurons_no)
            print("Test score:", score[0])
            print('Test accuracy:', score[1])
            accuracy.append(score[1])
            print()
```

```
Model: "sequential_24"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_35 (Dense)            (None, 9)                 90

 activation_6 (Activation)   (None, 9)                 0

 dense_36 (Dense)            (None, 64)                640

 activation_7 (Activation)   (None, 64)                0

 dense_37 (Dense)            (None, 3)                 195

 activation_8 (Activation)   (None, 3)                 0

=================================================================
Total params: 925
Trainable params: 925
```

In [206]:
```
print(accuracy)
y = accuracy; N = len(y); x = range(N); width = 1./1.5;
plt.ylim(0.75,.82)
plt.bar(x,y,width); plt.show()

print()
print("Execution Time %s seconds: " % (time.time() - start_time))
```

```
[0.8050597906112671, 0.8050597906112671, 0.8050597906112671]
```




```
Execution Time 1441.4669842720032 seconds:
```

In [ ]: