IBM Data Science Specialisation

Capstone Project

Title: Predicting automobile crash severity in New Zealand

Fernando Castellanos - October 2020

1 Business Problem/ Problem Description and Background

According to the World Health Organization, road traffic injuries caused an estimated 1.35 million deaths worldwide in the year 2016 (*Global status report on road safety 2018, WHO*). The report indicates that more than half of all the deaths are among vulnerable users: cyclists, pedestrians or motorcyclists. Additionally, road traffic injuries are currently the leading cause of death for children and young adults. These statistics provide a regrettable picture: even with all the safety devices installed in modern cars (such as seat belts, airbags, anti-lock brakes, shatter-resistant glass and head restraints), fatalities due to automobile crashes still occurring in high numbers. Hence, it is important to have tools to help reduce the number of accidents and their related fatalities.

1.1 Interest/ Target Audience

Thanks to the advancement of data collection and data analysis, it is possible now to better understand how automobile crashes happen and how to attempt to predict their outcome (severity). A practical use of such a predictive tool would be to enable the transport, security and emergency local agencies to analyse incoming reports of accidents and being able to determine on-line the severity of such a crash. This can be used to dispatch the adequate emergency response and activate any traffic controls.

Estimating the severity of an automobile crash is difficult and its accuracy will depend on the data available and the type of model used to define the problem. Nowadays, transport agencies around the world accumulate a huge variety of traffic information, and in most instances, they collect data of all reported automobile crashes. These datasets have usually well-defined features describing aspects such as weather, location, light and road conditions at the time of the crash and its severity. Making use of one of these datasets will give us better chances to construct a usable model.

1.2 Deliverables

The goal is to predict the severity of a given automobile crash sample. To achieve this I will implement a complete machine learning process from getting adequate data, performing exploratory data analysis and formulating the problem into a machine learning model and interpreting the results. All files for the project are available under my Github repository here, with the exemption of the data file and some additional python packages. Because of the size of the dataset file(119 MB) it cannot be uploaded in Github, hence I installed a copy of the original data file in my Dropbox account. The Jupyter notebook loads the dataset from this location in Dropbox. The Github repository contains:

- This project report
- The Jupyter notebook with the complete python code (Capstone_project_crashes_final.ipynb)
- The presentation (Capstone_project_presentation)

- A link to the Jupyter nbviewer showing the complete notebook with all its inputs and outputs. I have done this last one as a precautionary action because the code uses several uncommon libraries, which might create problems for the person reviewing my program and the full execution of the notebook takes a couple of hours.

2. The Dataset

2.1 Data source

After searching the internet for automobile crashes datasets I discovered that New Zealand Transport Agency has an open data repository, that includes *"Crash Analysis System (CAS) data"* (on a personal note I live in New Zealand, so this project could give me some interesting insights that I can relate to). This dataset provides reported automobile crash information and is the type of data required for the project. The dataset is available in several formats, I used the CSV version, found here as disaggregated crash data. This CSV file has a size of 199MB and because of its size, it is distributed as a zip file with a size of 31 MB. Files of this

The crash data provides information about each traffic crash reported by the New Zealand Police since 1 January 2000. The dataset is updated quarterly and the current set was last updated in July 2020 and includes crash data up to the 29th of February. The data covers automobile crashes on all New Zealand roadways or places where the public have legal access with a motor vehicle. It is important to note that not all crashes are reported to the Police and the level of reporting increases with the severity of the crash. Due to the nature of non-fatal crashes, it is believed that these are under-reported.

2.2 Dataset characteristics

The particular dataset I used for my modelling has a total of 725548 samples, with 72 features on each of them. The New Zealand Transport Agency provides a metafile or a webpage with a brief description of the features. A table presenting my compilation of these descriptions is included in Appendix A, one important aspect of these definitions is that for categorical features it includes the valid classes. This information was important during the data exploration to determine which values were proper classes. The feature with the information regarding the crash severity is named "*crashSeverity*", it is a categorical feature that can take the values: 'Fatal Crash', 'Serious Crash', 'Minor Crash', 'Non-Injury Crash'. This is determined by the worst injury sustained in the crash at the time of entry.

Using the methods .shape, .info and .head I obtained a first glimpse at the dataset. The information shows that the features in the dataset are both categorical and numerical, and a few spatial geodata ones used to indicate the location of the crash site. The categorical features are automatically labelled as 'object' and most of the numerical are labelled as 'float64'. It is possible to see that several features are missing values and two of them do not have any values.

Several categorical features provide information about different attributes and conditions of the crash event. All these features have a small number of possible values. For example, the feature "*roadLane*" informs about the road configuration at the crash site. Its possible values are: '1-way', '2-way', 'Median' and 'Off road' which are self-explanatory. Another example is the feature "*light*" which informs about the natural light conditions at the time and place of the crash. Its possible values are: 'Bright Sun', 'Overcast', 'Twilight, 'Dark' or 'Unknown', which again are self-explanatory.

The dataset has a large number of numerical features, which are in most cases counters providing details about a variety of objects that could be involved in the crash. For example, the feature *"parkedVehicle"* indicates how many times a parked or unattended vehicle was struck in the crash. The feature "debris" indicates how many times debris, boulders or items dropped or thrown from a vehicle(s) were struck in the crash. Additionally, there are other counters such as *"seriousInjuryCount"* which informs about the number of people seriously injured in the crash. A similar counter

"fatalCount" provides the number of fatalities for a crash of severity 'Fatal crash'. So, for example, a crash of severity 'Serious crash' would have a *"fatalCount"* of zero and most probably a value greater than zero for *"seriousInjuryCount"*.

Another group of numerical features are values describing various speeds, such as "speedLimit" which provides the speed limit in force at the crash site at the time of the crash or "temporarySpeedLimit" which provides the temporary speed limit at the crash site if one exists (e.g. for road works). Besides these, some other numerical features provide information about the year of the crash, the number of lanes on the road of the crash and identifiers.

2.3 The target feature '*crashSeverity*'

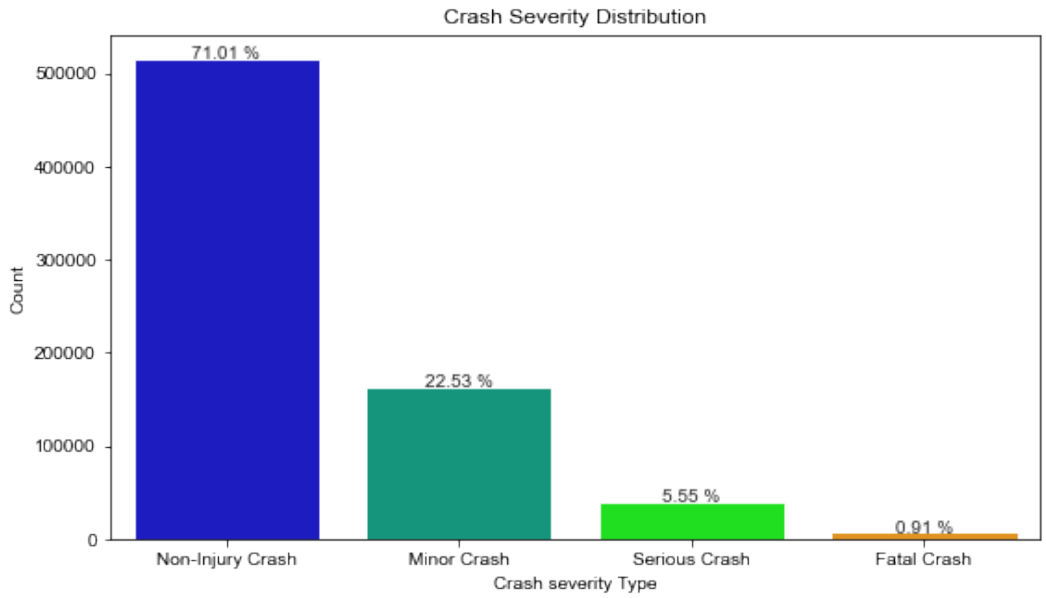

Figure 2.1: Values distribution for 'crashSeverity'

The target feature, representing the crash severity is `*crashSeverity*'. It is a categorical feature, with a full set of values (725548 samples). The feature has four classes: 'Non-Injury Crash' with 515182 samples, 'Minor Crash' with 163479 samples, 'Serious Crash' with 40300 samples and 'Fatal Crash' with 6587 samples. A graph with the values distribution is shown below.

The graph and values indicate a large imbalance among the four classes. The class 'Non-injury Crash' is very large compare with the other three: more than 3 times the 'Minor Crash' class, more than 12 times the 'Serious Crash' class and more than 78 times the 'Fatal Crash' class. Hence, the two last classes are seriously imbalanced and some models could don't work too well at identifying the minority classes. Additionally, some of the models performance metrics could conceal these issues.

For this study of automobile crashes, the imbalance is due to the nature of the problem. The possibility of a crash being a fatal one is very small, compared with the possibility of being a minor one, without injuries. Hence, the imbalance cannot be solved by getting additional data. Further in this study I will use three general strategies to help to overcome this imbalance: 1) resample the dataset, 2) use different algorithms and 3) use several performance metrics.

3. Data Cleaning – Initial Removal of Non-relevant features

As discussed in the previous section the dataset is quite large. Using the description of the features, as per Appendix A it is possible to examine and determine that some of the features are not of interest for the goal of predicting the crash severity. Hence, it is possible to remove some features initially. The initial features that were dropped are:

- ‘X’, ‘ Y’ = geolocation of the crash site – It provides the geographical coordinates for the precise location of the crash site. It is not required for this project.
- ‘OBJECTID’ = unique crash identification – It is of no interest in this project.
- ‘areaUnitID’ = unique identifier of an area unit – This feature has a vague definition and it is of no interest in this project.
- ‘crashDirectionDescription’ = geo direction reference – It has an unclear definition and it is of no interest in this project.
- ‘crashFinancialYear’, ‘crashYear’ = financial and calendar year at the time the crash occurred – The project does not look at the evolution of the crashes in time, hence it is of no use for the project.
- ‘crashLocation1’ and ‘crashLocation2’ = specific description of the crash location, using traffic and landmark markers. These two features are limited to the crash location and because of this, no two samples are the same. This large amount of possibilities(classes) makes them of no use for this project.
- ‘fatalCount’, ‘seriousInjuryCount’ and ‘minorInjuryCount’ = the number of fatalities or people injured during the crash – It is of no use for the project and these features are highly correlated to our target (‘crash severity’).
- ‘meshblockID’ = unique block identifier – It has an unclear definition and it is not required for this project.
- ‘region’ = geographical region where the crash occurred. This feature provides a reference to very large areas of the country. Instead of using this feature, the feature ‘tlaName’ was used as it provides a more discrete sectionalisation. Additionally, the number of data samples for ’region’, ‘tlaName’ and ‘tlaId’ is the same, so there is no benefit in keeping the three of them. Hence, ’region’ was dropped.
- ‘tlaId’ = unique identifier for a territorial local authority, instead the project used the feature ‘tlaName’ with the names of the local authority.

As mentioned before, using the .info() method was possible to observe that two of the features have not data. These features are:

- ‘crashRoadSideRoad’ = indicates if the main car involved in the crash was on the crash road or a side road.
- ‘intersection’ = indicates if the crash occurred at an intersection.

These two features were dropped because they do not provide any information.

Other two features have very few values, less than 29000 non-NaN, which represents less than 4% of the possible total data for a feature. These amount of data is not enough to support analysis such as correlation neither help with the modelling. The two features are:

- ‘advisorySpeed’ = it is the advised speed at the crash site. It has only 26532 values out of 725548, representing just about 4%.
- ‘temporarySpeedLimit’ = it is the temporary speed at the crash site. It has only 9676 values out of 725548, representing just about 2%.

The Jupyter notebook includes graphs and listings of the values distribution for both of these features. Both features are missing a large amount of data and a probable reason for this could be the omission of data at the crash site. Also in most instances, these features would not have a value,

as they refer to advisory and temporary speed arrangements that only are applicable for special circumstances such as works in progress. Hence, these two features were dropped from our dataset.

After dropping all the features mentioned in this section, the number of features was reduced from 72 to 53.

4. Data Exploration

4.1 Data exploration and wrangling – missing data, sanitation

The features list information obtained previously (method .info()), indicates that several features are missing data (NaN). A closer inspection shows that the majority of them are counters providing details about a variety of objects that could be involved in the crash (e.g 'bicycle' indicates how many bicycles were involved in the crash, 'postOrPole' indicates how many poles or posts were hit during the crash). Most probably this missing data corresponds to intentionally omitted values during the entering of data at the crash site. These omissions would indicate that the respective feature does not apply to the specific crash event and then, it can be replaced by a value of zero(0).

A list of the counters for which their missing values were replaced with zero are: 'bicycle', 'bridge', 'bus', 'carStationWagon', 'cliffBank', 'debris', 'ditch', 'fence', 'guardRail', 'houseOrBuilding', 'kerb', 'moped', 'motorcycle', 'objectThrownOrDropped', 'otherObject', 'otherVehicleType', 'overBank', 'parkedVehicle', 'pedestrian', 'phoneBoxEtc', 'postOrPole', 'roadworks', 'schoolBus', 'slipOrFlood', 'strayAnimal', 'suv', 'taxi', 'trafficIsland', 'trafficSign', 'train', 'tree', 'truck', 'unknownVehicleType', 'vanOrUtility', 'vehicle' and 'waterRiver'.

Other features that contain NaN values were assessed and two basic options were used to resolve the missing values: 1) one option is to remove the samples containing NaN values, this option has to be properly evaluated and make sure that not too many samples are lost, especially of the minority classes and 2) another option is to replace these NaN values with suitable ones, following some logic criteria to calculate the replacements(e.g average feature value, extrapolation, etc.). As mentioned the first option has to be considered carefully, as per by removing the samples the dataset distribution with respect to the crash severity might be influenced (e.g. it will not be advisable to delete a large portion of samples with class 'Fatal Crash', as their numbers are very low to start with).

Looking initially at the features with NaN values and their context in the dataset the following possible solutions were proposed:

- 'speedLimit' = traffic speed limit at the crash site – Using the fact that most urban areas in New Zealand have a speed limit of 50 km/h and 100 km/h for outside urban areas, it could be possible to replace the NaN values. The feature 'urban' provides this location information (crash site in an urban area or not) for each crash event and it can be mapped to replace the NaN values accordingly to the crash location. Because this replacement requires the use of another column(feature) as a lookup reference, it is not possible to use a simple replace function for the substitution.

- 'tlaName' = name of the local authority where the crash occurred – Any NaN value could be replaced with 'Unknown' as there is not a way to determine an appropriated replacement value with the information available.

- NumberOfLanes_ = the number of lanes at road crash site – Any NaN value could be replaced with '2' (two lanes) as this is the most common number of lanes in New Zealand roads.

- 'holiday' = indicates whether the crash happened during a holiday, the list of possible values(holidays) are: Christmas New Year, Easter, Queens Birthday and Labour Weekend. It is

assumed the reason for a NaN value is that there was not a holiday at the time of the crash and the data entry was intentionally left blank at the time of the crash. Hence, the NaN values could be replaced with 'None'.

- 'directionRoleDescription' = direction of the principal vehicle involved in the crash – Any NaN and Null value could be replaced with 'Unknown' as there is not a way to determine a replacement value.

After this initial evaluation, a more detailed analysis of these features with NaN values was carried out to determine the best possible solution. For further details on the calculations please refer to the Jupyter notebook which includes the evaluation of the feature values distribution and the feature NaN values distribution in terms of the crashSeverity class. These distributions were calculated using customised functions (create_count_percentages and map_vacios) that are based on .value_counts, .is_null, .fillna and .groupby. Additionally, there is a graph showing the feature values distribution, with most of the graphs produced with a customised function that reduces the number of commands.

- 'speedLimit' = traffic speed limit at the crash site. It has 492 samples with NaN values, corresponding to 0.06% of the total set. The feature values distribution is shown in the graph below.

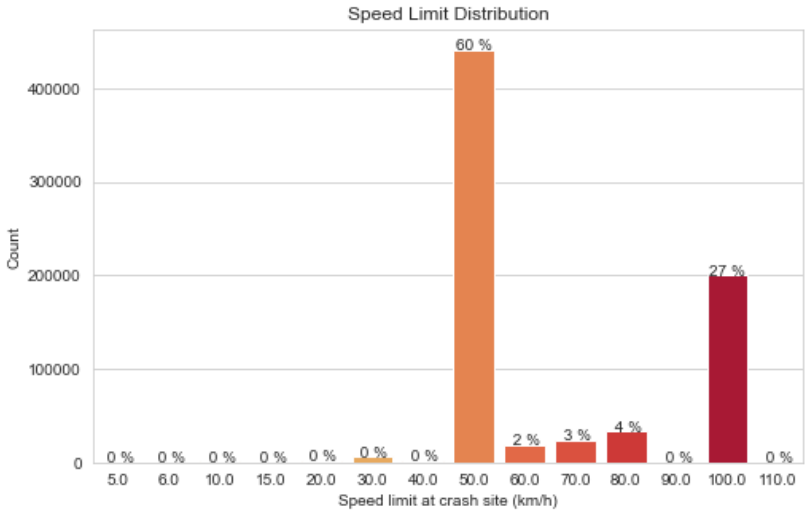

Figure 4.1: speedLimit distribution of values (% based on full set of values)

The data and graph show that the most common speeds are 50 and 100 km/h, supporting the initial idea of how to substitute the NaN values. Other 12 possible classes have very little impact on the general feature distribution. From the total of 492 samples with NaN values, only one corresponds to a 'Fatal Crash' and the large majority are 'Non-Injury Crash'. Hence, the replacement of these NaN values will not affect the class balance of the dataset and the 492 NaN values were replaced with 50 or 100 km/h values depending on the value of the corresponding feature 'urban'.

- 'tlaName' = name of the local authority where the crash occurred. It has 35 samples with NaN values, corresponding to 0.005% of the total set. The feature has a total of 67 possible classes (local government areas). Because of the large number of feasible values, it is not possible to show the complete feature distribution instead, the 14 classes with the largest values are shown in the graph below.

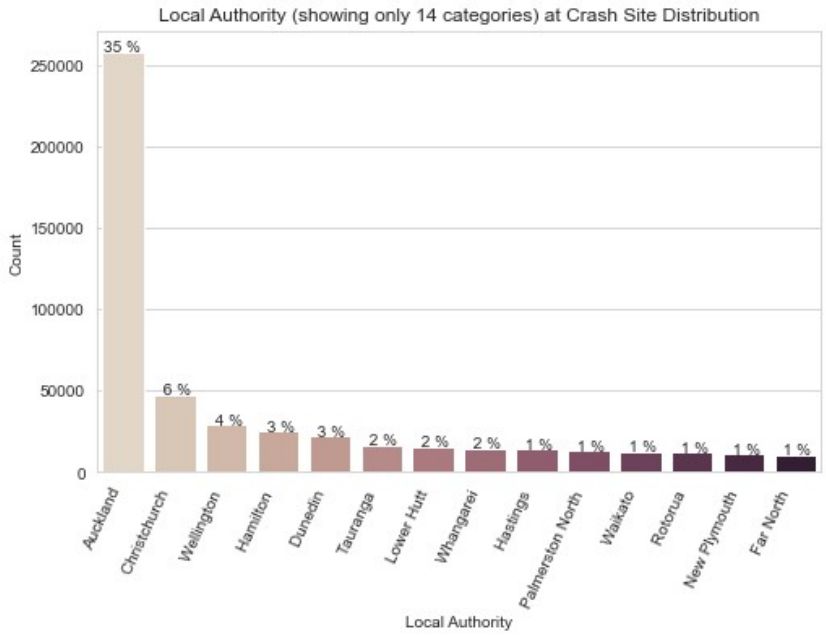

Figure 4.2: tlaName distribution of values (% based on full set of values) – Only 14 classes shown

The data and graph behave as expected, with most of the automobile crashes occurring in the main cities: Auckland – with a large percentage -, Christchurch, Wellington, Hamilton and Dunedin. The data indicates that 35 values are missing(NaN), of those the large majority are 'Non-Injury Crash' and there is one 'Minor Crash', one 'Serious Crash' and none 'Fatal Crash'. Therefore, for this case was better to disregard the initial proposed solution and instead delete the 35 samples, given that it is a small number and their deletion will not affect the dataset as a whole.

- 'NumberOf Lanes' = number of lanes at the road where the crash occurred. It has 1193 samples with NaN values, corresponding to 0.16% of the full set. The distribution of values shows that there are 8859 values – corresponding to 1.22% - assigned to 0.0. A graph of the values distribution is shown below.

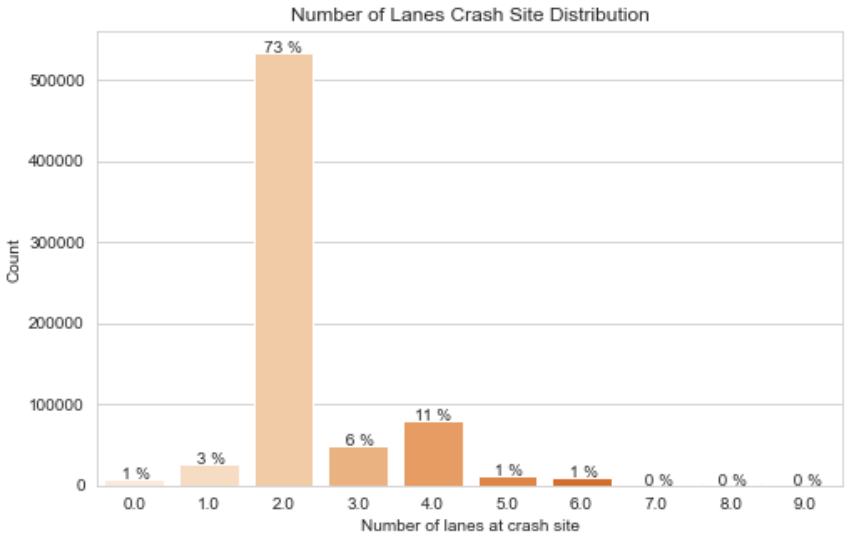

Figure 4.3: NumberOfLanes distribution of values (% based on full set of values)

The data and graph indicate that of the 1193 NaN values, 13 belong to the class 'Fatal Crash' and the rest is divided among the other three classes. This feature shows a discrepancy, there are 8859 samples with a value of 0 number of lanes, which cannot represent a realistic crash scenario. Of those 8859 samples, 61 are 'Fatal Crash' and a large majority are 'Non-

Injury Crash'. The features listing and description (Appendix A) does not include this specific value or an explanation for it or a relationship with another feature. Because the number of samples with value 0 is large it was decided to leave these samples as its removal might have a repercussion on the composition of the dataset. The missing values (NaN) were replaced by '2 lanes', which is the most common value, as shown in the graph.

- 'holiday' = describes whether the crash happened during a holiday (possible values are: Christmas New Year, Easter, Queens Birthday and Labour Weekend). It has 685925 samples with NaN values, which corresponds to 94.5% of the full set. The distribution of the feature values is shown in the graph below.

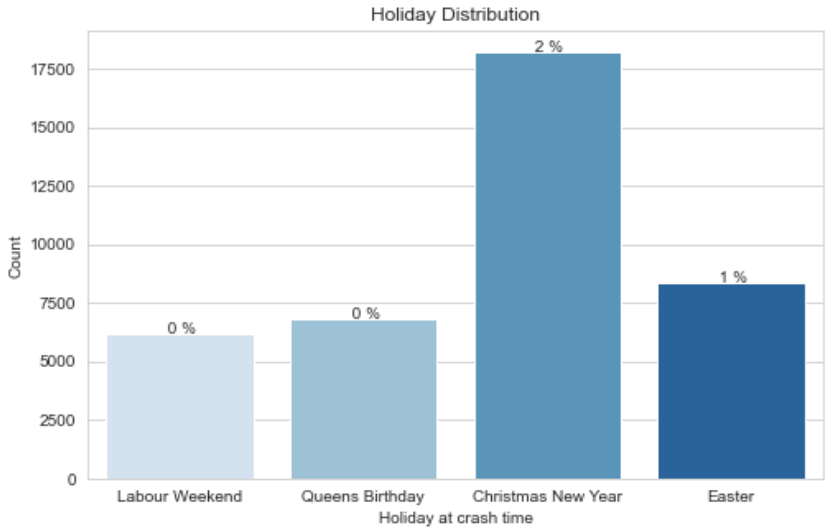

Figure 4.4: holiday distribution of values (% based on full set of values)

The data and graph indicate that most of the data correspond to missing values(NaN), 94.5% of the total. Because of this large value, there was no need for the composition analysis in terms of crash severity, as there would be a large number of samples with 'Fatal Crash'. Examining the feature description in Appendix A it states that one of the possible classes is 'None' however, the dataset does not contain any sample with this value. It is clear that the `None' class is there to represent days no-holiday days and most probably the 'NaN' values in the dataset represent the same type of no-holiday days. It is quite possible that at the crash site, during the data entry, the value was intentionally left blank for days outside the holidays. Hence, it was decided to replace the 'NaN' values with 'None'.

- 'directionRoleDescription' = geographical direction of the main vehicle at the crash site. It has 118 samples with 'NaN' values and 3070 samples with 'Null' values corresponding to 0.016% and 0.42% of the full set, respectively. The distribution of the feature values is shown in the graph below.

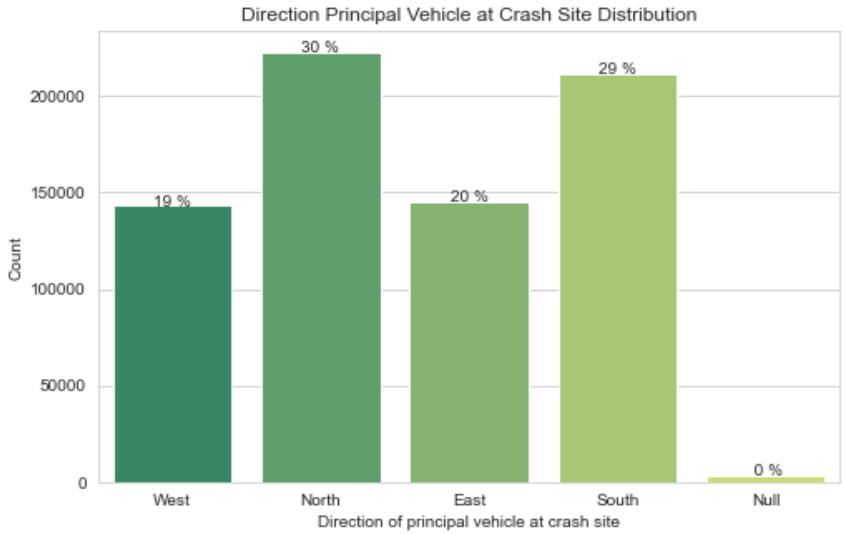

Figure 4.5: directionRoleDescription distribution of values (% based on full set of values)

The data and graph show that the data is quite balanced among the four cardinal points. However, there is data under two other different classes, 'NaN' and 'Null' The feature description (Appendix A) does not include 'Null' as a possible class, but there are 3070 samples that have this value and only 2 of those are 'Fatal Crash'. The large majority of these 'Null' samples are 'Non-Injury Crash'. Additionally, there are 118 samples with missing values (NaN) and only 4 of them are 'Fatal Crash'. The rest of the missing data splits among the other crash severity classes, with 'Minor Crash' being the largest.

This feature can only take one of the four cardinal points classes. The 'Null' values seem like a mistake, given that there is no such a class in the definition and the missing (NaN) values seem like forgotten data. A summary of the data shows that these two sets of values, 'Null' and 'NaN' represent a total of 3188 samples, but only 6 of them are 'Fatal Crash'. As there is not a way to replace any of these values with an appropriate value (e.g. derived from another feature) and there are only a few 'Fatal Crash' samples in the group, it was decided to drop the samples corresponding to the 'Null' and 'NaN' values.

4.2 Data Exploration – Review categorical features, sanitation

This section continues the data exploration, check of data sanity and sanitation of the categorical features.

- 'crashSHDescription' = indicates if the crash occurred in a State Highway – This feature has only three classes: 'No', 'Yes' and 'Unknown'. The last class is listed in the feature definition in Appendix A, so this is a valid one for this feature and the feature has only 42 samples with 'Unknown'. The data shows that only around 30% of the crashes occur on a Highway. Figure 4.6 shows the value distribution for 'crashSHDescription'.
- 'flatHill' = indicates if the road where the crash occurred is flat or hilly. This feature has only three classes: 'Flat', 'Hill Road' and 'Null'. The data shows that most of the crashes (80%) occur on flat terrain (Figure 4.6). The feature description, in Appendix A, does not include a class 'Null' and there are 3115 samples with this value. Looking at the composition of these 'Null' samples in terms of their crash severity, it shows that only 7 samples are 'Fatal Crash' and most of the values are `Non-Injury Crash'. This distribution shows that removing the 'Null' samples from the dataset would not alter its balance. Hence, the 3115 samples were removed from the dataset.

- ‘light’ = describes the natural light encountered at the time and place of the crash site. The values distribution indicates that most of the data is more or less equally distributed among three classes: ‘Bright sun’, ‘Overcast’ and ‘Dark’. Additionally, there are 1592 samples with the value of ‘Unknown’. The feature definition, in Appendix A, states that ‘Unknown’ is a valid class for this feature, so these samples are proper and they can remain unchanged. Figure 4.6 shows the value distribution.

- ‘roadCharacter’ = description of the general nature of the road at the crash site. This feature attempts to categorise some special road locations using classes such as ‘Bridge’, ‘Motorway ramp’, and ‘Speed hump’. The values distribution indicates that most of the data is of class ‘Nil’ with 693768 samples, representing 96% of the full set. The feature description, in Appendix A, states that this is a valid class. The ‘Nil’ value can be interpreted as when the crash site did not correspond to any of the other classes, basically a normal section of road. The data indicates that crashes at bridges, motorway ramps and rail xings are the most common. Figure 4.6 shows the values distribution for the feature.

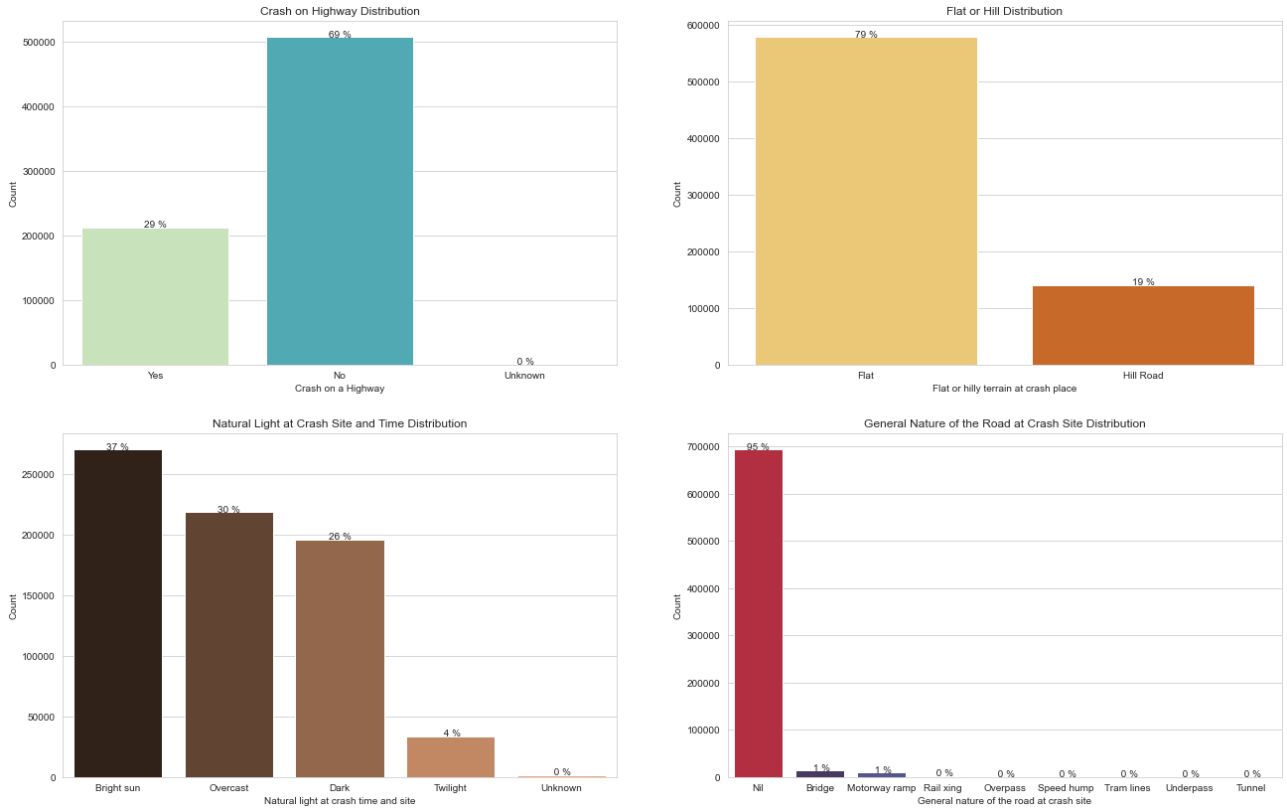

Figure 4.6: Values distributions for ‘crashSHDescription’, ‘flatHill’, ‘light’ and ‘roadCharacter’ (% based on full set of values)

- ‘roadLane’ = general nature of the road (ways of traffic) at the crash site. The data shows that most of the crashes (close to 90% - 645114 samples) occur on ‘2-way’ roads, followed by ‘1-way’ roads with 63419 samples (8.82%), ‘Off road’ with 10435 samples (1.45%) and ‘Null’ with 242 samples (0.03%). The feature description, in Appendix A, indicates that ‘Null’ is not a valid class for the feature. Looking at the composition of the 242 ‘Null’ samples in terms of their crash severity, it shows that there are no ‘Fatal Crash' samples and most of them (179) are ‘Non-Injury Crash’. This suggests that removing the ‘Null’ samples from the dataset would not alter its balance. Hence, the samples were removed from the dataset. Figure 4.7 shows the feature values distribution.

- ‘roadSurface’ = describes the type of road surface at the crash site. The feature values distribution indicates that there are four classes: ‘Sealed’, ‘Unsealed’, ‘Null’, and ‘End of seal’. The data shows that most of the crashes (close to 98% - 704650 samples) occur on sealed

roads. The ‘Null’ and ‘End of seal’ classes do not appear as valid ones in the feature description, in Appendix A. Looking at the composition of the 52 samples with ‘Null’ and 30 samples with ‘End of seal’ in terms of their crash severity, it shows that there are no ‘Fatal Crash’ samples for any of the two groups. Most of the samples are with ‘Minor Crash’ and ‘Non-Injury Crash’. These data distributions show that removing the ‘Null’ and ‘End of seal’ samples from the dataset would basically, not alter its balance. Hence, the samples were removed from the dataset. Figure 4.7 shows the feature values distribution.

- ‘streetLight’ = describes the status of street lightning at time and site of the crash. The feature values distribution shows that there are four classes: ‘Null’, ‘Off’, ‘On’ and ‘None’. Most of the data is of class ‘Null’ with 289461 samples, representing 40% of the full set. However the feature description, in Appendix A, does not include the ‘Null’ class, but there is an `Unknown' class. It has been assumed that for this case the ‘Null’ class can be reclassified as ‘Unknown’, as both of them refer to the same absence of information and most probably is an entry data mistake. Hence, the ‘Null’ values were renamed as ‘Unknown’.

  Additionally, there is a ‘None’ set of values and from the feature definition, in Appendix A, this is a valid class for this feature. Figure 4.7 shows the feature values distribution.

- ‘trafficControl’ = describes any traffic control signals at the crash site. The data shows that most of the data is of classes ‘Nil’, with 294516 samples representing 41%, and ‘Unknown’, with 181174 samples representing 25%. These two classes are included in the feature definition in Appendix A, making them valid. The samples with ‘Nil’ values are important for the modelling as they indicate the lack of traffic signals, which could help to manage traffic at the crash site. The feature values distribution shows that most crashes, where a traffic signal exists, occur at ‘Give way’, ‘Traffic signals’ and ‘Stop’ signals. Figure 4.7 displays the distribution for the feature.

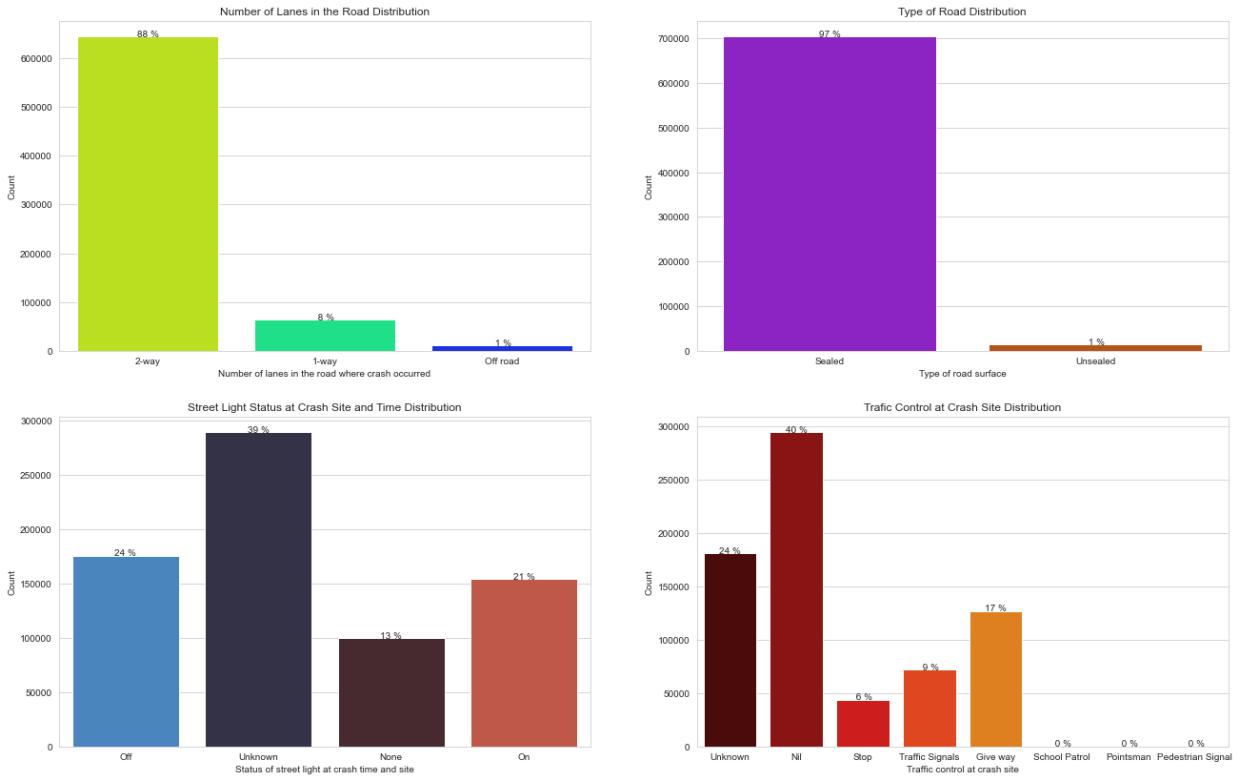

Figure 4.7: Values distribution for ‘roadLane’, ‘roadSurface’, ‘streetLight’ and ‘trafficControl’ (% based on full set of values)

- ‘urban’ = provides a generic location of the crash site. The data shows that there are only two classes ‘Urban’ and ‘Open’ and that most of the crashes (486482 samples representing 67.7%) occur in urban areas, with more than twice the number than in the open road. Figure 4.8 shows the feature values distribution.
- ‘weatherA’ = first data field to describe weather conditions at crash time and site. The data indicates that most of the crashes (559993 samples representing 77.9%) occur during fine weather, followed by rainy weather. The feature data distribution includes a class ‘Null' that is not included in the feature description, Appendix A, instead there is an ‘Unknown’ class. Most probably the two classes refer to the same type of conditions/data. Hence, the ‘Null' values were renamed `Unknown'. A graph of the data is shown below (Figure 4.8).

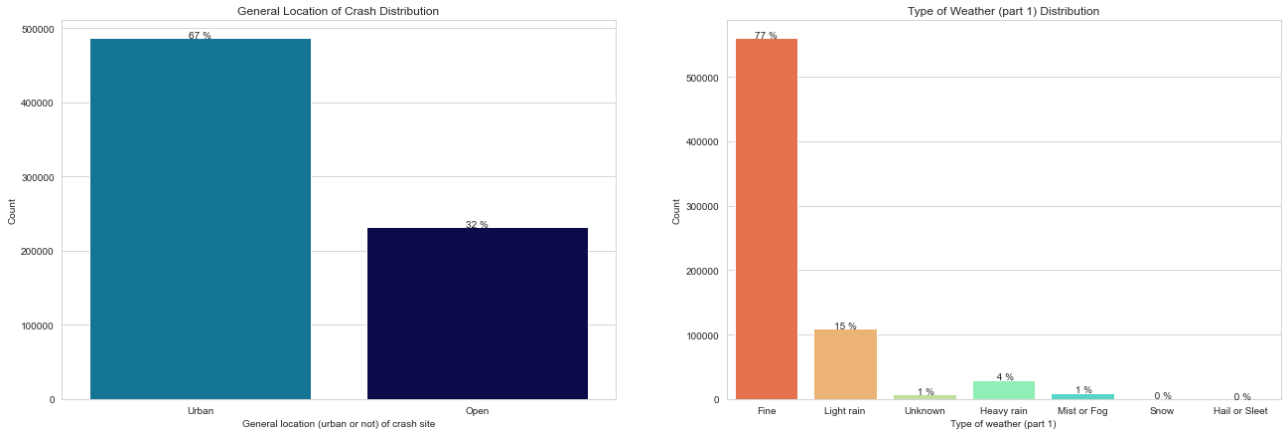

Figure 4.8: Values distribution for ‘urban’ and ‘weatherA’ (% based on full set of values)

- ‘weatherB’ = second data field to describe weather conditions at crash time and site. The data shows that most of the crashes (699312 samples representing 97.3%) are under the class ‘Null’. As per the previous weather feature, this class ‘Null’ does not appear in the feature description, Appendix A, instead an ‘Unknown’ class. As before, it is safe to assume that the two classes refer to the same type of conditions/data. Hence, the ‘Null’ values were renamed ‘Unknown’. Additionally, there is one sample under the class ‘None’, that was renamed as ‘Unknown’. A graph of the data is shown below ().

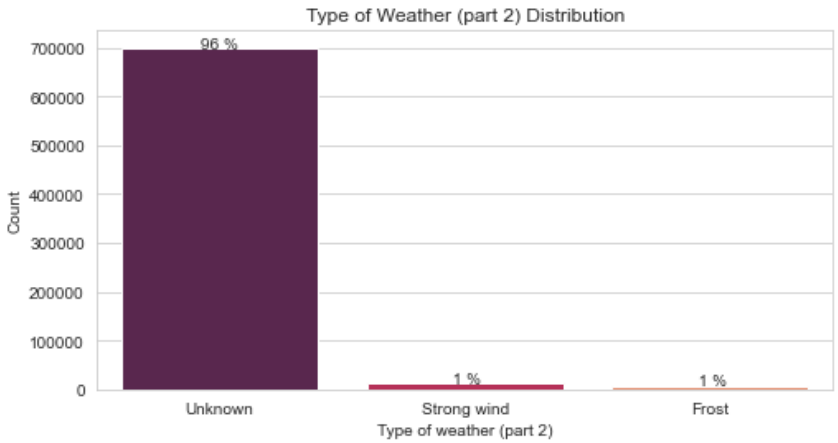

Figure 4.9: Values distribution for ‘weatherB’ (% based on full set of values)

4.3 Data wrangling – Converting data to correct type

After reading the data file, python/pandas by default assigned all numerical features as type float and the categorical features as type object. To have a better representation of the data all the numerical features were converted to type integer, as most of them represent just counters and a few additional ones take only integer values.

The categorical features were converted to type category, including 'NumberOfLanes' and 'speedLimit'. The category type is not usually referred to in introductory books and courses. However, I found that this is a valid pandas type and it is explicitly defined to use for categorical features. The type category appears and acts as a string in most cases but internally is represented by integers. This allows the data to be sorted in a custom order and to more efficiently store the data. For more information refer to the pandas documentation here.

4.4 Data wrangling – Summary

In the previous sections, it was depicted how the dataset was examined and modified, in most instances to clean the data. At this point, there was confidence that the dataset was a consistent set, without any NaN values and where all the values corresponded to valid feature classes. The samples containing initial NaN values were removed from the dataset or the NaN values were replaced with appropriate values. Similarly, other values such as 'Null' have been dealt with similarly. A summary of the changes is shown below:

Table 4.1 Changes to the dataset

	Number of samples	Number of features
Initial size dataset	725548	72
New size dataset	718886	53
Difference	**6662**	**19**

The removal of the 6662 samples was done keeping in mind that there are very few total 'Fatal Crash' samples in the dataset and it was important to remove as few as possible. To accomplish this the distribution of the candidate samples for removal with respect to 'CrashSeverity' was used to evaluate the impact. Out of the 6662 samples removed only 13 corresponded to 'Fatal Crash' samples.

5. Data Exploration – Features Correlation

5.1 Visual exploration

To analyse the relationships among the categorical features, and especially their relationship with the target feature ('crashSeverity') a series of plots were used. The plots are of the type categorical vs categorical. They present the values distribution of the features different classes into the classes of the target feature.

Figure 5.1 shows the relationship of the target feature with 'crashSHDescription' and 'DirectionRoleDescription'. The plots on the left contain all the categories, while the ones on the right contain only the distributions for 'Serious Crash' and 'Fatal Crash' allowing to 'zoom' into these categories values. Both features display expected distributions, starting with a large number of samples for 'Non-Injury Crash' and decreasing quite rapidly across the other classes. From these graphs is difficult to extract any conclusion regarding the relationship between the features and the target one, as there are no changes to the general pattern described above.

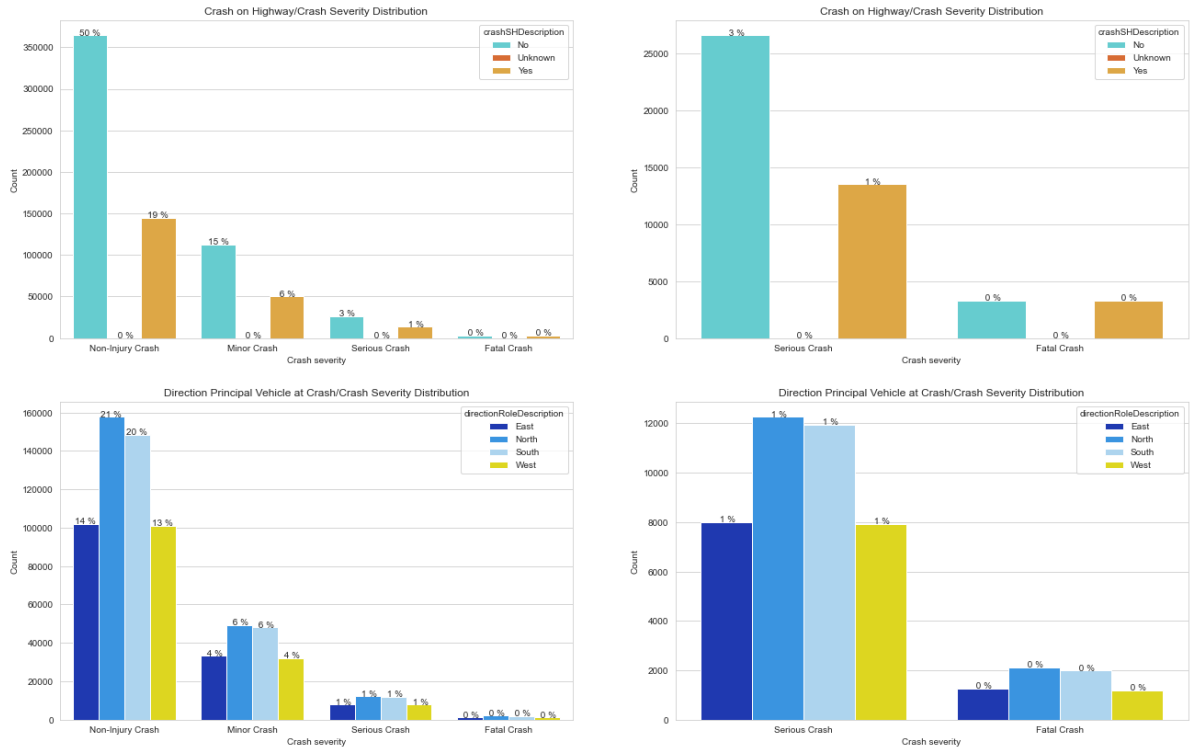

Figure 5.1: Relationship of 'crashSHDescription' and 'DirectionRoleDescription' with 'crashSeverity'

Figure 5.2 shows the graphs for the relationship of 'crashSeverity' with 'streetLight', 'tlaName', 'trafficControl' and 'urban'. As with the previous plots, it is not possible to draw any conclusion as they follow the decaying pattern describe previously. Graphs for the rest of categorical features follow a similar pattern and can be found in Appendix B.

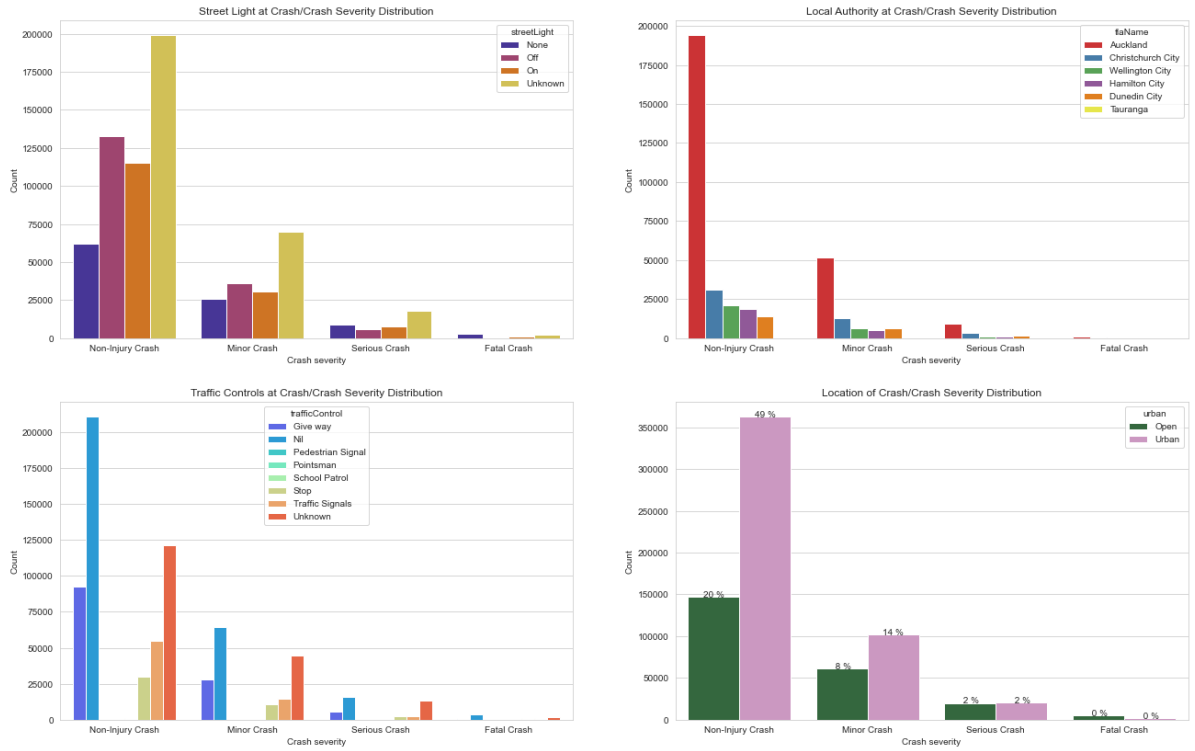

Figure 5.2: Relationship of 'streetLight', 'tlaName', 'trafficControl' and 'urban' with 'crashSeverity'

After examining all the plots, the conclusion is that in this study, the visual inspection of the categorical features does not allow to identify features that could have strong relationships with the target feature.

5.2 Correlation analysis

Another method to explore the relationships among the features is to analyse their correlation matrix. The Chi-Square Test of Independence algorithm can be used to analyse datasets that only have categorical features, so it could be used to examine that part of the dataset here. However, it could be better if a correlation methodology was able to handle categorical and numerical types.

After conducting some literature research it was possible to identify two new methodologies that can manage datasets with categorical and numerical(continuous) features. These two methods are:

- Phi_K: this is a new and practical correlation coefficient based on several refinements to Pearson's hypothesis test of independence of two variables. The combined features of Phi_K form an advantage over existing coefficients. First, it works consistently between categorical, ordinal and interval variables. Second, it captures non-linear dependency. Third, it reverts to the Pearson correlation coefficient in case of a bi-variate normal input distribution. These are useful features when studying the correlation matrix of variables with mixed types. The methodology is available as a Python package and can be easily installed. For additional information:
    - Package website: Phi_K Correlation Analyser Library
    - Technical paper: A new correlation coefficient between categorical, ordinal and interval variables with Pearson characteristics. M. Baak, R. Koopman, H. Snoek, S. Klous. Computational Statistics and Data analysis. March 2019.
- Dython: this is another new methodology that calculates the correlation among the features in a dataset with both categorical and continuous features. The method uses Pearson's R for the analysis of continuous/continuous cases, correlation ratio for categorical/continuous cases and Cramer's V or Theil's U for categorical/categorical. The methodology is available as a Python package and can be easily installed. For additional information:
    - Package website: Dython
    - Technical paper: The Search for Categorical Correlation. Shaked Zychlinski. Towards Data Science. Feb. 2018.

5.2.1 Correlation matrix – categorical features

As a first look at the correlations, the categorical features were analysed to explore the two new methods and compare against the traditional Chi-Square Test of Independence.

The list of categorical features is: 'crashSeverity', 'crashSHDescription', 'directionRoleDescription', 'flatHill', 'holiday', 'light', 'NumberOfLanes', 'roadCharacter', 'roadLane', 'roadSurface', 'speedLimit', 'streetLight', 'tlaName', 'trafficControl', 'urban', 'weatherA' and 'weatherB' and a sub-dataset was created with just these features.

The results for Phi_K are shown below. Phi_K range of values is 0-1, with 0 no correlation and 1 very strong correlation.

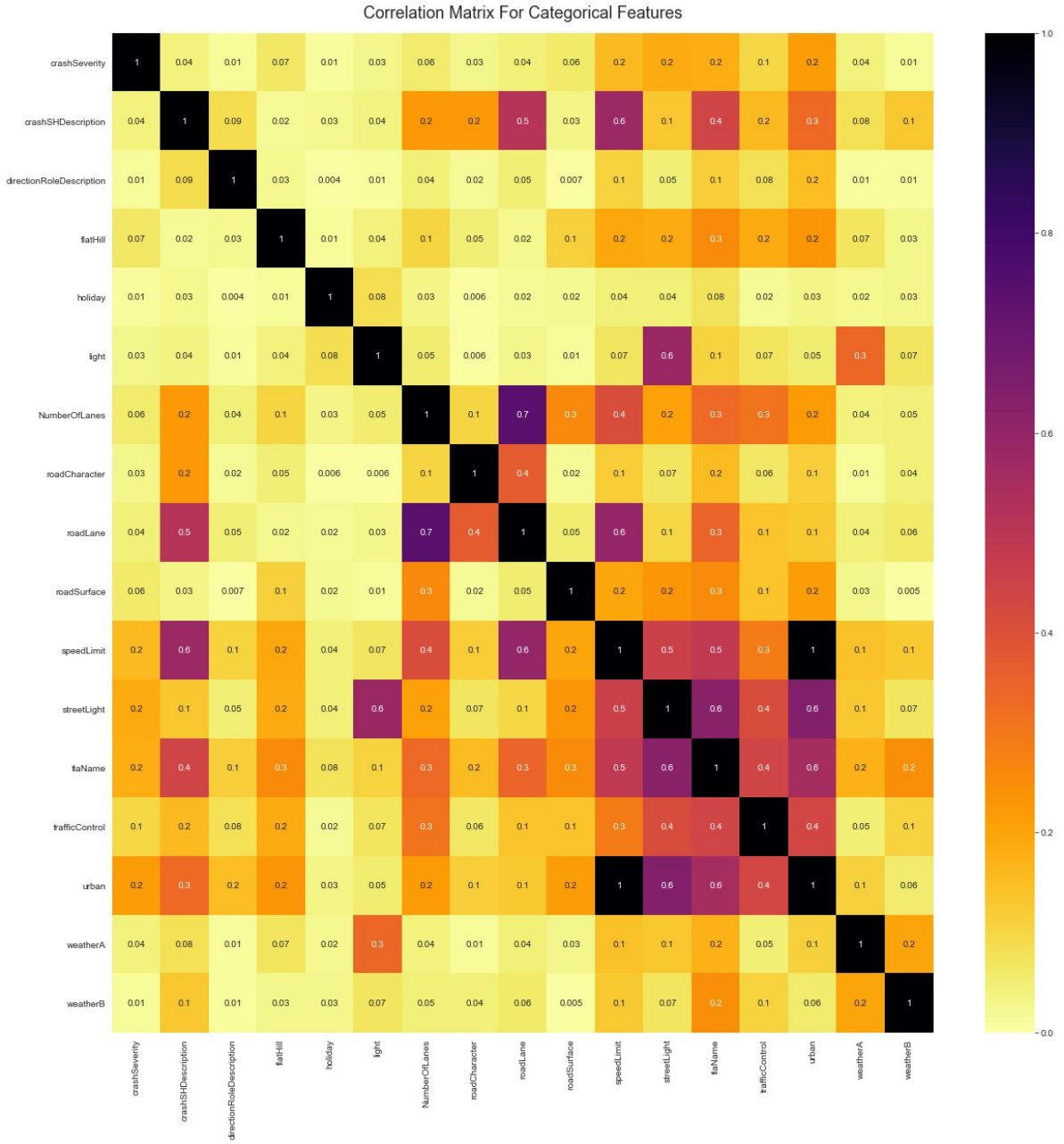

Figure 5.3: Phi_K correlation matrix for the categorical features

From the correlation matrix is possible to conclude:

- There are not strong correlations of the target feature('crashSeverity') with any of the other categorical features.
- The matrix correctly recognises very strong correlations as self, 'urban' with 'speedLimit' due to the way the 'speedLimit' NaN values were resolved and 'roadLane' with 'NumberOfLanes'.
- These results support the findings obtained with the visual exploration, there are no strong correlations with the target feature.
- There are a few features that have a low level of correlation with the target feature('crashSeverity'): 'speedLimit', 'streetLight', 'tlaName' and 'urban'.

The Phi_K methodology offers the possibility to calculate a significance matrix, which examines the statistical significance of the correlation values calculated before. This matrix was calculated and it is included in Appendix C. The significance matrix corroborates that the relationships of the features listed above with the target feature are important.

The next step was to use the Dython methodology. The resultant correlation matrix is shown below, with the range of values between 0 and 1, with 0 no correlation and 1 very strong correlation.

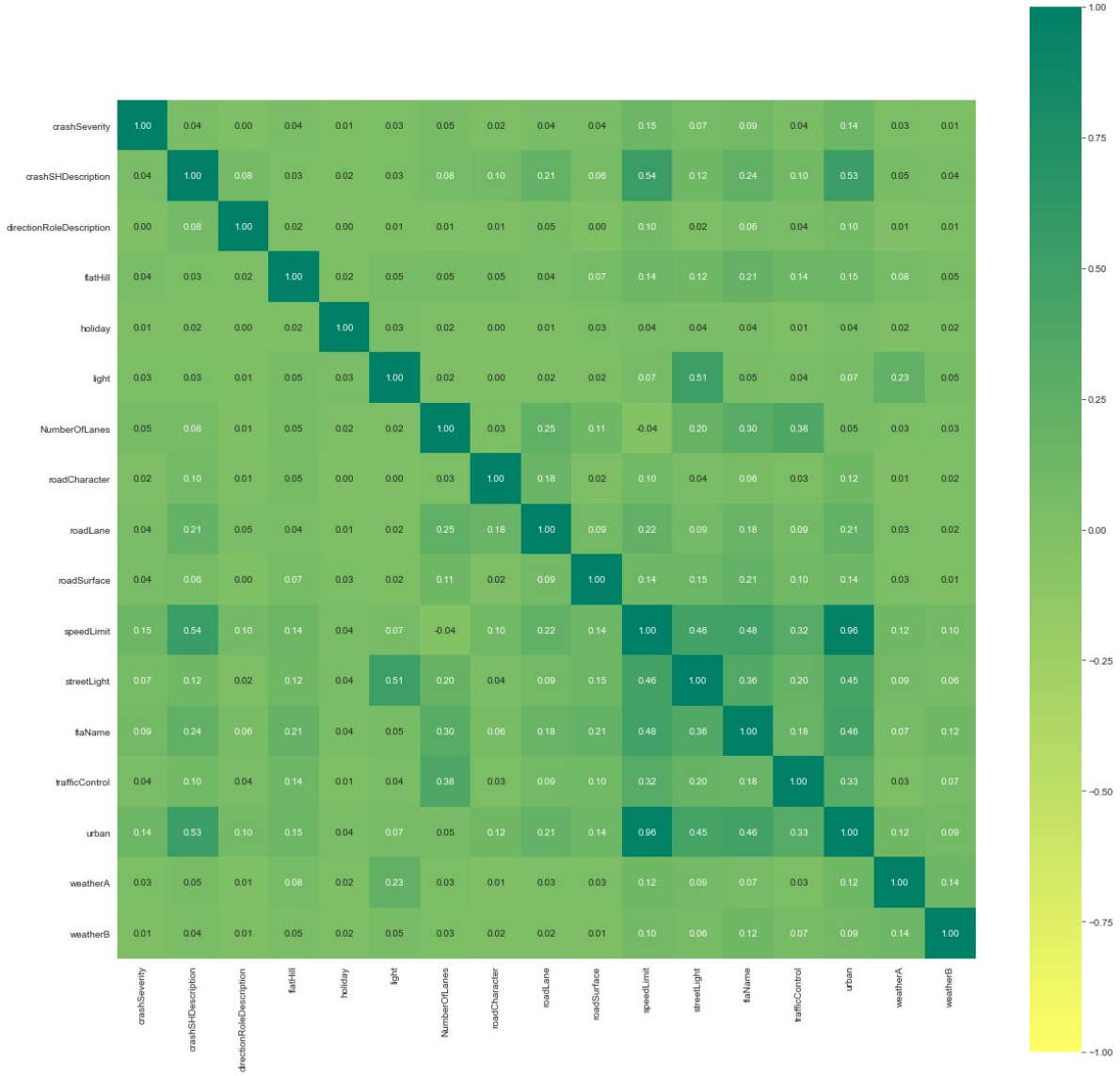

Figure 5.4: Dython correlation matrix for categorical features

The results with the Dython methodology resemble closely the results found with the Phi_K algorithm. However, appears as if this method is not able to discriminate as much as the Phi_K method.

To validate the previous methodologies the Chi-square Test of Independence was used, as it is the method usually employ to identify relationships among categorical features. To use the *chi2* function available in sklearn, the target feature was separated from the other categorical features and assigned to a result target vector, and the rest of features as an array of sample vectors. Additionally, all the categorical values were converted to integers, because this *chi2* function accepts only numerical inputs. For this instance, the 'OrdinalEncoder' from the package 'category_encoders' was selected.

The ‘category_encoders’ package is a popular one, providing a large variety of methods to transform categorical features into numeric ones. The package can be found on its website and it is easy to install. For this instance, the ordinal encoder was selected for its simplicity and to avoid extra variables, as with other encoders.

The output of the function is two numerical vectors: ch2 and pval, with pval the significance, which provides a clearer view of the results. The results were:

Table 5.1 Chi-Square pval for categorical features

Feature	pval
crashSHDescription	3.12381306e-055
directionRoleDescription	1.02361169e-002
flatHill	2.08343564e-039
holiday	5.60877767e-008
light	4.50553488e-039
NumberOfLanes	1.04678137e-048
roadCharacter	1.06205056e-003
roadLane	1.69101500e-037
roadSurface	1.20564885e-004
speedLimit	0.00000000e+000
streetLight	7.07206817e-109
tlaName	0.00000000e+000
trafficControl	0.00000000e+000
urban	0.00000000e+000
weatherA	3.00744255e-019
weatherB	1.57564464e-002

The results from the Chi-Square method agree with results that the Phi_K and Dython algorithms produced. The only exception is the feature ‘trafficControl’ that has been flagged by Chi-Square, the Dython method does not recognise it and Phi_K recognises it but at a very low level. Given these results, it was decided to add ‘trafficControl’ to the selected categorical features.

5.2.2 Correlation matrix – all features

The previous section showed that the two new correlation matrix methodologies do a good job identifying features with important relationships to the target feature. Given the fact that they are capable to calculate the correlation for categorical and continuous features,

was decided to use them to select features with the full set of features. Besides these two new methods, it was not possible to find any other previous ones.

The resultant correlation matrix for the Phi_K is:

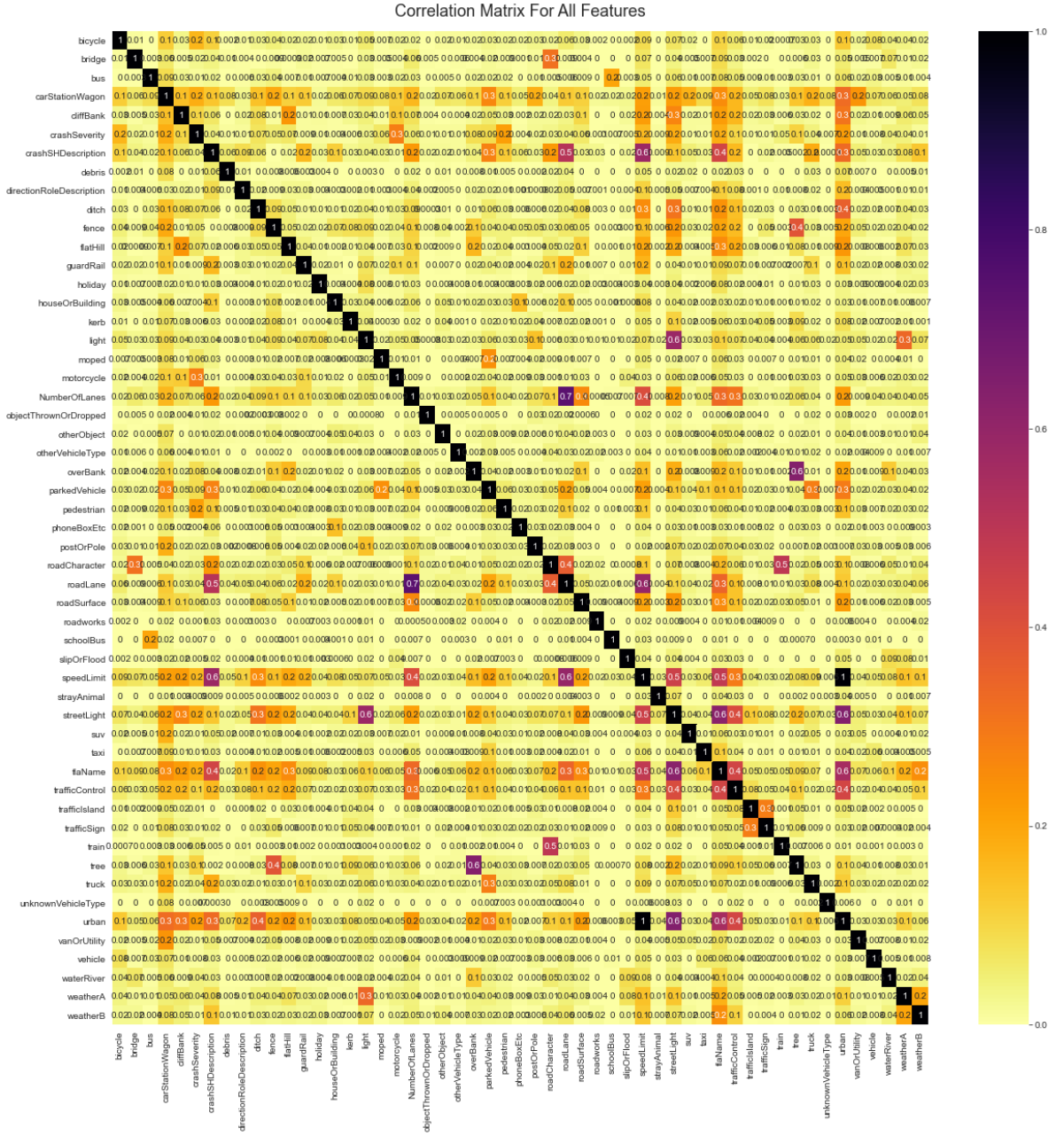

Figure 5.5: Phi_K correlation matrix for the full set of features

The results from the Phi_K matrix show:

- As with the previous matrices, there are not strong correlations of the target feature('crashSeverity') with any of the other features.
- As with the previous Phi_K for categorical features, this matrix correctly recognises very strong correlations as self, 'urban' with 'speedLimit' due to the way the 'speedLimit', 'light' with 'streetLight' and 'overBank' with 'tree'.
- These results support the findings obtained with the visual exploration and the previous correlation matrices, there are not strong correlations with the target feature.

- There are a few numerical features that have a low level of correlation with the target feature('crashSeverity'): 'bicycle', 'carStationWagon', 'cliffBank', 'motorcycle' and 'pedestrian'. These add to the categorical ones that were selected before.

The Dython matrix results are:

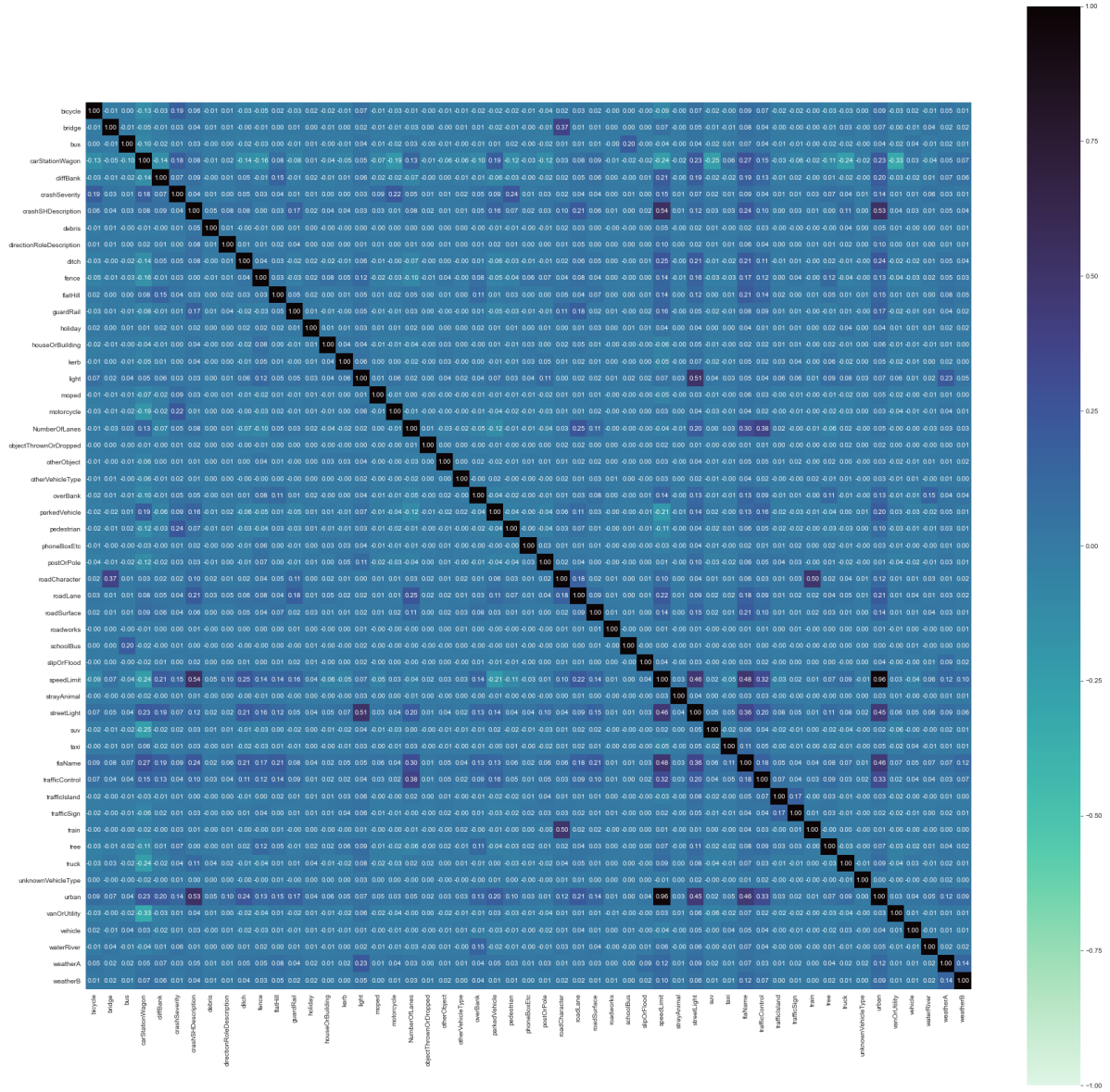

Figure 5.6: Dython correlation matrix for the full set of features

The Dython matrix for the full set of features corroborates the results from previous methods and shows again that the Dython matrix can not distinguish as well as the Phi_K matrix method.

5.3 Features selection

Based on the analysis presented in the previous sections, it is possible to determine that the visual examination did not show any clear relationships and the correlation matrices show some few weak relationships. This lack of strong relationships between the target feature 'crashSeverity' and other features most probably will represent a problem for the classification machines to develop an accurate model of the system.

Supported by the results from the Phi_K method and the Chi-Square the list of featured features to use for the rest of the analysis is: 'bicycle', 'carStationWagon', 'cliffBank', 'motorcycle', 'pedestrian', 'speedLimit', 'streetLight', 'tlaName', 'trafficControl' and 'urban'.

Using this set of features it was possible to reduce the dataset to reflect only the selected features and the target feature.

6. Imbalanced Dataset – Three Variations

At beginning of the analysis it was noticed that the target feature 'crashSeverity' is highly imbalanced, the composition of the classes is (a graph of it is shown Figure 2.1):

Table 6.1 Class composition for 'crashSeverity'

Class	Number of samples (percentage)
Non-Injury Crash	509326 (71.01 %)
Minor Crash	162882 (22.53 %)
Serious Crash	40104 (5.55 %)
Fatal Crash	6574 (0.91 %)

The table shows that the two minority classes, 'Serious Crash' is below 10%, 'Fatal Crash' is below 1% and the majority class 'Non-injury Crash' has close to 71% of the data. This represents a very large imbalance among the four classes. It is possible to use the dataset in this form and proceed with train/test and modelling. However, when one or more classes are very rare, like in this case, many models will not work too well at identifying the minority classes. Additionally, it might appear that the model is performing correctly during the testing, but it might fail and not generalise during production after is exposed to larger datasets.

In this case, the imbalance is due to the nature of the data itself and can not be resolved by getting additional data. There are three things that can help to mitigate the issue: 1) during the building of the dataset resample the data, 2) during modelling use a variety of algorithms and 3) use a variety of metrics to measure the performance of the models. Hence, the first step is to resample the dataset.

There are two main methods to resample or attempt to balance the data:

- Under-sampling: this method is very simple, the idea is to reduce or remove samples to bring the class with more counts(majority class) to a level that is more in balance with the class with the fewer counts(minority class). There are several ways of accomplishing this, but the most popular is to remove randomly samples that belong to the majority class.
- Over-sampling: this method increases the number of samples belonging to the minority class to a level that is more in balance with the majority class. There are several techniques to do this, such as randomly duplicate samples from the minority class. Given the great difference between classes for the crash dataset, it has been selected to combine the use of a synthetic sample generator to increase samples and the reduction of the majority classes.

6.1 Splitting the dataset into training and testing subsets

Before trying to balance the dataset, it is important to split the dataset into the subsets for training and testing. This should be done before to avoid 'bleeding' from the testing set to the training set. The splitting was carried out with a split of test=0.25. After the splitting, the composition of the training target feature was evaluated, with the following results.

Table 6.2 Class composition for 'crashSeverity' in training set after splitting

Class	Number of samples (percentage)
Non-Injury Crash	381994 (70.85 %)
Minor Crash	122161 (22.66 %)
Serious Crash	30078 (5.58 %)
Fatal Crash	4931 (0.91 %)

The table shows that after splitting the dataset the relative composition of all the classes is quite close to the original one.

6.2 Creating the balanced datasets

To have a better understanding of the benefits of balancing the dataset, three datasets were used to train and test the models. The three sets are:

- The original imbalanced set, named 'Full set'. Corresponds to the train set output from the split phase.
- One balanced set created using under-sampling, named 'Under-sampled'
- One balanced set created using oversampling, named 'Over-sampled'

To facilitate the creation of the balanced sets the 'imbalanced-learn' library was installed and some of its packages imported. Imbalanced-learn is a python library offering a number of re-sampling techniques commonly used in datasets showing strong between-class imbalance. For more information go to the imbalanced-learn documentation website.

6.2.1 Under-sampling variations

To generate this variation it is necessary to reduce the number of samples of the classes that have a larger number of them, compared with the number of samples of the minority class 'Fatal Crash'. Therefore the samples of the classes 'Non-Injury Crash', 'Minor Crash' and 'Serious Crash' were levelled with the number of 'Fatal Crash' samples (4931). There are several methodologies that can be used to reduce the number of samples. For this analysis, the 'RandomUnderSampler' method was used. This method is a fast and easy way to balance the data by randomly selecting a subset of data for the targeted classes. Once the method was applied it was confirmed the amount of samples for all the classes was 4931. The under-sampled dataset (X and y) was properly labelled, to differentiate from the other variations.

6.2.2 *Over-sampling variations*

For this variation, it is necessary to increase the sample numbers that correspond to the minority classes and levelled them with the majority class, 'Non-injury Crash'. However, because of the high imbalance of the dataset, there are very large sample numbers differences between the largest (381994) and the smallest (4931) classes. Hence, it is not reasonable to create more than 377000 new samples out of the few available, without introducing overlapping and other errors. Instead, the amount of 50000 was selected as a viable amount to set for all the classes. To achieve this the samples were increased for "Fatal Crash' and 'Serious Crash' and reduced for 'Non-Injury Crash' and 'Minor Crash'.

There are several methodologies that can be used to increase the number of samples, such as duplicating samples and others. If too many samples are duplicated there is the risk of overfitting, as a lot of samples would be the same. A solution to overfitting is creating synthetic samples, a popular method is SMOTE (Synthetic Minority Over Sampling Technique), which varies attributes of the observations to create new synthetic samples. Here a variation of the later called SMOTEC is used. This method is available in the library 'imbalanced-learn' and is designed to manage categorical features. The SMOTEC methodology was applied to the 'Fatal Crash' and 'Serious Crash' classes to create additional samples and level them to 50000 samples each. After this, the second part is to reduce the sample numbers of the 'Non-Injury Crash' and 'Minor Crash' classes to 50000 samples as well. This was done using the 'RandomUnderSampler' method, in a similar way as done in the previous section. After this, all the classes had 50000 samples.

6.2.3 *Encoding the categorical features*

The following step is to encode the categorical features. Most of the machine learning algorithms can not handle categorical variables unless they are converted to numerical values. Additionally, many algorithm's performances vary based on how categorical variables are encoded.

As indicated before the library category_encoders was used to encode the dataset. The encoding method selected was the 'OneHotEncoder', this method maps each category to a vector that contains 1 and 0 denoting the presence or absence of the feature. The number of vectors depends on the number of categories for features. This method produces a large number of additional columns, if the number of classes/categories is very high for the feature. This effect can slow down the learning period.

Using the 'OneHotEncoder' the three variations of the train datasets, 'Full set', 'Over-sampled' and 'Under-sampled', were encoded and the test dataset. Initially, the datasets had 10 columns, after the encoding the number of columns jumped to 100.

7. Classification/machine learning algorithms

After following all the previous steps of cleaning, deleting, examining, selecting, splitting and encoding the datasets, the train sets were ready to start training the machine learning algorithms and evaluate their performance against the test set. As mentioned previously, one of the ways to overcome issues related to the imbalance of the dataset is to use various classifiers as some can be sensitive to the imbalance. In order to study a variety of classification methods the following six were selected:

- **Decision Tree Classifier** (DecisionTreeClassifier()). Decision trees are widely used models for classification and regression tasks. Essentially, they learn a hierarchy of "if-else" questions, leading to a decision. It works by breaking down a dataset into smaller and smaller subsets based on "if-else" criterium. Different sorting criteria will be used to divide the dataset, with the number of examples getting smaller with every division.

- **Histogram-based Gradient Boosting Classification Tree** (HistGradientBoostingClassifier()). This is an 'ensemble' classifier. Ensemble methods combine multiple estimators into one model to improve the generalisation and robustness compare to a single estimator. In boosting methods, base estimators are built sequentially and one tries to reduce the bias of the combined estimator.
- **Complement Naive Bayes Classifier** (ComplementNB()). It is an adaptation of the standard Naive Bayes algorithm that is particularly suited for imbalanced data sets. Specifically, CNB uses statistics from the complement of each class to compute the model's weights.
- **k-Nearest Neighbor Classifier** (KNeighborsClassifier()). The k-Nearest Neighbors (kNN) algorithm is arguably the simplest machine learning algorithm. Building the model only consists of storing the training dataset. To make a prediction for a new test point, the algorithm finds the closest data points in the training dataset, it "nearest neighbours". It checks the distance from the test point to the known training values. The group of data points/class that would give the smallest distance between the training points and the testing point is the class that is selected.
- **Linear Support Vector Classifier** (LinearSVC()). This is a linear support vector machine (linear SVMs). Support Vector Machines work by drawing a line between the different clusters of data points to group them into classes. Points on one side of the line will be one class and points on the other side belong to another class. The classifier will try to maximize the distance between the line it draws and the points on either side of it, to increase its confidence in which points belong to which class. When the testing points are plotted, the side of the line they fall on is the class they are put in.
- **Linear Discriminant Analysis** (LinearDiscriminantAnalysis()). This method finds a linear combination of features that separates two or more classes of objects or events. As a linear classifier, it separates classes using a line, a plane or a hyperplane. It works by reducing the dimensionality of the dataset, projecting all of the data points onto a line. Then it combines these points into classes based on their distance from a chosen point or centroid.

A function was written to loop over the three different datasets and the six classification methods, collecting performance metrics at the class level (precision, recall and f1-score) and at average level (precision, recall, f1-score, accuracy and balanced accuracy) across the classes.

7.1 Performance metrics

The general practice with all classification problems is to use multiple evaluation metrics, and applicable a confusion matrix to measure their performance. A single evaluation metric like a f1-score can make it look like the algorithm is doing well however, it can hide issues such as poor performance with False Positives or False Negatives.

The figure below (Figure 7.1) shows the confusion matrix for a binary classification case and its relation to metrics such as precision, recall and f1-score. By definition, a confusion matrix is a summary of prediction results on a classification problem. The number of correct and incorrect predictions are summarized with count values and broken down by each class. The matrix gives an insight not only into the errors being made by the classifier but more importantly the types of errors that are being made. It is this breakdown that overcomes the limitation of using classification metrics such as accuracy.

As seen in Figure 7.1 each row corresponds to an actual (true) observations and each column corresponds to a predicted one. Hence, four quadrants show the distribution of the samples between the correct predictions (diagonal values – TP: true positive, TN: true negative) and

the error in the predictions (values off-diagonal). These errors are divided into two categories: 1) values that are positive, but were erroneously classified as negative, false negatives (FN) and, 2) values that are negative, but were erroneously classified as positive, false positives (FP).

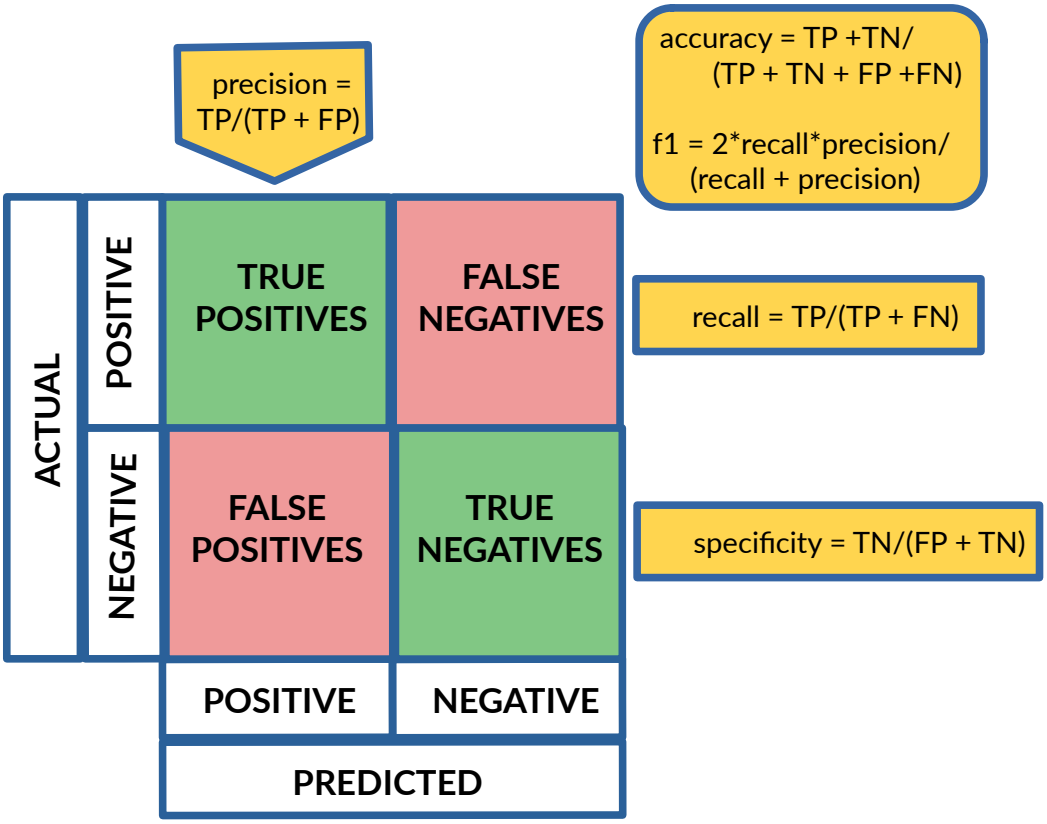

Figure 7.1: Confusion matrix and classification metrics

Based on the matrix it is possible to look at some of the classification metrics:

- **Accuracy**: it is the ratio of correct predictions to total predictions made.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Equation 7.1. Accuracy

- **Precision**: it is a measure of a classifiers exactness. A low precision can also indicate a large number of False Positives.

$$precision = \frac{true\ positives}{(true\ positives + false\ positives)}$$

Equation 7.2. Precision

- **Recall**: it represents the ability of the classifier to find all the relevant cases within a dataset. Recall can be seen as a measure of a classifiers completeness. A low recall indicates many False Negatives.

$$recall = \frac{true\ positives}{(true\ positives + false\ negatives)}$$

Equation 7.3. Recall

- **f1-score**: it is the harmonic mean of precision and recall, taking both metrics into account. It conveys the balance between the precision and the recall metrics.

$$f1 = \frac{2 \times recall * precision}{recall + precision}$$

Equation 7.4. f1-score

For the crash severity analysis, all these metrics were used. Given that the crash severity analysis is a multiclass classification problem, it would be better to use the metrics at the class level instead of average ones across the classes, to measure the performance of the classifiers. This is even more critical for imbalanced datasets, which requires the class level metrics, as mentioned previously and that applies to the case here.

Internally the 'loop_clasf_eng' function calculates precision, recall and f1-score values for both, per class and average. Additionally, it calculates the accuracy and balanced accuracy, with the later been a better metric to deal with imbalanced datasets. Balanced accuracy is defined as the average of recall obtained on each class. During the construction and testing of this notebook other average metrics were studied, such as Matthews correlation coefficient, hamming loss and Cohen kappa score. These metrics provided the same result than the accuracy score, in terms of dataset type and algorithm with the highest score, hence were removed from the calculations.

7.2 Results and analysis

7.2.1 *Average metrics – Accuracy and Balanced accuracy*

Average metrics accuracy and balanced accuracy (Figure 7.2), were used here to determine the general behaviour of the different dataset types and algorithms. The results obtained with accuracy and other average metrics, cannot be used to determine the best dataset type or classifier, because of their inability to weight properly the minority classes. However, they will provide a sensible comparison with the metrics at the class level.

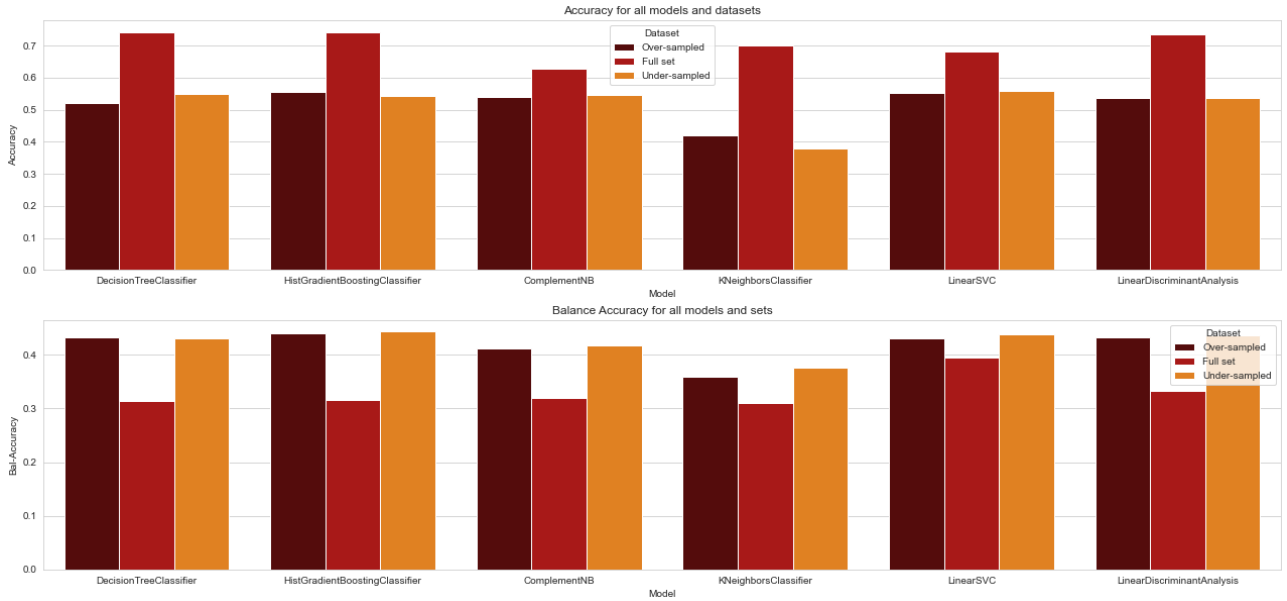

Figure 7.2: Accuracy and balanced accuracy – average metrics

From the plots is clear that the accuracy metric determines that the 'Full set' dataset is the best option, independent of the classifier. This is contrary to the results of balanced accuracy, which shows than the 'Over-sampled' and 'Under-sampled' datasets are better with any algorithm.

This discrepancy is due to the fact that accuracy does not take into account the poor performance of minority classes with the 'Full set' dataset.

In terms of the classifiers, the accuracy metric shows that 'DecisionTreeClassifier', 'HistGradientBoostingClassifier' and 'LinearDiscriminantAnalysis' are the best methods with the 'Full set'. The balanced accuracy metric identifies 'DecisionTreeClassifier', 'HistGradientBoostingClassifier', 'LinearSVC' and 'LinearDiscriminantAnalysis' as the best algorithms with the 'Over-sampled' and 'Under-sampled' datasets.

7.2.2 Best scenario with average values

To determine with more precision the best scenario (combination dataset type and classifier) using average metrics across the classes, the average metrics precision, recall, f1-score and accuracy were calculated and added for each possible scenario and sorted to obtain the best one, based on the total score.

The result was that the scenario with the 'Full set' dataset and the classifier 'HistGradientBoostingClassifier' is the best if the average metrics are used.

7.2.3 Confusion matrix for best scenario average metrics

The confusion matrix for the scenario of 'Full set' dataset and the classifier 'HistGradientBoostingClassifier' was calculated as shown in Figure 7.3.

From the confusion matrix is clearly visible the large majority of the 'Non-Injury Crash' samples are correctly predicted (124150) while only 27 'Fatal Crash' samples are. The matrix indicates that only the 'Non-Injury Crash' has a high percentage of correct prediction (97.5%), while 'Minor Crash' is only 20.9%, 'Serious Crash' 5.2% and 'Fatal Crash' 1.6%. The matrix indicates that most of the misclassified samples from 'Minor Crash', 'Serious Crash' and 'Fatal Crash' are labelled as Non-Injury Crash. These misclassified samples are many times more than the correctly predicted samples. However, the accuracy score is 0.74, which is a good score.

As summary is possible to say that the scenario of 'Full set' dataset and algorithm 'HistGradientBoostingClassifier' is totally skewed and would be of no use for the purpose of this project as it cannot identify properly the most critical classes and misclassify many critical ones as 'Non-Injury Crash'. In the real world, this would translate that for most of the crashes the classifier would dispatch a very basic response unit, when in reality for a large amount of the cases there is need of a more complete emergency response team. This misclassification could have serious implications in the medical attention required by victims at the crash site.

7.2.4 Best scenario with metrics at class level

After the analysis using average metrics, the most appropriate metrics, at the class level, were used to determine the best scenario of dataset type and classifier. For each of the possible combinations (scenarios) of dataset type and model, the values of precision, recall and f1-score were calculated for each class. For each scenario and class the metrics were added and the ones with the highest scores for classes 'Serious Crash' and 'Fatal Crash' were selected and sorted. Finally, the top scenarios for these two classes were compared and the common scenario across these minority classes with the highest score was selected.

From the analysis, the best option to find a compromise with high metrics scores for the performance of the minority classes, `Fatal Crash' and `Serious Crash', is the scenario with 'Under-sampled' dataset and the 'LinearSVC' model.

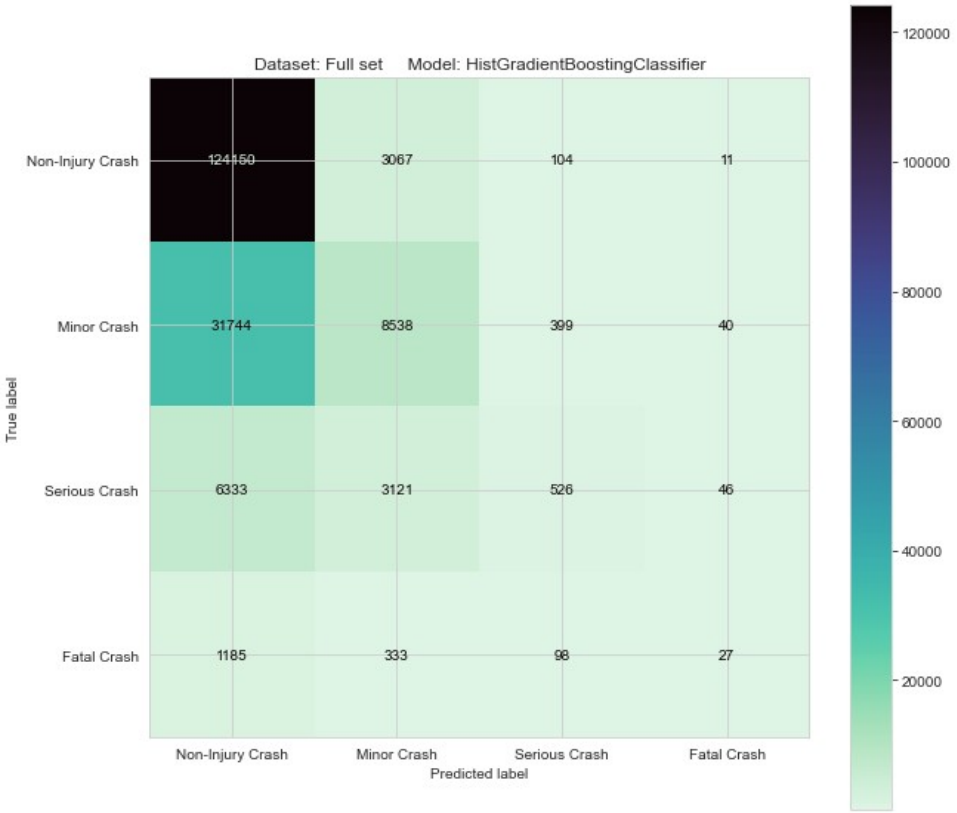

Figure 7.3: Confusion matrix for 'Full set' dataset and classifier 'HistGradientBoostingClassifier'

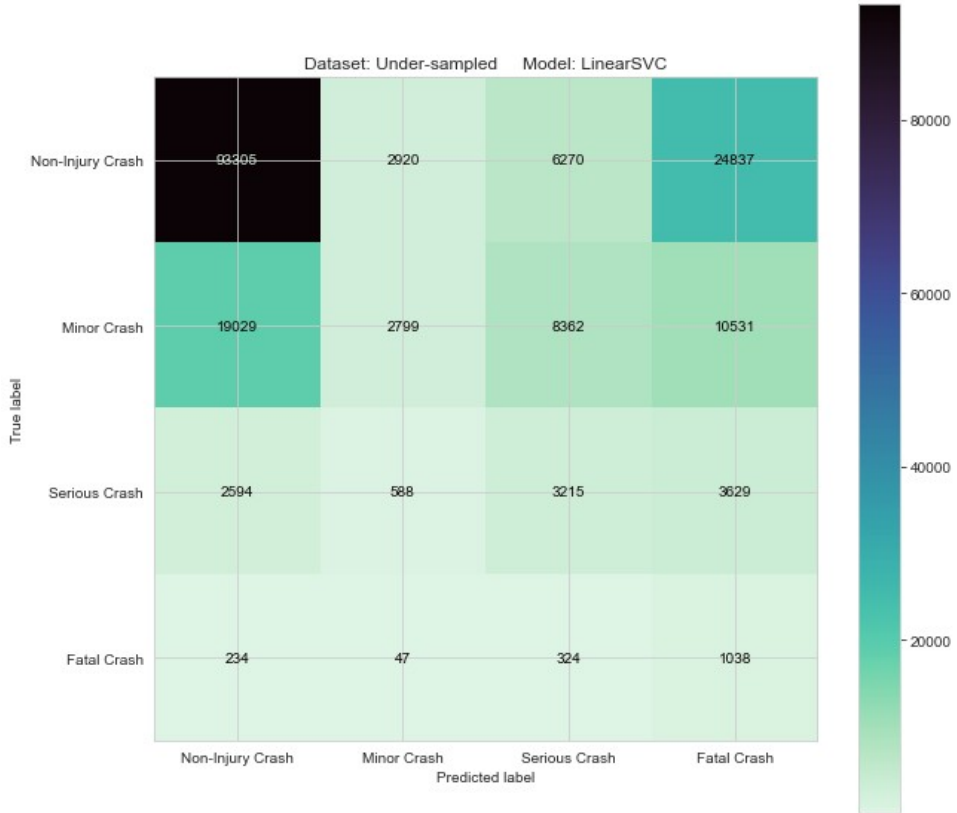

Figure 7.4: Confusion matrix for 'Under-sampled' dataset and classifier 'LinearSVC'

7.2.5 Confusion matrix for the best scenario using metrics at class level

The confusion matrix for the scenario of 'Under-sampled' dataset and the classifier 'LinearSVC' was calculated as shown in Figure 7.4. The results show that using metrics at the class level has improved the performance of 'Serious Crash' and 'Fatal Crash' classes with the detriment of 'Minor Crash' and 'Non-Injury Crash'. One important issue is the high misclassification of other classes as Fatal Crash. In terms of correct predictions, the percentages are now 73.3% for 'Non-Injury Crash', 6.9% for 'Minor Crash', 32% for 'Serious Crash' and 63.2% for 'Fatal Crash'. The percentages for minority classes are a lot better than the previous scenario.

Using this scenario there are 1038 correct predicted 'Fatal Crash' samples compare with only 27 samples from the previous scenario. Even if this number has improved, a new problem is that there are close to 40000 misclassified samples as 'Fatal Crash'. This makes this classifier and companion dataset not reliable as this misclassification means that only 3% of the samples labelled as 'Fatal Crash' are correctly predicted. This would mean that if applied to the real world for most of the cases identified by the classifier as 'Fatal Crash', a full set of emergency services will be dispatched to the crash location and they would not be needed. This would represent a waste of money, resources and critical time of the emergency services.

7.2.6 Comparison of the selected scenario using average metrics vs the selected scenario using metrics at class level

Figure 7.5 and Figure 7.6 are plots showing the precision (blue colours), recall (red colours) and f1-score (green colours) at the class level for the two selected algorithms: 1) HistGradientBoostingClassifier (based on average metrics) and 2) LinearSVC (based on class level metrics). For each classifier the values for the three datasets: 'Full set,' 'Over-sampled' and 'Under-sampled.' On each plot the dataset selected is highlighted. Table 7.1 collects the numerical values of the metrics for the two selections.

Figure 7.5: Results for model HistGradientBoostingClassifier

Comparing values of the HistGradientBoostingClassifier/Full set with the ones from LinearSVC/Under-sampled is possible to notice that the three metrics (precision, recall and f1-score) do not change slightly for the class 'Non-Injury Crash' between the two scenarios, with a reduction of recall and f1-score. All the metrics for 'Minor Crash' are reduced,

especially recall and f1-score. For the two minority classes, 'Serious Crash' and 'Fatal Crash', precision is reduced by an appreciable amount, while recall and f1-score increase by a large amount, especially recall for 'Fatal Crash'. values while the values for class 'Fatal Crash' show a big change for recall. Similarly, there are important changes for class 'Serious Crash'.

These changes corroborate information that was obtained with the confusion matrices and how the scenario selection using the at the class level metrics is able to choose the best possible scenario for the minority classes.

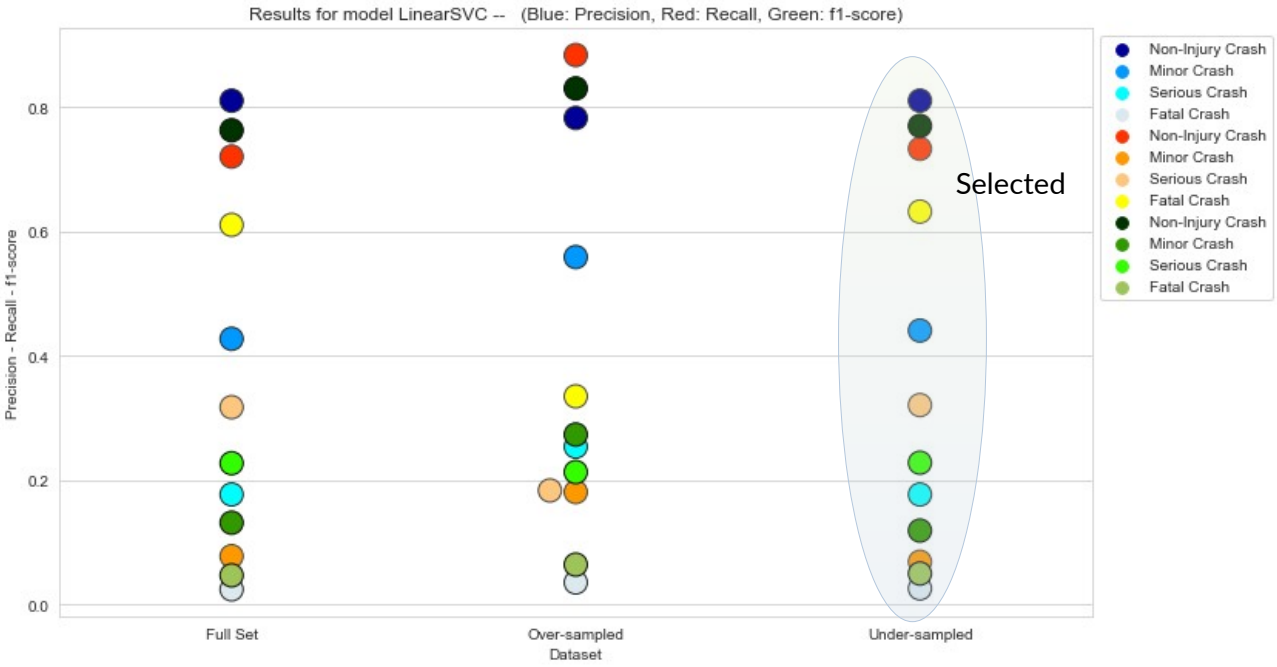

Figure 7.6: Results for model LinearSVC

Table 7.1 Metrics for the two scenarios: HistGradientBoostingClassifier & Full set (selected using average metrics) and LinearSVC & Under-sampling (selected using at the class level metrics)

Metrics						
	Precision		Recall		f1-score	
Classes	HistG & Full set	LinearSVC & Under	HistG & Full set	LinearSVC & Under	HistG & Full set	LinearSVC & Under
Non-Injury Crash	0.75976	0.81021	0.97507	0.73277	0.85405	0.76954
Minor Crash	0.56908	0.44051	0.20888	0.06874	0.30560	0.11892
Serious Crash	0.46731	0.17693	0.05775	0.32067	0.10280	0.22804
Fatal Crash	0.25210	0.02593	0.01826	0.63177	0.03405	0.04981

8. Conclusions

Working in this project has been of great benefit for me, as it helped me to have a better understanding of the different steps of data analysis and how they interact with each other. There were several aspects of the project that provided with new insights around specific items:

- To work with a dataset that had a large number of categorical features as oppose to the usual numerical features. I learned about the categorical type and its use. I had to do some extensive research to find methods that allow analysing the correlation of categorical and numerical features together. This research proved to be very successful and two methods were found that gave very solid results. The variety of encoding methods that can be used on categorical features, so classifiers can understand the data.
- To work with an extremely imbalanced dataset. The need for special methods and techniques to overcome the skewness of the data. The use of resampling to "artificially" balance the dataset by several techniques. The use of a variety of classifiers to avoid any model sensitivity to the imbalance. The use of metrics at the class level and the confusion matrix to measure the performance of the classifiers.
- The need for a variety of plots required to learn a mixture of plotting options. The use of 'heatmaps' to represent correlation matrices and confusion matrices, superposition of swarm plots, a variety of colours and palettes and filtering of plotting classes were just some of the acquired knowledge.

The initial goal of this project was to train a model that would be able to predict the crash severity, 'crashSeverity'. Unfortunately, this goal was not achieved even after applying techniques such as resampling, encoding and using several classifiers. The main reason for the lack of success rests in what was found during the correlations exploration, 'crashSeverity' does not have strong relations with any of the other features. As a consequence, the selected features for the analysis were the best ones from a pool of weakly correlated features (with values of 0.2 out of 1). The results of the classifiers have shown that the selected features have a relation with the target feature, but at the found low level.

From the results but especially from the confusion matrices is possible to conclude:

- Using average metrics the scenario/combination 'HistGradientBoostingClassifier ' and 'Full set' is the best. Some of its characteristics:
    - Accuracy = 0.74155, balanced accuracy = 0.31499
    - From the confusion matrix. 'Non-Injury Crash' recall (% correct prediction) = 97.5%. 'Fatal Crash' recall (% correct prediction) = 1,6%.
    - A large amount of misclassifies are labelled as 'Non-Injury Crash' from the other classes.
    - Very few 'Serious Crash' and 'Fatal Crash' samples are correctly classified. Most of the misclassifies are labelled as 'Non-Injury Crash'.
    - This scenario/combination is not a good selection for the project goal, as most of the samples from the critical classes, 'Serious Crash' and 'Fatal Crash' would be classified as 'Non-Injury Crash'. In the real world, this classifier would dispatch basic services to most crashes when in reality they require a full emergency service team. This would endanger lives and property.

- Using at class level metrics the scenario/combination ‘LinearSVC ‘ and ‘Under-sampled’ is the best. Some of its characteristics:
    - Balanced accuracy = 0.43849, accuracy = 0.55840
    - From the confusion matrix. ‘Non-Injury Crash’ recall (% correct prediction) = 73.3%. ‘Fatal Crash’ (% correct prediction) = 63.2%, almost at the same level that for ‘Non-Injury Crash’.
    - Many misclassifies are labeled as ‘Fatal Crash’ from the other classes. The misclassifies are many times more than the number of ‘Fatal Crash’ samples.
    - A good amount of ‘Serious Crash’, ‘Minor Crash’ and ‘Non-Injury Crash’ are correctly classified.
    - Even if the prediction of ‘Fatal Crash’ and ‘Serious Crash’ classes show an increase by a large amount, this scenario/combination is not a good selection for the project goal. The reason is that a large number of samples are misclassified as ‘Fatal Crash’ from the other classes. The amount of misclassified samples is several times more than the correctly classified ‘Fatal Crash’. In the real world, this classifier would send a full emergency team for minor crashes, this would result in wasting emergency resources, time and money.

Even if the final goal was not achieved, the analysis has determined some conclusions:

- Two new correlation matrices, including categorical and numerical features, were found and proved to deliver good and consistent results.
- The use of resampling proved to help all the classifiers to interpret the dataset better and improve their performance.
- Average metrics, use to measure the classifier performance, do not provide a complete view of the classes and tend to hide the poor performance of minority classes. This is especially important for imbalanced datasets.
- The only average metric, use to measure classifier performance, that provides some clarity when the dataset is imbalanced is the ‘Balanced Accuracy’.
- At class level metrics, use to measure the classifier performance, provide a complete view of the different classes and allows to recognise any class poor performance.
- The confusion matrix has shown to be the best metric to analyse the performance of a classifier. The matrix allows a very complete review of the way the testing set was classified.

Appendix A

This table is based on the information taken from the New Zealand Transport Agency open data repository, in particular from a metafile and webpage with brief descriptions of the features.

NEW ZEALAND AUTOMOBILE CRASH DATA - FEATURES DESCRIPTIONS		
Feature name	**Feature type**	**Feature description**
X	Numerical (geo)	Geographical coordinate – longitude of the crash.
Y	Numerical (geo)	Geographical coordinate – latitude of the crash.
OBJECTID	numerical	Unique ID for the crash.
advisorySpeed	numerical	The advisory speed at the crash site at the time of the crash.
areaUnitId	Numerical (geo)	The unique identifier of an area unit.
bicycle	numerical	Derived feature to indicate how many bicycles were involved in the crash.
bridge	numerical	Derived feature to indicate how many times a bridge, tunnel, the abutments, handrails were struck in the crash.
bus	numerical	Derived feature to indicate how many buses were involved in the crash (excluding school buses which are counted apart).
carStationWagon	numerical	Derived feature to indicate how many cars or station wagons were involved in the crash.
cliffBank	numerical	Derived feature to indicate how many times a cliff or bank was struck in the crash. This includes retaining walls
crashDirectionDescription	Categorical (text)	The direction of the crash from the reference point. Values possible are 'North', 'East', 'South' or 'West'.
crashFinancialYear	numerical	The financial year in which a crash occurred, if known.
crashLocation1	text	Part 1 of the 'crash location'. Maybe a road name, route position (RP), landmark, or other, e.g. 'Ninety Mile Beach'. Used for location descriptions in reports etc.
crashLocation2	text	Part 2 of the 'crash location' (crash_locn). Maybe a side road name, landmark etc. Used for location descriptions in reports etc.
crashRoadSideRoad	Categorical (text)	Indicates whether the principal vehicle in a crash was on the crash road or side road at the time of the crash. Note that 'on side road' can only happen if the crash occurred at an intersection. Possible values are 1: Crash Road, 2: Side Road
crashSeverity	Categorical (text)	The severity of a crash. Possible values are 'Fatal crash', 'Serious crash, 'Minor crash', 'Non-injury crash'. This is determined by the worst injury sustained in the crash at the time of entry.
crashSHDescription	Categorical (text)	Indicates where a crash is reported if occurred on a State Highway. Possible values include 'Yes' where the crash occurred on a SH, 'No' and 'Unknown'.
crashYear	numerical	Year of the crash (yyyy).

debris	numerical	Derived feature to indicate how many times debris, boulders or items dropped or thrown from a vehicle(s) were struck in the crash.
directionRoleDescription	Categorical (text)	The direction (dirn) of the principal vehicle involved in the crash. Possible values are 'North', 'South', 'East' or 'West'.
ditch	numerical	Derived feature to indicate how many times a ditch or water able drainage channel was struck in a crash.
fatalCount	numerical	The number of fatalities for crash of severity 'Fatal crash'.
fence	numerical	Derived feature to indicate how many times a fence was struck in the crash. This includes letterbox(es), hoardings, private roadside furniture, hedges, sight rails, etc.
flatHill	Categorical (text)	Whether the road is flat 'Flat' or sloped 'Hill road'.
guardRail	numerical	Derived feature to indicate how many times a guard or guard rail was struck in the crash. This includes 'New Jersey' barriers, 'ARMCO', sand-filled barriers, wire catch fences, etc.
holiday	Categorical (text)	Indicates whether the crash happened during a holiday and which one. Possible values: ‘None’, ‘Christmas/New Year’, ‘Easter’, ‘Queens Birthday’, ‘Labour Weekend’
houseOrBuilding	numerical	Derived feature to indicate how many times houses, garages, sheds or other buildings were struck in the crash.
intersection	Categorical (text)	Indicates if a crash happened at an 'Intersection', 'At Landmark' or 'Unknown'.
kerb	numerical	Derived feature to indicate how many times a kerb was struck in the crash, that contributed directly to the crash.
light	Categorical (text)	Describes the light at the time and place of the crash. Possible values: 'Bright Sun', 'Overcast', 'Twilight, 'Dark' or ' Unknown'.
meshblockId	numerical	UniqueID number for the meshblock.
minorInjuryCount	numerical	The number of minor injured people.
moped	numerical	Derived feature to indicate how many mopeds were involved in the crash.
motorcycle	numerical	Derived feature to indicate how many motorcycles were involved in the crash.
NumberOfLanes	numerical	The number of lanes on the crash road.
objectThrownOrDropped	numerical	Derived feature to indicate how many times objects were thrown at or dropped on vehicles in the crash.
otherObject	numerical	Derived feature to indicate how many times an object was struck in a crash and the object struck was not pre-defined. This feature includes stockpiled materials, rubbish bins, fallen poles, fallen trees, etc.
otherVehicleType	numerical	Derived feature to indicate how many other vehicles (not included in any other category) were involved in the crash.
overBank	numerical	Derived feature to indicate how many times an embankment was struck or driven over during a crash. This feature includes other vertical drops driven over during a crash.

parkedVehicle	numerical	Derived feature to indicate how many times a parked or unattended vehicle was struck in the crash. This feature can include trailers.
pedestrian	numerical	Derived feature to indicate how many pedestrians were involved in the crash. This includes pedestrians on skateboards, scooters and wheelchairs.
phoneBoxEtc	numerical	Derived feature to indicate how many times a telephone kiosk traffic signal controllers, bus shelters or other public furniture was struck in the crash.
postOrPole	numerical	Derived feature to indicate how many times a post or pole was struck in the crash. This includes light, power, phone, utility poles and objects practically forming part of a pole (i.e. 'Transformer Guy' wires).
region	Categorical (text)	Identifies the local government (LG) region. The boundaries match territorial local authority (TLA) boundaries.Possible values are: ‘Auckland’, ‘Waikato’, ‘Canterbury’, ‘Wellington’, ‘Bay of Plenty’, ‘Otago’, ‘Manawatu/ Wanganui’, ‘Northland’, ‘Hawkes Bay’, ‘Nelson/Marlborough’, ‘Southland’, ‘Taranaki’, ‘Gisborne’, ‘West Coast’.
roadCharacter	Categorical (text)	Describes the general nature of the road. Possible values include 'Bridge', 'Motorway Ramp', ‘Rail xing’, Overpass', 'Speed hump', 'Underpass', 'Tunnel', 'Tram lines' or 'Nil'.
roadLane	Categorical (text)	The lane configuration of the road. Possible values : '1-way', '2-way, 'Median' and 'Off road'.
roadSurface	Categorical (text)	The road surface description at the crash site. Possible values: 'Sealed' or 'Unsealed'.
roadworks	numerical	Derived feature to indicate how many times an object associated with roadworks (including signs, cones, drums, barriers, but not roadwork vehicles) was struck during the crash.
schoolBus	numerical	Derived feature to indicate how many school buses were involved in the crash.
seriousInjuryCount	numerical	The number of seriously injured people.
slipOrFlood	numerical	Derived feature to indicate how many times landslips, washouts or floods (excluding rivers) were objects struck in the crash.
speedLimit	numerical	The speed limit in force at the crash site at the time of the crash.
strayAnimal	numerical	Derived feature to indicate how many times a stray animal(s) was struck in the crash. This feature includes wild animals such as pigs, goats, deer, straying farm animals, house pets and birds.
streetLight	Categorical (text)	The street lighting at the time of the crash. Possible values 'On', 'Off', 'None' or ' Unknown'.
suv	numerical	Derived feature to indicate how many SUVs were involved in the crash.
taxi	numerical	Derived feature to indicate how many taxis were involved in the crash.
temporarySpeedLimit	numerical	The temporary speed limit at the crash site if one exists (e.g. for road works).
tlaId	numerical	The unique identifier for a territorial local authority (TLA). Each crash is assigned a TLA based on where the crash occurred.
tlaName	Categorical (text)	Indicates the local authority who handled the crash. Each region has several TLA.

trafficControl	Categorical (text)	The traffic control signals at the crash site. Possible values are 'Traffic Signals', 'Stop Sign', 'Give Way Sign', 'Pointsman', 'School Patrol', 'Nil' or ' Unknown'.
trafficIsland	numerical	Derived feature to indicate how many times a traffic island, medians (excluding barriers) was struck in the crash.
trafficSign	numerical	Derived feature to indicate how many times traffic signage (including traffic signals, their poles, bollards or roadside delineators) was struck in the crash.
train	numerical	Derived feature to indicate how many times a train, rolling stock or jiggers was struck in the crash, whether stationary or moving.
tree	numerical	Derived feature to indicate how many times trees or other growing items were struck during the crash.
truck	numerical	Derived feature to indicate how many trucks were involved in the crash.
unknownVehicleType	numerical	Derived feature to indicate how many vehicles were involved in the crash (where the vehicle type is unknown).
urban	Categorical (text)	Derived feature using the 'spd_lim' variable. Possible values are 'Urban' (urban, spd_lim < 80) or 'Open' (open road, spd_lim >=80 or 'LSZ')
vanOrUtility	numerical	Derived feature to indicate how many vans or utes were involved in the crash.
vehicle	numerical	Derived feature to indicate how many times a stationary attended vehicle was struck in the crash. This includes broken down vehicles, workmen's vehicles, taxis, buses.
waterRiver	numerical	Derived feature to indicate how many times a body of water (including rivers, streams, lakes, the sea, tidal flats, canals, watercourses or swaps) was struck in the crash.
weatherA	Categorical (text)	Describes the type of weather at the crash time/place. A derived variable using the 'spd_lim' variable. Values that are possible are 'Fine', 'Mist', 'Light Rain', 'Heavy Rain', 'Snow', 'Unknown'.
weatherB	Categorical (text)	Second data field to describe the type of weather at the crash time/place when applicable. Values 'Frost', 'Strong Wind' or 'Unknown'.

Appendix B

Feature correlation - Visual exploration.

The plots in this appendix are part of the visual exploration used to investigate possible relationships between the target feature 'crashSeverity' and other categorical features.

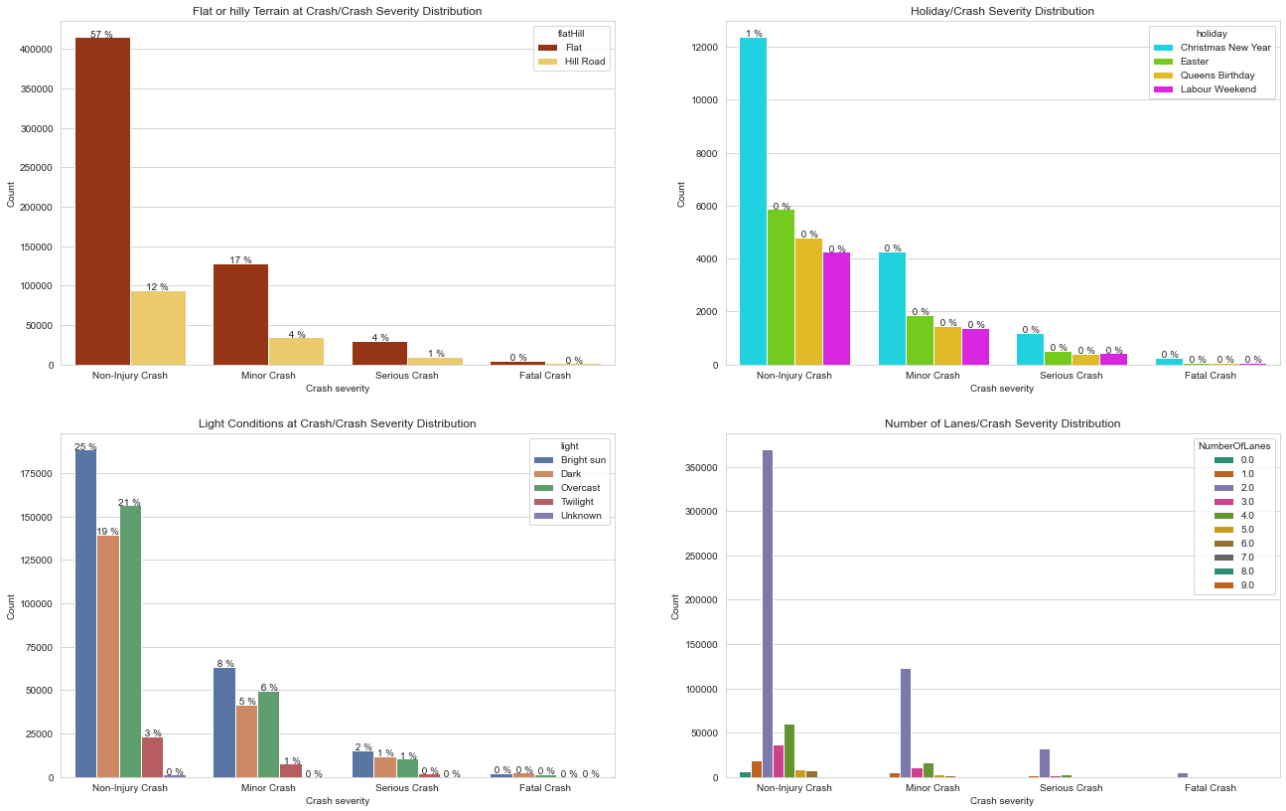

Figure B.1: Plots showing relationship between 'flatHill', 'holiday', 'light' and 'NumberOf Lanes' with the target feature 'crashSeverity'

From the plots is not possible to determine if there is a strong relationship, as all plots follow the same decaying pattern due to the possibility of occurrence of the classes.

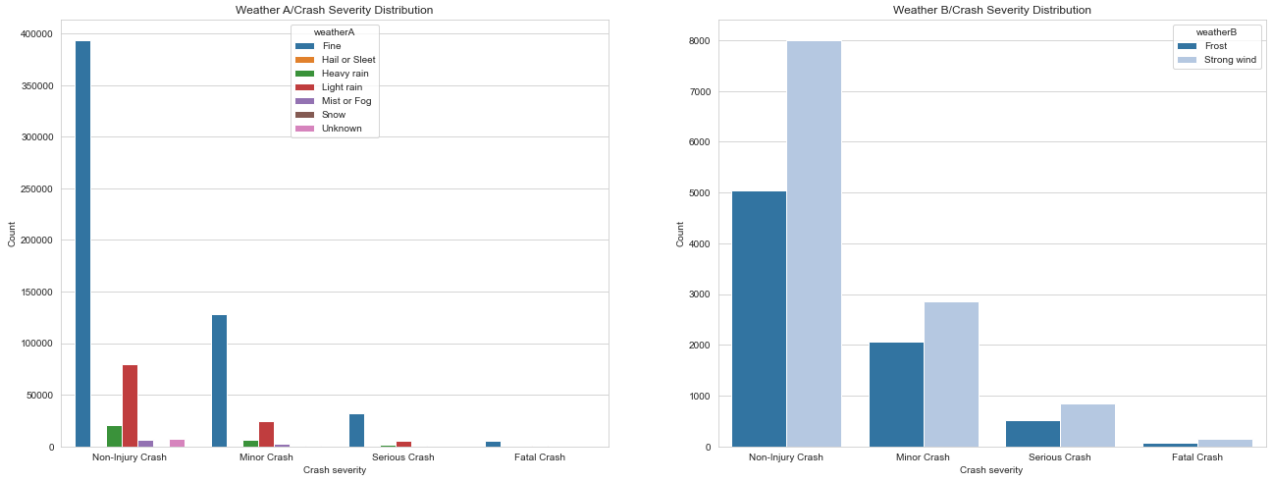

Figure B.2: Plots showing relationship between 'weatherA' and 'weatherB' with the target feature 'crashSeverity'

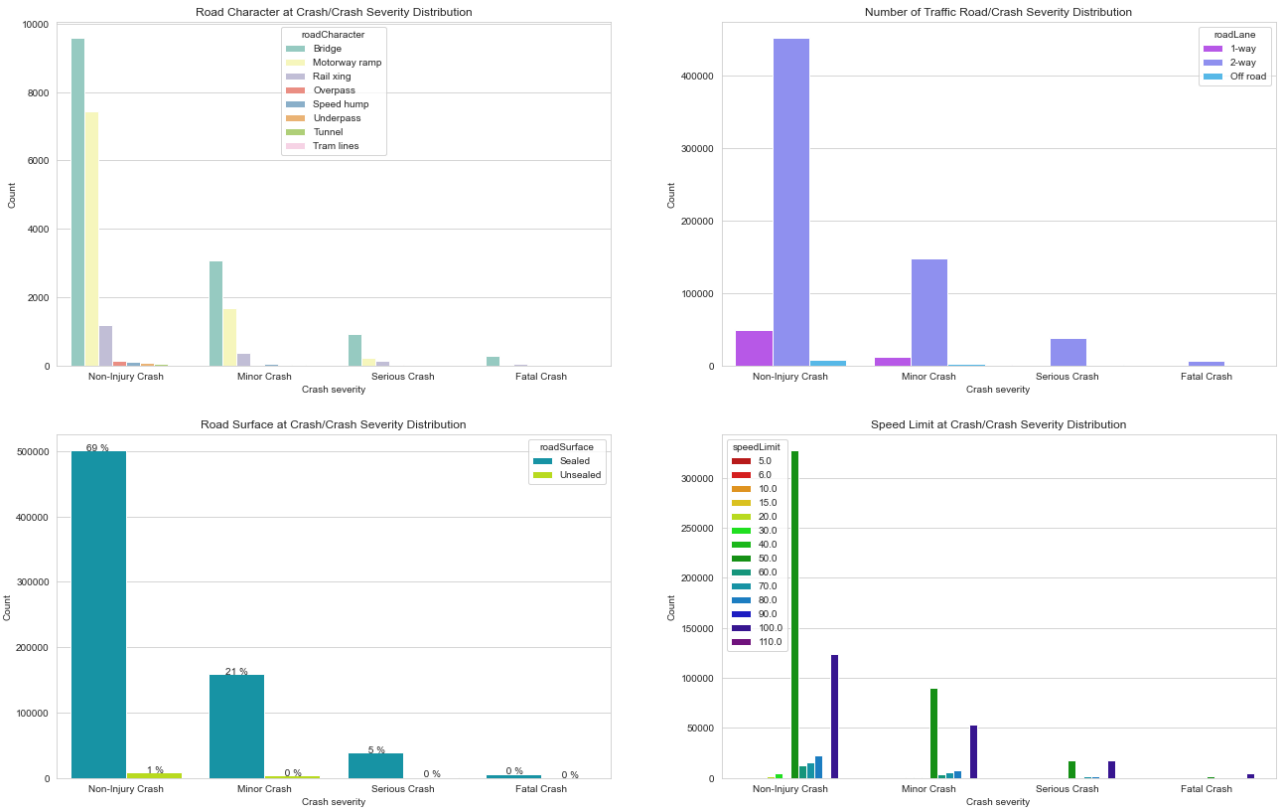

Figure B.3: Plots showing relationship between 'roadCharacter' , 'roadLane', 'roadSurface' and 'speedLimit' with the target feature 'crashSeverity'

These plots follow the same pattern as the others and they do not show any significant changes in the proportionality of the samples for each class.

Appendix C

Phi_K significance matrix for the categorical features

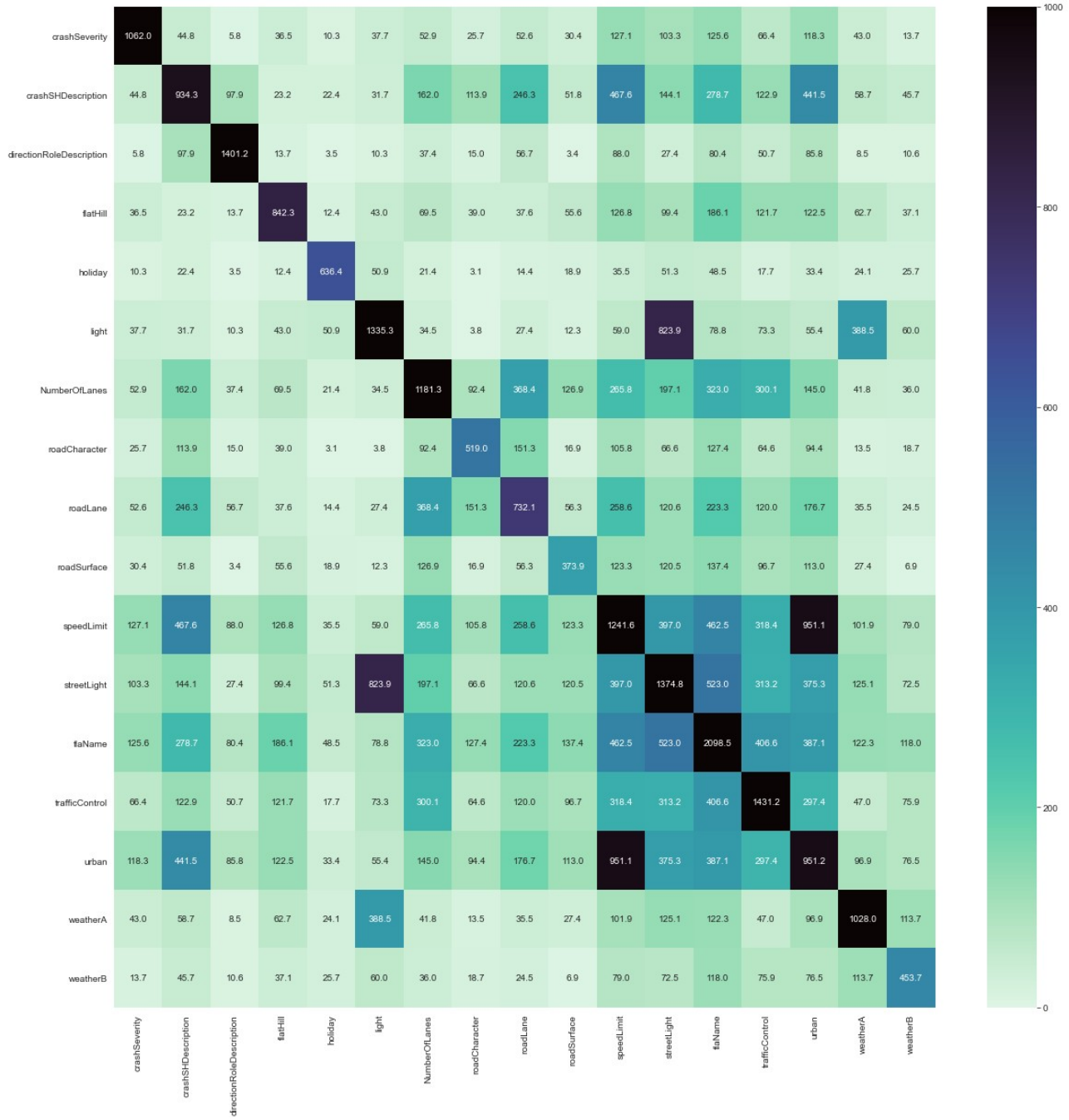

Figure C.1: Phi_K sensitivity matrix