milestone4

Frank Guo

2024-05-21

Milestone 4 - Crashes Analysis

By Xiaodong Guo

1. Goal.

Our project has a significant objective- to construct models identifying the pivotal factors contributing to severe crashes. This crucial task is based on the data provided by the New Zealand Transport Agency(NZTA).

2. Data Source.

The original data, meticulously collected, originated from the Waka Kotahi NZ Transport Agency's open data portal(the tutor provided the link in the assignment piece). We specifically downloaded the dataset named "Crash Analysis System (CAS) data" from the "Crash" catalogue, which encompasses all traffic crashes reported to us by the NZ Police. The data format is a "CVS" file. It was created on 3/25/2020 and last updated on 3/14/2024.

The data includes crash datas from 2000 to 2023.

3. Data Processing.

Load Data. We load the data from csv flie. The dataset we got have 72 columns,and 821744 rows. All the descriptions of attributes will be listed in ***Appendix 4***.

```
## [1] 821744     72
```

1.drop columns not related to our object. ***We*** select following columns by common sense of mine.There are also have some description columns, that is long string values to desript an event or street name. That looks no sense,should drop them too.

Like "crashLocation1","crashLocation2", the location of crash is too detailed, we will keep region instead.

Also, the column like "minorInjuryCount","seriousInjuryCount","fatalCount", they are highly related to define if the crash is severe. It is not superised that you will get more than 99% accuracy in prediction if these features are included. We will remove these columns too.

Compared crashYear with crashFinancialYear, the crashFinancialYear is more related to the domain business. So the crashFinalcialYear is kept.

```
columns_to_drop <- c("X","Y","OBJECTID","areaUnitID","crashDirectionDescription","","crashDistance",
                     "tlaId","tlaName","debris","meshblockId","northing","easting","crashLocation1",
                     "crashLocation2","directionRoleDescription","crashSHDescription","otherObject",
                     "phoneBoxEtc","minorInjuryCount","seriousInjuryCount","fatalCount","crashYear",
                     "objectThrownOrDropped")

data <- select(data, -one_of(columns_to_drop))
```

2. dropping columns that all values are almost Null(more then 99% of the data is null). Columns like these are too sparse. The column names are crashRoadSideRoad" and "intersection".

```
na_percentage <- colMeans(is.na(data))
columns_with_high_na <- names(na_percentage[na_percentage > 0.99])
print(columns_with_high_na)
```

```
## [1] "crashRoadSideRoad" "intersection"
```

```
data <- data %>% select(-columns_with_high_na)
```

3. Define the target lable. Define crashSeverity == "Fatal Crash" and crashSeverity == "Serious Crash" as severe crashes given numeric value 1,Define crashSeverity == "Minor Crash" | crashSeverity == "Non-Injury Crash" as not severe crashes given numeric value 0.

The ***"crashSeverity"*** Label will be the target label.

```
table(data$crashSeverity)
```

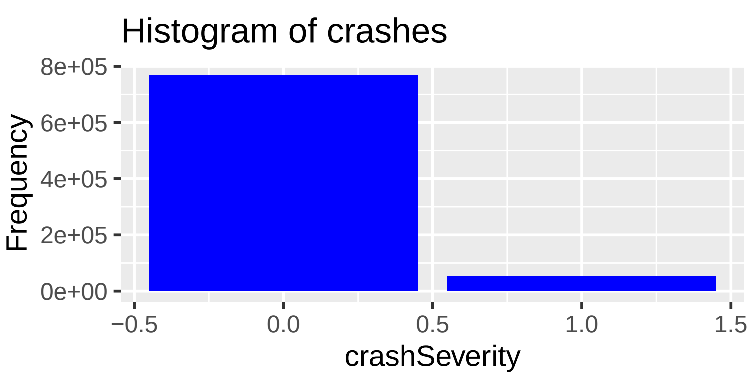
```
##
##      Fatal Crash      Minor Crash Non-Injury Crash    Serious Crash
##             7589           191336           575954            46865
```

After mapping:

```
data <- data %>%
  mutate(crashSeverity = ifelse(crashSeverity == "Fatal Crash" | crashSeverity == "Serious Crash", 1,
                    ifelse(crashSeverity == "Minor Crash" | crashSeverity == "Non-Injury Crash", 0,
                           2))) %>%  filter(crashSeverity != 2)
table(data$crashSeverity)
```

```
##
##      0      1
## 767290  54454
```

There are 767290 regular crashes and 54454 severe crashes. Our target is to find the cause of severe crashes,but the severe observations' size is really small compares to the regular crashes. So we can't use the whole dataset directly, it will overfit the majority(regular crashes) and underfit the minority(severe crashes), cause bias to majority,loss the importance information for our target(server crashes) inference.



4.Dealing with other predictors. ***1. Attributes "weatherA" and "weatherB".***

They are String values could be treated as factors,not too many factors in each attribute, and the combinations also not too many factors but are more sensitive to understand the whole weather situation. In my opinion, these two could be combined as one attribute "weather",much easier to display and dealing with it later.

Also For these two attributes, the Na value or String "None" are replaced as "Others" condition.What we got is like the output.

```
data$weatherA <- ifelse(data$weatherA %in% c("None", "Null"), "Others", data$weatherA)
data$weatherB<- ifelse(data$weatherB %in% c("None", "Null"), "", data$weatherB)

data <- data %>% unite(weatherA,weatherB,col=weather,sep=" ")
table(data$weather)
```

```
##
##                   Fine                   Fine Frost           Fine Strong wind
##                 621264                         7140                       7217
##           Hail or Sleet          Hail or Sleet Frost Hail or Sleet Strong wind
##                     88                           21                         23
##             Heavy rain             Heavy rain Frost     Heavy rain Strong wind
##                  29836                           43                       3274
##             Light rain             Light rain Frost     Light rain Strong wind
##                 120517                          333                       3360
##            Mist or Fog            Mist or Fog Frost    Mist or Fog Strong wind
##                  10277                          894                        135
##                 Others                 Others Frost         Others Strong wind
##                  15137                          438                        203
##                   Snow                   Snow Frost           Snow Strong wind
##                    982                          385                        177
```

2. ***Deal with "region" and "holiday"***.

Replace the "Null","None" value in region with "Others".

List all columns have the value "". That is holiday and regions.Replace the"" in other character attributes with "Others". The result will be used in Data Analysis block, so omitting the talbe() here.

```
empty_columns <- colnames(data)[apply(data == "", 2, any)]

# Print the empty columns
print(empty_columns)
```

```
##  [1] NA        NA        NA        NA        NA        NA        NA
##  [8] NA        NA        "holiday" NA        NA        NA        NA
## [15] NA        NA        NA        NA        NA        NA        "region"
## [22] NA        NA        NA        NA        NA        NA        NA
## [29] NA        NA        NA        NA        NA        NA        NA
## [36] NA        NA        NA
```

```
data[data == ""] <- "Others"
```

3. ***All the attributes with Na value***.

List all the other attributes with Na value,then check these attributs.

According the descriptions of these attributes, we can use 0 to fill na value.Using 2 examples to explaint why 0 be used:

For "advisorySpeed" or "temporarySpeedLimit" attribute, the value is mean special speed limitation applied or advised in the road which is involed in the crash. use 0 here means no special speed limit applied(according the rode code, that is open road.follows open road speed limit).

For other attributes in the list, the value indicates the number of items involved in the crash. the Na value means no item(named by attribute name) is involved,that equals to 0.

```r
na_columns <- sapply(data, function(x) any(is.na(x)))
columns_with_na <- names(data)[na_columns]
print(columns_with_na)
```

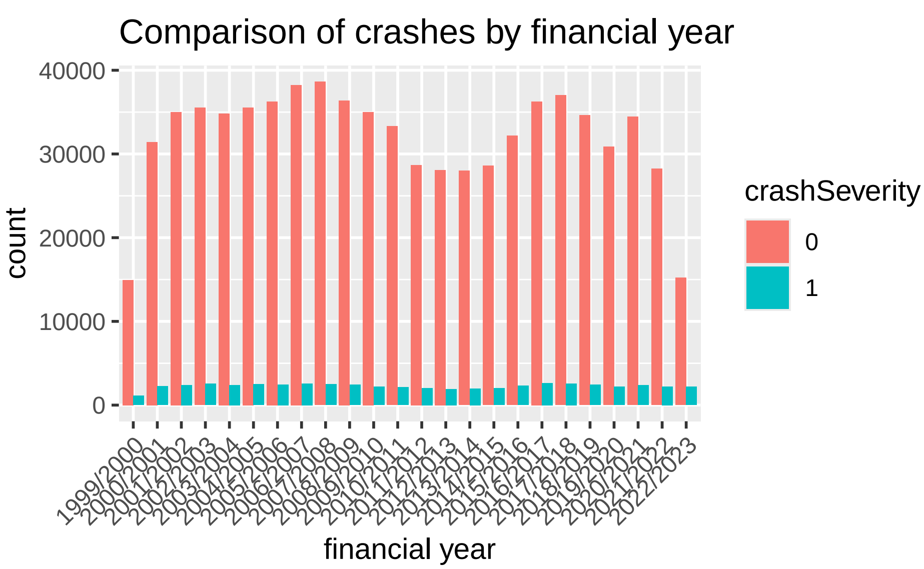
```
##  [1] "advisorySpeed"       "bicycle"             "bridge"
##  [4] "bus"                 "carStationWagon"     "cliffBank"
##  [7] "ditch"               "fence"               "guardRail"
## [10] "houseOrBuilding"     "kerb"                "moped"
## [13] "motorcycle"          "NumberOfLanes"       "otherVehicleType"
## [16] "overBank"            "parkedVehicle"       "pedestrian"
## [19] "postOrPole"          "roadworks"           "schoolBus"
## [22] "slipOrFlood"         "speedLimit"          "strayAnimal"
## [25] "suv"                 "taxi"                "temporarySpeedLimit"
## [28] "trafficIsland"       "trafficSign"         "train"
## [31] "tree"                "truck"               "unknownVehicleType"
## [34] "vanOrUtility"        "vehicle"             "waterRiver"
```

```r
data <- data %>%
  mutate_at(vars(one_of(columns_with_na)), ~replace_na(., 0))
```

Finally, all the Na or missing data is imputated and remedied.

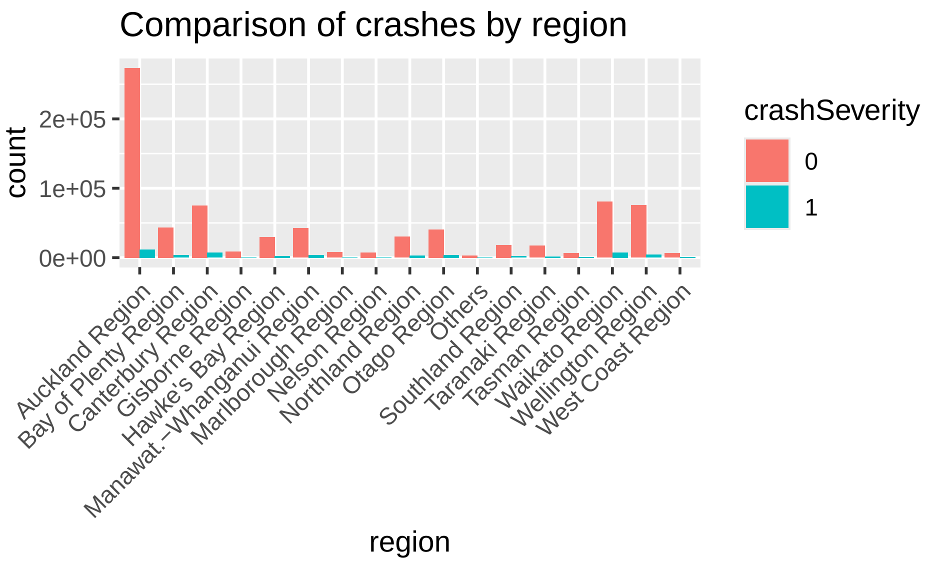
exploint data:

1. at the financial year vs crashes, it is fluctuated, indicated some relations with the year,confirmed by Chi-test. By intuition and guess, maybe budget for traffic bureau matters.

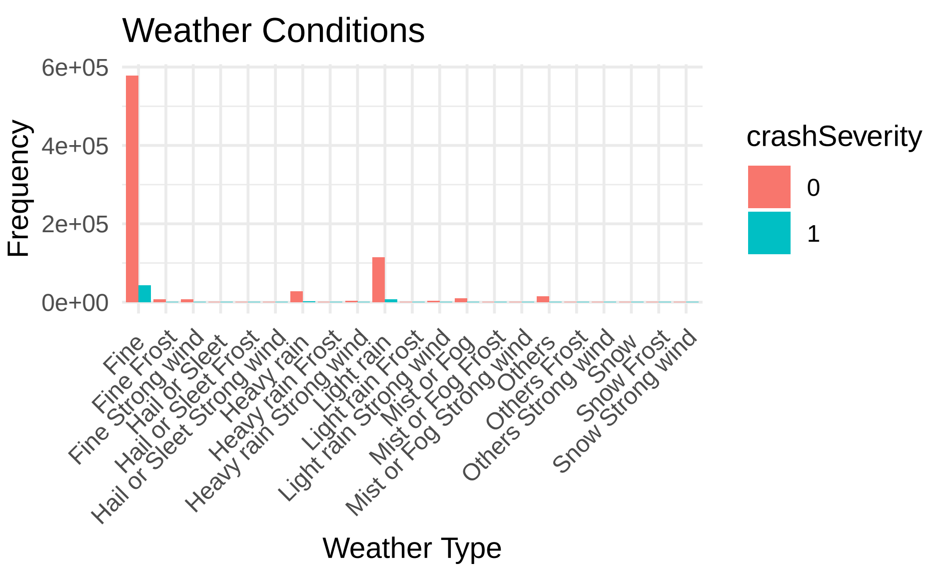



```
##
##  Pearson's Chi-squared test
##
## data:  table(data$crashFinancialYear, data$crashSeverity)
## X-squared = 1143.2, df = 23, p-value < 2.2e-16
```

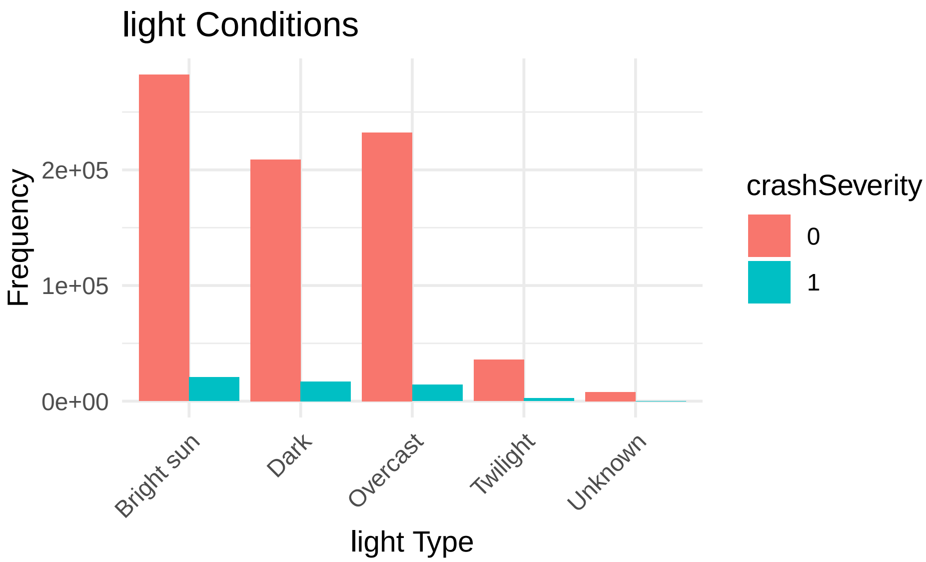
2. ***Comparing*** the crashes by region, Auckland region looks obviously high than others, considering of population density, it looks reasonable.While the ratio of severe crashes looks low. While regions like Gisbone,northland, southland, hawkesbay and westcoast looks has much hight ratio of severe crashes.



3. ***Most*** of crashes happen in fine weather,and also, most of predictor attributes show strong skew. If we skip the fine weather stuation. It is clear that the light rain weather looks notable too.

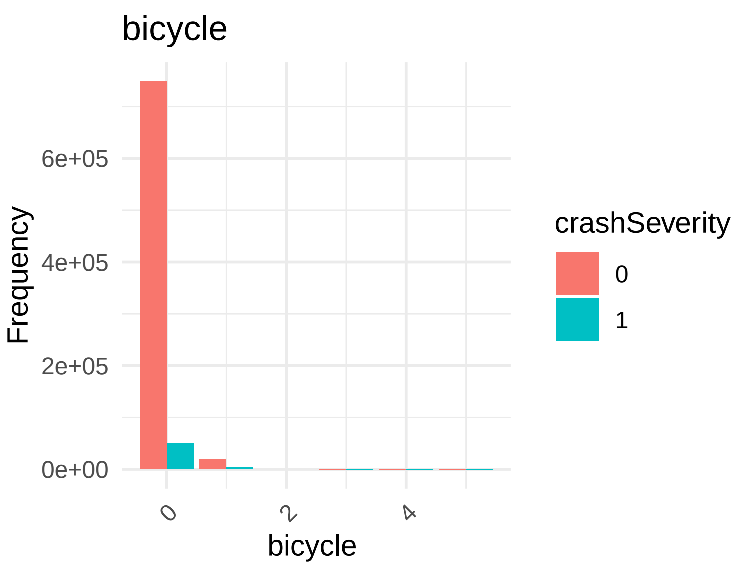


4. ***Crashes*** happened all the light situation,the number of severe crashes looks almost the same in sunny ,dark or overcast.While in dark or twilight,the severe crashes ratio looks higher.

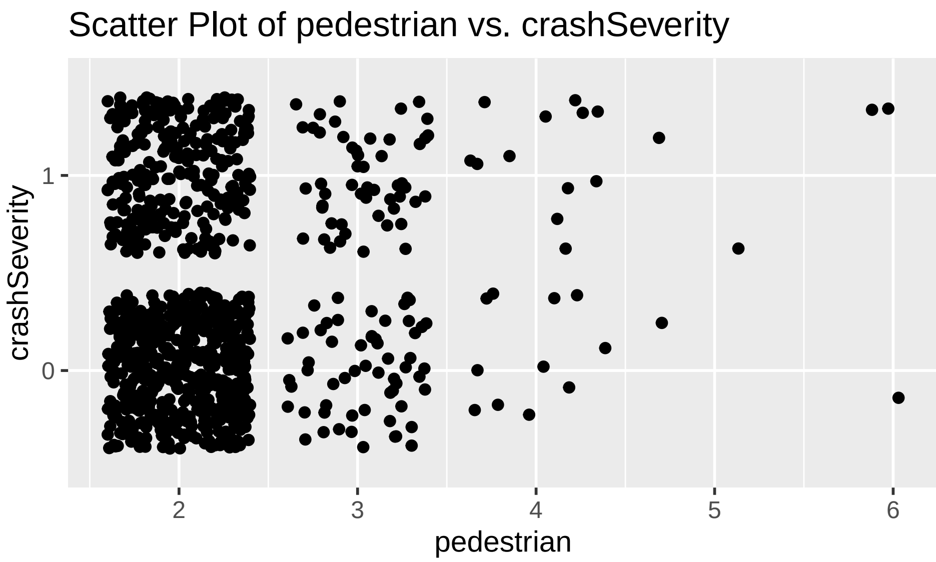



```
##
##  Pearson's Chi-squared test
##
## data:  table(data$light, data$crashSeverity)
## X-squared = 1078.4, df = 4, p-value < 2.2e-16
```

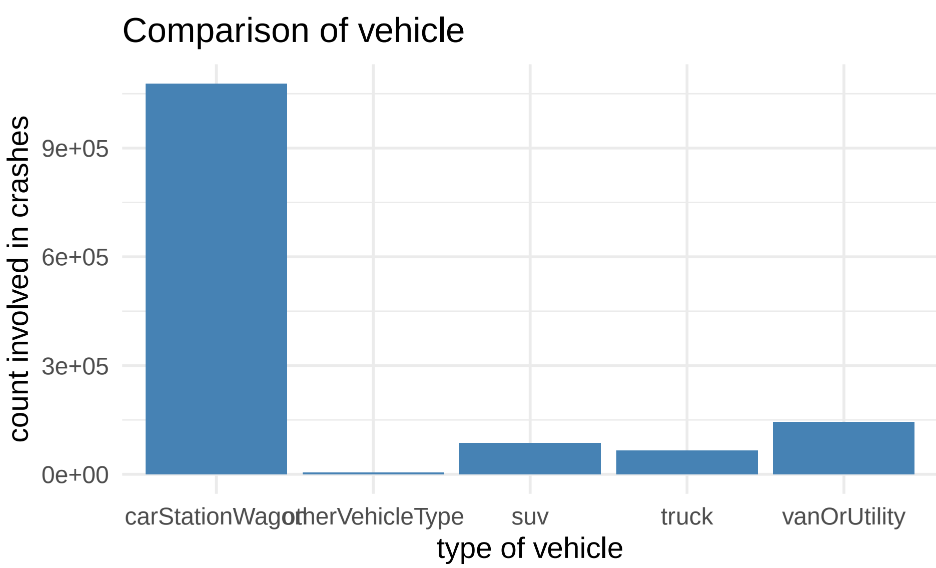
5. ***From*** the plot, we can tell that more bicycles involed in craches, more likly the crash to be a severe crash.Also, I skiped the bicycle = 1 and 0 because of the high proportion of the data at those values.



6. ***As*** same as bicycles, from the Scatter plot, more pedestrians involved in the crash, the crash is more likely to be a several crash.I also skip the pedenstrain<2 in the plot.



7. ***From*** the bar plot, we can see that the carStationWagon type of car is the most related car type in crashes. By chi-test, it shows strong relations with crashes.



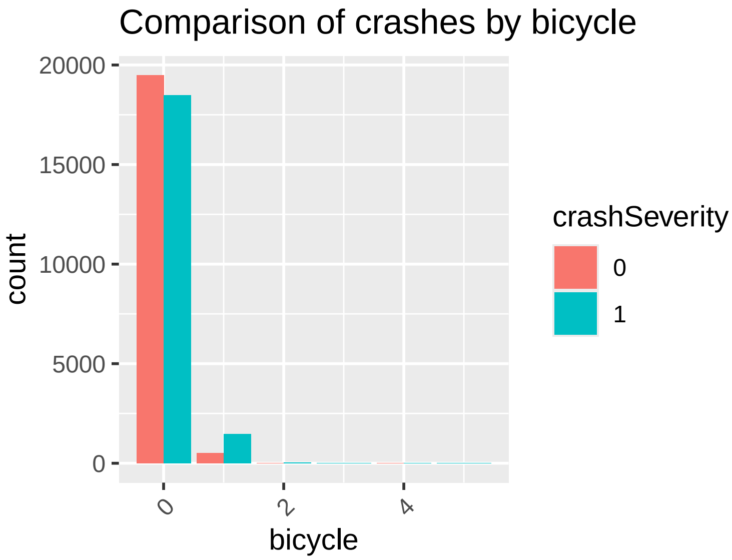

```
##
##  Pearson's Chi-squared test
##
## data:  table(data$carStationWagon, data$crashSeverity)
## X-squared = 19647, df = 11, p-value < 2.2e-16
```

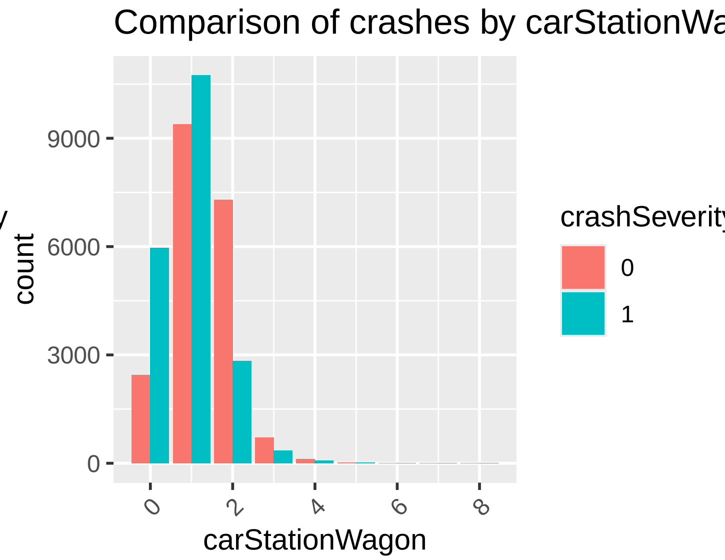
5. Analytical Plan.

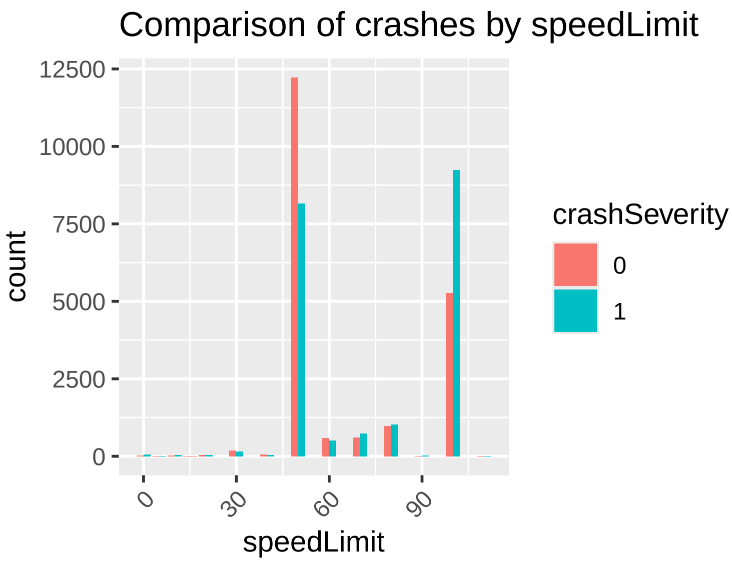
Sampling Strategy.

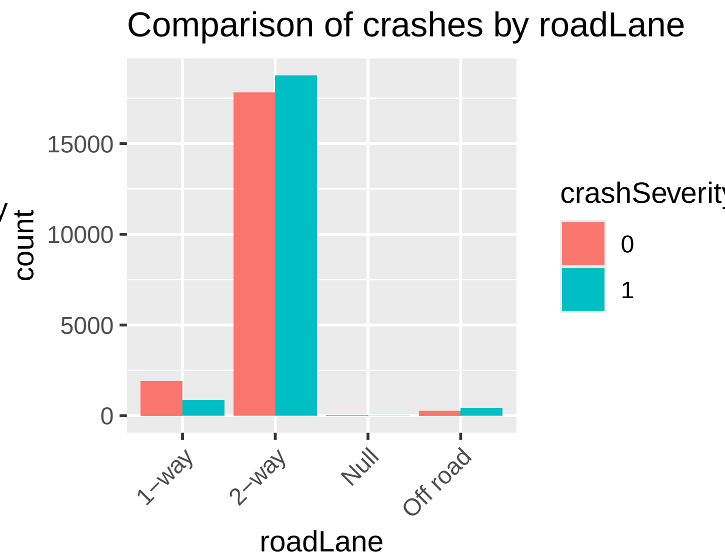
Because of the imbalance of the dataset, I decisied to equally sample from severe crashes and regular crashes,that is ***20000*** observations from each type of crashes. ***We*** will use 80% of the sample data as training data, and 20% of the sample data as test data. To fit the model, we will factorise the charactor attributes,and numberic them.

explore the sample data. *I* use bicycle,carStationWagon as samples to explore the distribution of the predictors. As we can see the most of same bicycle attribute value , having different target lable. It makes the inference be different, and indicates low accuracy of prediction. While carStationWagon shows some good trend to distinct the crashes compared to bicycle.









Fitting Strategy

I fit the data using logistic regression, random forest and xgboost, to compare the performance using F1 score , accuracy ,TPR. Because we more focus on the factors of severe crashes,I applied 2 times heavier penality for those are severe crashes,but misclassified to archive higher TRP. List the 15s important factors from the top as result for every fit.

1. Fit logistic regression model. class_weights <- ifelse(training_data$crashSeverity == 1, 2, 1) # Penalize severe crashes class more heavily.

lr_model <- glm(crashSeverity ~ ., data = training_data, weights=class_weights, family = "binomial").

Output is the confusion matrix, f1 score, accuracy, and write the importance list to importance_glm.csv.It is very similiar with the list by P values:

```
##           Reference
## Prediction    0    1
##          0 2256  568
##          1 1740 3436
```

```
##        F1
## 0.6615836
```

```
## Accuracy
##   0.7115
```

2. fit decision tree model with random forest ensemble with permutation

Permutation importance provide a more accurate estimate of variable importance, especially in situations where the relationship between predictors and the response is nonlinear or non-monotonic.

using

class_weights <- ifelse(training_data$crashSeverity == 1, 2, 1) # Penalize 'setosa' class more heavily.

cv_rf <- ranger(crashSeverity ~ ., data = training_data, num.trees = 500,
mtry = 6, min.node.size = 3, case.weights = class_weights, importance = "permutation", sample.fraction = 0.8, num.fold = 5, verbose = FALSE).

Ouput the confusion matrix,F1 score and accuracy of the prediction. Then write the importance list to file importance_rf.csv.

```
##           Reference
## Prediction    0    1
##          0 2300  523
##          1 1696 3481
```

```
##        F1
## 0.6745857
```

```
## Accuracy
## 0.722625
```

3. Using xgboost ensemble with logistic regression to predict.

Tried to use CV to find the best hyper-paramters of xgboost, but need too long time to run in my computer.So interupted and just use the parameters like the following.

class_weights <- ifelse(training_data$crashSeverity == 1,2, 1) # Penalize 'setosa' class more heavily.

positive_weight <- sum(class_weights[training_Y == 1]) / sum(class_weights[training_Y == 0]).

xgb_model <- xgboost(data = as.matrix(training_X), label = training_Y, max_depth = 3, eta = 0.1, nrounds = 300, scale_pos_weight = positive_weight, # Set scale_pos_weight objective = "binary:logistic", verbose = FALSE).

Ouput the confusion matrix,F1 score and accuracy of the prediction. Then write the importance list to importance_xgb.csv.

```
##           Reference
## Prediction    0    1
##          0 2258  499
##          1 1738 3505
```

```
##        F1
## 0.6687398
```

```
## Accuracy
## 0.720375
```

4. Also tried others like neural network to predict.Required more computaional resources but the result is no better then random forest. And the importance of the factors is not convient to get during the process of modeling.So abandoned.

Result.

compared with three models, the random forest got the F1 score 0.6804665 accuracy 0.726. While logistic regression got F1 score 0.6739194 Accuracy 0.719,and Xgboost got F1 0.6777761 and 0.726875. They got different importance lists. While random forest and xgboost is better but similar performance. I'd like to merge the importance list gotten from these two model to get a merged importance list. If a feature is deemed important by both models, it's likely that the feature is truly important. This can make your interpretation more robust. The code will be in ***Appendix 3***.

The top 15 key factors will be:

carStationWagon. ***speedLimit***. ***motorcycle***. ***pedestrian***. ***bicycle***. ***region***. ***roadLane***. ***crashFinancialYear***. ***streetLight***. ***tree***. ***vanOrUtility***. ***fence***. ***weather***. ***NumberOfLanes***. ***postOrPole***.

Btw, the full sorted list of every model will be in the ***appendix 1***.

Discussion.

1. ***There*** are 767290 regular crashes and 54454 severe crashes. Our target is to find the cause of severe crashes,but the severe observations' size is really small compares to the regular crashes. I decisied to equally sample from severe crashes and regular crashes,that is ***20000*** observations from each type of crashes. To find the key factors of severe crashes, I put 2 times heaver penality for misclassfying the server crashes.
2. The Data provided is not only imbalance in the target class,but also very unbalance in predictors(as you can see in the plot of bicycles).
3. To analysis the sampled data,as we can see the with the simlilar attribute value, having different target lable. It makes the inference be different, and indicates low accuracy of prediction.
4. I fitted the model with different method(logistic, random forest and xgboost). It turned out that the random forest and xgboost had similar performance, the xgboost given higher TPR.
5. Permutation importance provide a more accurate estimate of variable importance, especially in situations where the relationship between predictors and the response is nonlinear or non-monotonic.So that is used to build Random Forest here.
6. Tried to find best hyper parameters for random forest and xgboost, but very time consuming, and the final outcomes no outstanding improvement compared to the current variable. Some related code still keep in the source code,but ommited.
7. Tried more heavier penality, but the accuracy((TP+TN)/TOTAL) lower than 70%, using 2 times heavier finnally.
8. Random forest and xgboost have similar performance,but got different importance lists. Combine two importance after normalizing provide more robust interpration for the feature is deemed important by both models(Appendix 3).
9. All the code(used finnally) related to the three models are in the Appendix 2.

Appendix.

Appendix 1. The importance lists from all the models and combined:

Logistic regression	Random Forest permutation	xgboost	combined
pedestrian	carStationWagon	carStationWagon	carStationWagon
motorcycle	speedLimit	speedLimit	speedLimit
bicycle	motorcycle	pedestrian	motorcycle
tree	pedestrian	motorcycle	pedestrian
postOrPole	urban	bicycle	bicycle
moped	vanOrUtility	crashFinancialYear	region
roadLane	bicycle	region	roadLane
speedLimit	fence	roadLane	crashFinancialYear
truck	streetLight	weather	streetLight
vanOrUtility	tree	streetLight	tree
fence	truck	tree	vanOrUtility
cliffBank	region	light	fence
suv	roadLane	NumberOfLanes	weather
ditch	suv	postOrPole	NumberOfLanes
bus	postOrPole	moped	postOrPole
carStationWagon	crashFinancialYear	trafficControl	truck
bridge	NumberOfLanes	fence	light
otherVehicleType	weather	advisorySpeed	suv
weather	cliffBank	vanOrUtility	trafficControl
NumberOfLanes	advisorySpeed	truck	moped
overBank	trafficControl	guardRail	advisorySpeed
houseOrBuilding	light	parkedVehicle	cliffBank
waterRiver	flatHill	flatHill	flatHill
trafficIsland	parkedVehicle	suv	parkedVehicle
guardRail	moped	holiday	guardRail
parkedVehicle	ditch	roadSurface	trafficSign
advisorySpeed	trafficSign	cliffBank	roadSurface
region	guardRail	overBank	ditch
trafficControl	bus	houseOrBuilding	overBank
train	overBank	roadCharacter	holiday
strayAnimal	roadSurface	trafficSign	roadCharacter
urban	roadCharacter	waterRiver	houseOrBuilding
kerb	trafficIsland	temporarySpeedLimit	bus
roadCharacter	kerb	ditch	temporarySpeedLimit
slipOrFlood	houseOrBuilding	bus	trafficIsland
trafficSign	temporarySpeedLimit	otherVehicleType	waterRiver
flatHill	holiday	bridge	kerb
temporarySpeedLimit	strayAnimal	strayAnimal	strayAnimal
streetLight	waterRiver	trafficIsland	bridge
schoolBus	bridge	kerb	otherVehicleType
roadworks	slipOrFlood	train	slipOrFlood
light	unknownVehicleType	slipOrFlood	train
roadSurface	otherVehicleType	taxi	roadworks
holiday	train	vehicle	vehicle
vehicle	roadworks	roadworks	schoolBus
unknownVehicleType	schoolBus	schoolBus	taxi
crashFinancialYear	vehicle	unknownVehicleType	unknownVehicleType
taxi	taxi	NA	NA

Appendix 2. Codes of models:

1. For logistic regression:

```
library(caret)


test_Y <- test_data$crashSeverity

#test_X <- test_data %>% select(-crashSeverity)

test_lr <- test_data
class_weights <- ifelse(training_data$crashSeverity == 1, 2, 1)  # Penalize severe crashes class more h

lr_model <- glm(crashSeverity ~ ., data = training_data, weights=class_weights, family = "binomial")

#print(lr_model)
predictions <- predict(lr_model, newdata = test_lr, type = "response")
binary_predictions <- factor(ifelse(predictions >= 0.5, 1, 0), levels = levels(test_Y))
# accuracy <- mean(binary_predictions == test_Y)
test_Y <- factor(test_Y)

confusion_matrix <- confusionMatrix(binary_predictions, test_Y)
accuracy <- accuracy <- confusion_matrix$overall["Accuracy"]
print(confusion_matrix$table)
#print(confusion_matrix)

f_score <- confusion_matrix$byClass["F1"]

print(f_score)

#knitr::kable(table(binary_predictions,test_Y))

print(accuracy)

# # Extract coefficients
# coefficients <- coef(lr_model)
#
# # Calculate absolute values of coefficients
# abs_coefficients <- abs(coefficients)
#
# # Create a dataframe to store coefficients and their absolute values
# coef_df <- data.frame(predictor = names(coefficients), coefficient = coefficients, abs_coefficient = 
#
# # Sort coefficients based on absolute values
# sorted_coef_df <- coef_df[order(abs_coefficients, decreasing = TRUE), ]
#
# # Print sorted coefficients
# knitr::kable(head(sorted_coef_df,15))

importance <- varImp(lr_model, scale = FALSE)

variable_names <- rownames(importance)
#print(variable_names)
```

```
importance_df <- data.frame(importance)
#glimpse(importance_df)
importance_scores <- importance[, 1]

# Create a data frame with variable names and importance scores
importance_df <- data.frame(
  Variable = variable_names,
  Importance = importance_scores
)

# Convert Importance column to numeric
importance_df$Importance <- as.numeric(as.character(importance_df$Importance))

# Sort the data frame by Importance column in descending order
importance_df_sorted <- importance_df[order(importance_df$Importance, decreasing = TRUE), ]

# Print the sorted importance scores
importance_df_sorted <- as.data.frame(importance_df_sorted)
write.csv(importance_df_sorted, "importance_glm.csv", row.names = FALSE)

#knitr::kable(head(importance_df_sorted,15))
```

2. ***For Random Forest:***

```
library(ranger)
# Define class weights
class_weights <- ifelse(training_data$crashSeverity == 1, 2, 1)  # Penalize 'setosa' class more heavily

# Cross-validation with ranger
cv_rf <- ranger(crashSeverity ~ ., data = training_data, num.trees = 500,
                   mtry = 6,
                  min.node.size = 3,
                  case.weights = class_weights,
#                  importance = "impurity",
                importance = "permutation",
                 sample.fraction = 0.8,
                  num.fold = 5,  # Number of folds for cross-validation
              verbose = FALSE  # Print progress
        )

# Get cross-validation results
#cv_results <- cv_rf$prediction.error

#print(cv_results)
# Find the fold with the lowest prediction error
#best_fold <- which.min(cv_results)

# Get the corresponding model
#best_model <- cv_rf

# Print information about the best model
#print(best_model)

predictions <- predict(cv_rf, data = test_data)
```

```r
predicted_values <- predictions$predictions

# accuracy <- mean(binary_predictions == test_Y)
test_Y <- factor(test_Y)

confusion_matrix <- confusionMatrix(predicted_values, test_Y)
accuracy <- confusion_matrix$overall["Accuracy"]
print(confusion_matrix$table)
#print(confusion_matrix)

f_score <- confusion_matrix$byClass["F1"]

print(f_score)

#knitr::kable(table(binary_predictions,test_Y))

print(accuracy)

importance_measures <- importance(cv_rf)

# Convert the named numeric vector to a dataframe
importance_df <- data.frame(
  Feature = names(importance_measures),
  Importance = as.numeric(importance_measures)
)


#importance_df <-data.frame(name = names(importance_measures),value = unlist(importance_measures))
#importance_df <-data.frame(importance_measures)

#glimpse(importance_df)
sorted_importance_df <- importance_df[order(importance_df$Importance,decreasing = TRUE),,drop = FALSE]

write.csv(sorted_importance_df, "importance_rf.csv", row.names = FALSE)
# knitr::kable(head(sorted_importance_df,15))
```

***3. For Xgboost:

```r
library(xgboost)

library(caret)

test_Y <- test_data$crashSeverity
test_Y <-  as.numeric(test_Y)-1
test_X <- test_data %>% select(-crashSeverity)

training_X <- training_data %>% select(-crashSeverity)

training_Y <- training_data$crashSeverity

#X <- as.matrix(training_X)  # Features
training_Y <- as.numeric(training_Y)-1
#table(training_Y)
```

```r
class_weights <- ifelse(training_data$crashSeverity == 1,2, 1)  # Penalize 'setosa' class more heavily
positive_weight <- sum(class_weights[training_Y == 1]) / sum(class_weights[training_Y == 0])

# Train the XGBoost model
xgb_model <- xgboost(data = as.matrix(training_X), label = training_Y,
                     max_depth = 4, eta = 0.1, nrounds = 300,
                     scale_pos_weight = positive_weight,  # Set scale_pos_weight
                     objective = "binary:logistic",
                     verbose = FALSE)

# Predict probabilities
predictions <- predict(xgb_model, as.matrix(test_X))  # Probability of positive class

# Convert predictions to factor with levels "0" and "1"
predictions <- factor(ifelse(predictions >= 0.5, 1, 0))

# Convert test_Y to factor with levels "0" and "1"
test_Y <- factor(test_Y)

# Compute confusion matrix
conf_matrix <- confusionMatrix(predictions, test_Y)
print(conf_matrix$table)

accuracy <- conf_matrix$overall["Accuracy"]

#print(confusion_matrix)

f_score <- conf_matrix$byClass["F1"]

print(f_score)

#knitr::kable(table(binary_predictions,test_Y))

print(accuracy)
importance_scores <- xgb.importance(model = xgb_model)

importance_scores <- as.data.frame(importance_scores)
write.csv(importance_scores, "importance_xgb.csv", row.names = FALSE)

# Print the importance scores

#knitr::kable(head(importance_scores,15))

# Plot feature importance
```

Appendix 3. Code to Combine the importance list:

```r
# Load the data
importance_rf <- read.csv("importance_rf.csv")
importance_xgb <- read.csv("importance_xgb.csv")

# Normalize the importance
```

```r
importance_rf$Importance <- importance_rf$Importance / sum(importance_rf$Importance)
importance_xgb$Gain <- importance_xgb$Gain / sum(importance_xgb$Gain)

# Combine the importance from both models
combined_importance <- merge(importance_rf, importance_xgb, by = "Feature")
combined_importance$Combined <- combined_importance$Importance + combined_importance$Gain

# Sort the features based on this combined importance
combined_importance <- combined_importance[order(-combined_importance$Combined), ]

# Write the dataframe to a CSV file
write.csv(combined_importance, file = "merged_importance.csv", row.names = FALSE)
```

Appendix 4. All The Attributes description:

Attribute Name	Alias Name	Description
advisorySpeed	Advisory Speed	The advisory (adv) speed (spd) at the crash site at the time of the crash.
areaUnitID	Area Unit ID	The unique identifier of an area unit.
bicycle	Bicycle	Derived variable to indicate how many bicycles were involved in the crash.
bridge	Bridge	Derived variable to indicate how many times a bridge, tunnel, the abutments, handrails were struck in the crash.
bus	Bus	Derived variable to indicate how many buses were involved in the crash (excluding school buses which are counted in the SCHOOL_BUS field).
carStationWagon	Car/Station Wagon	Derived variable to indicate how many cars or station wagons were involved in the crash.
cliffBank	Cliff or Bank	Derived variable to indicate how many times a ‘cliff’ or ‘bank’ was struck in the crash. This includes retaining walls
crashDirectionDescription	Crash Direction Description	The direction (dirn) of the crash from the reference point. Values possible are ‘North’, ‘East’, ‘South’ or ‘West’.
crashDistance	Crash Distance	The distance (dist) of the crash from the reference point for the crash. The reference point is often the intersection of ‘crash road’ and ‘side road’ (refer to ‘cr_rd_sd_rd’ variable).
crashFinancialYear	Crash Financial Year	The financial (fin) year in which a crash occurred, if known. This is displayed as a string field. eg 2004/2005
crashLocation1	Crash Location 1	Part 1 of the ‘crash location’ (crash_locn). May be a road name, route position (RP), landmark, or other, e.g. ‘Ninety Mile Beach’. Used for location descriptions in reports etc.

Attribute Name	Alias Name	Description
crashLocation2	Crash Location 2	Part 2 of the ‘crash location’ (crash_locn). May be a side road name, landmark etc. Used for location descriptions in reports etc.
crashSeverity	Crash Severity	The severity of a crash. Possible values are ‘F’ (fatal), ‘S’ (serious), ‘M’ (minor), ‘N’ (non-injury). This is determined by the worst injury sustained in the crash at time of entry.
crashSHDescription	Crash SH Description	Indicates where a crash is reported to have occurred on a State Highway (SH) marked ‘1’, or on another road type marked ‘2’.
crashYear	Crash Year	The year in which a crash occurred, if known.
debris	Debris	Derived variable to indicate how many times debris, boulders or items dropped or thrown from a vehicle(s) were struck in the crash
directionRoleDescription	Direction Role Description	The direction (dirn) of the principal vehicle involved in the crash. Possible values are North, South, East or West.
ditch	Ditch	Derived variable to indicate how many times a ‘ditch’ or ‘waterable drainage channel’ was struck in a crash.
easting	Easting	The easting coordinate of an object (usually a crash) expressed in NZMG referred to the WGS84 datum to a precision of 1m. Please note, in some instances crashes are not able to be assigned to GPS co-ordinates. These crashes have been assigned eastings and northings of ‘0,0’ in this dataset. There are two main reasons that a GPS coordinate cannot be allocated to a crash. Firstly, that the crash has been reported but the location was unknown. Secondly in a small number of instances, a crash may have occurred on a road which is not yet captured on the CAS spatial layer.
fatalCount	Fatal Count	A count of the number of fatal casualties associated with this crash.
fence	Fence	Derived variable 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	Flat Hill	Whether the road is flat or sloped. Possible values include ‘Flat or ’Hill’.
guardRail	Guard Rail	Derived variable 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	Holiday	Indicates where a crash occurred during a ‘Christmas/New Year’, ‘Easter’, ‘Queens Birthday’ or ‘Labour Weekend’ holiday period, otherwise ‘None’.
houseOrBuilding	House or Building	Derived variable to indicate how many times a houses, garages, sheds or other buildings(Bldg) were struck in the crash
intersectionMidblock	Intersection Midblock	Derived variable to indicate if a crash occured at an intersection (intsn) or not. The ‘intsn_midblock’ variable is calculated using the ‘intersection’ and ‘junction_type’ variables. Values are ‘Intersection’ (where intersection variable = ‘Intersection’ or {‘Intersection’ = ‘At Landmark’ and junction_type is not in (‘Unknown’ or ‘Driveway’)} OR {Intersection = ‘Unknown’ and crash_dist <= 10}), otherwise ‘Midblock’ for crashes not meeting the criteria for ‘Intersection’).

Attribute Name	Alias Name	Description
kerb	Kerb	Derived variable to indicate how many times a kerb was struck in the crash, that contributed directly to the crash.
light	Light	The light at the time and place of the crash. Possible values: ‘Bright Sun’, ‘Overcast’, ‘Twilight, ’Dark’ or ’ Unknown’.
meshblockId	Meshblock ID	The unique identifier of a meshblock.
minorInjuryCount	Minor Injury Count	A count of the number of minor injuries (inj) associated with this crash.
moped	Moped	Derived variable to indicate how many mopeds were involved in the crash.
motorcycle	Motorcycle	Derived variable to indicate how many motorcycles were involved in the crash.
northing	Northing	The northing coordinate of an object (usually a crash) expressed in NZMG referred to the WGS84 datum to a precision of 1m. Please note, in some instances crashes are not able to be assigned to GPS co-ordinates. These crashes have been assigned eastings and northings of ‘0,0’ in this dataset. There are two main reasons that a GPS coordinate cannot be allocated to a crash. Firstly, that the crash has been reported but the location was unknown. Secondly in a small number of instances, a crash may have occurred on a road which is not yet captured on the CAS spatial layer.
NumberOfLanes	Number of Lanes	The number(num) of lanes on the crash road.
objectThrownOrDropped	Object thrown or dropped	Derived variable to indicate how many times objects were thrown at or dropped on vehicles in the crash.
otherObject	Other Object	Derived variable to indicate how many times an object was struck in a crash and the object struck was not pre-defined. This variable includes stockpiled materials, rubbish bins, fallen poles, fallen trees, etc.
otherVehicleType	Other Vehicle Type	Derived variable to indicate how many other vehicles (not included in any other category) were involved in the crash.
overBank	Over Bank	Derived variable to indicate how many times an embankment was struck or driven over during a crash. This variable includes other vertical drops driven over during a crash.
parkedVehicle	Parked Vehicle	Derived variable to indicate how many times a parked or unattended vehicle was struck in the crash. This variable can include trailers.
phoneBoxEtc	Phone Box etc.	Derived variable to indicate how many times a telephone kiosk traffic signal controllers, bus shelters or other public furniture was struck in the crash
pedestrian	Pedestrian	Derived variable to indicate how many pedestrians were involved in the crash. This includes pedestrians on skateboards, scooters and wheelchairs.
postOrPole	Post or Pole	Derived variable 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	Region	Identifies the local government (LG) region. The boundaries match territorial local authority (TLA) boundaries

Attribute Name	Alias Name	Description
roadCharacter	Road Character	The general nature of the road. Possible values include ‘Bridge’, ‘Motorway Ramp’, ‘Rail crossing’ or ‘Nil’.
roadLane	Road Lane	The lane configuration of the road. Possible values : ‘1’ (one way), ‘2’ (two way), ‘M’ (for where a median exists), ‘O’ (for off-road lane configuations), ’ ’ (for unknown or invalid configuarations).
roadMarkings	Road Markings	The road markings at the crash site. Possible values: ‘Ped Crossing’ (for pedestrian crossings), ‘Raised Island’, ‘Painted Island’, ‘No Passing Lanes’, ‘Centre Line’, ‘No Marks’ or ’ Unknown’.
roadSurface	Road Surface	The road surface description applying at the crash site. Possible values: ‘Sealed’ or ‘Unsealed’.
roadworks	Road works	Derived variable 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	School Bus	Derived variable to indicate how many school buses were involved in the crash.
seriousInjuryCount	Serious Injury Count	A count of the number of serious injuries (inj) associated with this crash.
slipOrFlood	Slip or Flood	Derived variable to indicate how many times landslips, washouts or floods (excluding rivers) were objects struck in the crash
speedLimit	Speed Limit	The speed (spd) limit (lim) in force at the crash site at the time of the crash. May be a number, or ‘LSZ’ for a limited speed zone.
strayAnimal	Stray Animal	Derived variable to indicate how many times a stray animal(s) was struck in the crash. This variable includes wild animals such as pigs, goats, deer, straying farm animals, house pets and birds.
streetLight	Street Light	The street lighting at the time of the crash. Possible values ‘On’, ‘Off’, ‘None’ or ’ Unknown’.
suv	SUV	Derived variable to indicate how many SUVs were involved in the crash.
taxi	Taxi	Derived variable to indicate how many taxis were involved in the crash.
tlaId	TLA ID	The unique identifier for a territorial local authority (TLA). Each crash is assigned a TLA based on where the crash occurred.
tlaName	TLA Name	The name of the territorial local authority (TLA) the crash has been attributed.
temporarySpeedLimit	Temporary Speed Limit	The temporary (temp) speed (spd) limit (lim) at the crash site if one exists (e.g. for road works).
trafficControl	Traffic Control	The traffic control (ctrl) signals at the crash site. Possible values are ‘Traffic Signals’, ‘Stop Sign’, ‘Give Way Sign’, ‘Pointsman’, ‘School Patrol’, ‘Nil’ or ’ N/A’.
trafficIsland	Traffic Island	Derived variable to indicate how many times a traffic island, medians (excluding barriers)was struck in the crash.
trafficSign	Traffic Sign	Derived variable to indicate how many times ‘traffic signage’ (including traffic signals, their poles, bollards or roadside delineators) was struck in the crash.
train	Train	Derived variable to indicate how many times a train, rolling stock or jiggers was struck in the crash, whether stationary or moving
tree	Tree	Derived variable to indicate how many times trees or other growing items were struck during the crash.

Attribute Name	Alias Name	Description
truck	Truck	Derived variable to indicate how many trucks were involved in the crash.
unknownVehicleType	Unknown Vehicle Type	Derived variable to indicate how many vehicles were involved in the crash (where the vehicle type is unknown).
urban	Urban	A derived variable using the ‘spd_lim’ variable. Possible values are ‘Urban’ (urban, spd_lim < 80) or ‘Open Road’ (open road, spd_lim >=80 or ‘LSZ’).
vanOrUtility	Van or Utility	Derived variable to indicate how many vans or utes were involved in the crash.
vehicle	Vehicle	Derived variable 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	Water River	Derived variable to indicate how many times a body of water (including rivers, streams, lakes, the sea, tidal flates, canals, watercourses or swanps) was struck in the crash.
weatherA	Weather A	Indicates weather at the crash time/place. See wthr_b. Values that are possible are ‘Fine’, ‘Mist’, ‘Light Rain’, ‘Heavy Rain’, ‘Snow’, ‘Unknown’.
weatherB	Weather B	The weather at the crash time/place. See weather_a. Values ‘Frost’, ‘Strong Wind’ or ‘Unknown’.