Milestone 4 - Crash Severity Analysis

Yifei Liu

04/06/2021

Goal

The goal of my project is to identify and analyse the key factors that contribute to the severity of a crash in New Zealand based on the crash data provided by the New Zealand Transport Agency.

Data Source

The data was retrieved from Waka Kotahi NZ Transport Agency (NTZA)'s CAS opendata public repository. The data was downloaded as a spreadsheet (csv) from the below URL on 26/04/2021. It was last updated 19/04/2021 at the time of download.

https://opendata-nzta.opendata.arcgis.com/datasets/crash-analysis-system-cas-data-1

Data Processing & Exploration

The downloaded dataset is already in a tidy format, with over 746,000 observation rows, and 70 variable columns.

1.1 - Narrowing down Columns

The first step is to narrow down the columns to only the relevant ones, this was achieved by interpreting the NZTA column description and determining if the column is within scope of our analysis. For example, columns such as 'outdatedLocationDescription' were deemed as irrelevant and removed. Other columns that consisted of overly sparse data were also removed in this step.

This process resulted in the column count being reduced from 70 to 31. (Appendix 1.1)

```
dfDimAfter # Dimension of Dataframe after column selection
```

```
## [1] 731052     31
```

1.2 - Dealing with missing data

```
head(naBef,16) # Count of 'NA' in each column before data cleaning
```

```
##      crashSeverity         fatalCount   minorInjuryCount seriousInjuryCount
##                  0                136                136                136
##            bicycle                bus    carStationWagon              moped
##                  5                  5                  5                  5
##         motorcycle   otherVehicleType         pedestrian          schoolBus
##                  5                  5             722835                  5
##                suv               taxi              train              truck
##                  5                  5             450755                  5
```

As we can see above, there are a lot of columns with NA values, so the second step is to analyse and clean up our data.

Due to the discrete nature of our dataset, calculation-based imputation cannot be used. This means the NA data either has to be dropped or replaced with 0/None.

For example, we discovered a pattern where the same 136 rows of 'fatalCount', 'minorInjuryCount' and 'seriousInjuryCount' contained NA. This pattern is a good indication of data 'missing not at random'. In this case I have deemed these rows to be faulty and dropped them entirely from the dataset. Luckily we have over 700,000 rows so dropping a small amount (136) of rows will not reduce the quality of our data by a significant margin.

Another example worth discussing is the 'vehicle information' column (bus, bicycle, pedestrian... etc). A majority of the rows (90%+) contain at least one vehicle type where the data is NA, we would be losing too much information if we were to drop a row if it contains 'NA' so I have decided to replace missing values with 0 in this context (i.e. If 'bus' information has not been reported, assume the value was 0).

There are also other data quality issues, such as columns which contain both 'Null' and 'None'. This issue is likely caused by the constant version updates to the CAS open data, older versions recorded normal 'weather' conditions as 'Null', while recent versions records them as 'None'. We should normalize these conflicting occurrences into a single value - 'None'.

At the end of appendix 1.2, we can observe that our data is free of NA values, with all problematic column dropped or replaced in a way that makes sense.

```
head(naAft,16) # Count of 'NA' in each column after data cleaning
```

```
##      crashSeverity         fatalCount   minorInjuryCount seriousInjuryCount
##                  0                  0                  0                  0
##            bicycle                bus    carStationWagon              moped
##                  0                  0                  0                  0
##         motorcycle   otherVehicleType         pedestrian          schoolBus
##                  0                  0                  0                  0
##                suv               taxi              train              truck
##                  0                  0                  0                  0
```

1.3 - Mutating new useful columns

The dataset contains 13 columns, each a unique type of vehicle (bicycle, bus, train... etc). The number stored in that column specifies how many of that vehicle type was involved in the crash. I created a generic count of all vehicles involved in the crash by mutating a new column that calculates the sum of vehicles involved irrespective of vehicle type.

```
# Table illustrating the structure of vehicle information
kable(head(df[,c('bicycle', 'carStationWagon', 'motorcycle', 'truck', 'vanOrUtility'
                 ,'totalVehiclesInvolved')], rows=3))
```

bicycle	carStationWagon	motorcycle	truck	vanOrUtility	totalVehiclesInvolved
0	1	0	0	0	1
0	2	0	0	0	2
0	0	0	1	1	2
0	2	0	0	0	2
0	1	0	0	1	2
0	0	1	0	0	1

Additionally, the crash severity column was also transformed into a factor. This will allow for greater flexibility later as factors allow our categorical data to be stored both as integers and strings. (Appendix 1.3)

1.4 - Balancing dataset & Train-Test Split

In order to validate our models in an unbiased manner, we have to split our data into a Train-Test split. I have chosen a 80-20 test-train split.

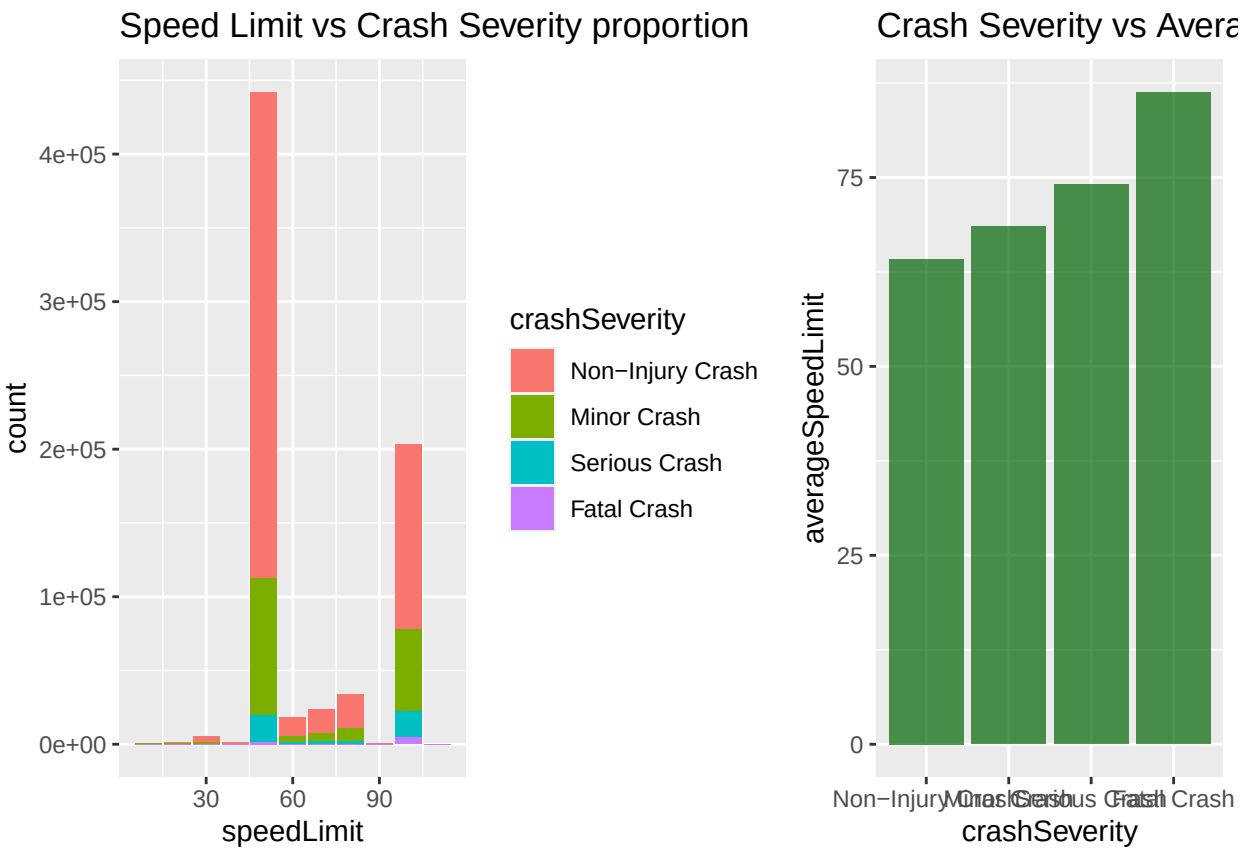
During the model-building process, I noticed that my data was extremely unbalanced, with non-injury crashes taking up almost 90% of the entries. After extended experimentation with the models I decided to deal with this data issue by downsampling the majority classes to match the minority class, as it was both faster & provided slightly better results over the class weighting approach.

```
kable(balance) # Distribution of classes before and after downsampling
```

description	NonInjury	Minor	Serious	Fatal
Training Set	411746	134643	33032	5420
Downsampled	5420	5420	5420	5420

1.5 - Summary of Data Exploration

```
ggarrange(plot1,plot2, nrow=1)
```



As we can see from the first graph, the majority of reported crashes have happened on 50km/h roads, followed by 100km/h and then 80km/h. I believe this pattern is more attributed to the distribution of speed limits on NZ roads (majority of roads are 50km/h in NZ).

I have also averaged the speed limit of all crash severities (from Non Injury Fatal). As we can see, as average speed limit increases, the crash severity also increases with it. This graph supports our initial exploratory direction of positive relationship between higher speed limits and higher crash severity.

```
knitr::kable(lightSeverity)
```

light	Non-Injury Crash	Minor Crash	Serious Crash	Fatal Crash
Bright sun	69.4%	23.9%	5.8%	0.8%
Dark	71.0%	21.6%	6.2%	1.3%
Overcast	71.2%	23.1%	4.9%	0.7%
Twilight	70.0%	23.4%	5.7%	0.9%

```
knitr::kable(weatherSeverity)
```

weatherA	Non-Injury Crash	Minor Crash	Serious Crash	Fatal Crash
Fine	70.1%	23.1%	5.9%	0.9%
Hail or Sleet	67.3%	19.2%	13.5%	0.0%
Heavy rain	71.0%	23.0%	5.0%	0.9%
Light rain	72.3%	22.2%	4.7%	0.8%
Mist or Fog	67.4%	24.7%	6.5%	1.5%
Snow	70.1%	22.8%	6.3%	0.8%

The two tables above are summaries of crash severity against 'lighting' and 'weather' conditions respectively. If we look towards the rightmost column representing fatal crashes, we can indeed see that 'Dark' lighting conditions and 'Mist or Fog' weather results in a higher fatal crash rate. The evidence above is in support of our second exploratory direction of positive relationship between poor lighting conditions / harsh weather conditions.
(Detailed code included in Appendix)

Analytical Plan

The nature of our investigation has been identified as a multi-class classification problem, I decided to use tree-based classification models because they are known to have favourable performance in this context.

The three models I have chosen are **Decision Trees** (rpart), **Random Forests** (ranger), and **Extreme Gradient Boost** (xgboost).

Decision Trees were chosen because it was easy to implement and allowed for a quick overview of the quality of our data. Through fitting the initial DT it was discovered that the columns 'fatalCount', 'seriousInjuryCount', and 'minorInjuryCount' were allowing the model to 'cheat' when it comes to predicting the crash severity. A number of irrelevant features were vetted in this fashion and excluded from subsequent model fittings where the results actually mattered.

Since the purpose of our investigation were to predict & find variable importance of NZ crash severity, **Random Forests** seemed to be the obvious choice. With the refined list of features we now have from fitting the DT previously, two ranger forests were fitted - one on all the imbalanced training set, the other on the downsampled/balanced training set. Downsampling was chosen as a class balancing technique as it was both faster and provided a better result when compared to class weighting for our dataset.

XGBoost was chosen as the final analytical method because I wanted some comparable models to Random Forests. The goal of this choice was to allow comparison between bagging and boosting methods in order

to determine which one is more appropriate for my dataset. Similarly to Random Forests, two XGBoost models were fitted - using imbalanced training data as well as downsampled/balanced training data.

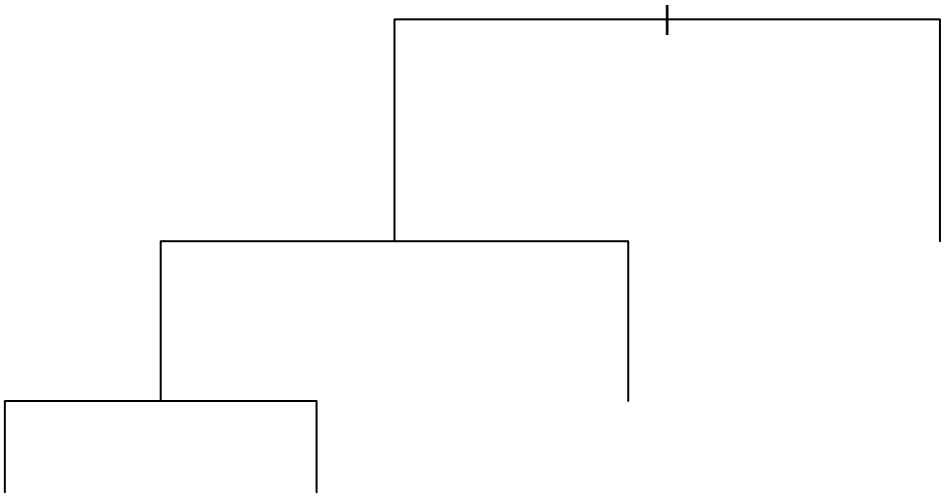
Results

```
printcp(tree)
```

Decision Tree

```
##
## Classification tree:
## rpart(formula = crashSeverity ~ bicycle + bus + carStationWagon +
##     moped + motorcycle + otherVehicleType + pedestrian + schoolBus +
##     suv + taxi + train + truck + unknownVehicleType + vanOrUtility,
##     data = train)
##
## Variables actually used in tree construction:
## [1] bicycle    motorcycle pedestrian
##
## Root node error: 173095/584841 = 0.29597
##
## n= 584841
##
##         CP nsplit rel error  xerror      xstd
## 1 0.053115      0   1.00000 1.00000 0.0020168
## 2 0.038251      1   0.94688 0.94688 0.0019843
## 3 0.021843      2   0.90863 0.90863 0.0019590
## 4 0.010000      3   0.88679 0.88679 0.0019438
```

```
plot(tree)
```



From the above information, we can infer that when 'pedestrians' are involved in a crash, the severity rate is the highest, this is followed by 'motorcycles' and 'bicycles'. The results are convincing as my research prior to this milestone suggested motorcycles were up to 30 times more likely to have fatal crashes than cars, and that direct contact with a pedestrian will undoubtedly result in the highest fatality rate.

```
tree_matrix$table
```

```
##                   Reference
## Prediction         Non-Injury Crash Minor Crash Serious Crash Fatal Crash
##   Non-Injury Crash           100263       25975          5124        1014
##   Minor Crash                  2674        7686          3134         341
##   Serious Crash                   0           0             0           0
##   Fatal Crash                     0           0             0           0
```

As we can see from the confusion matrix above, the decision tree has done a poor job of predicting the crash severity, due to the imbalanced nature of the dataset. This result prompted me to revise my Analytical plan to ensure the RF and XGB implentations are not affected by the same class balance problems our initial tree has faced.

```
forest1_matrix$table # Imbalanced Random Forest Confusion Matrix
```

Random Forest

```
##                   Reference
## Prediction         Non-Injury Crash Minor Crash Serious Crash Fatal Crash
##   Non-Injury Crash           100419       26487          5625        1123
##   Minor Crash                  2518        7174          2633         232
##   Serious Crash                   0           0             0           0
##   Fatal Crash                     0           0             0           0
```

```
forest2_matrix$table # Balanced Random Forest Confusion Matrix
```

```
##                   Reference
## Prediction         Non-Injury Crash Minor Crash Serious Crash Fatal Crash
##   Non-Injury Crash            72699       14060          1929         225
##   Minor Crash                  6084        5046           953          59
##   Serious Crash                2894        5074          2103         173
##   Fatal Crash                 21260        9481          3273         898
```

```
forest2_matrix$overall # Balanced Random Forest Overall Accuracy
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.5522567      0.1808279      0.5497034      0.5548079      0.7040305
## AccuracyPValue  McnemarPValue
##      1.0000000      0.0000000
```

By balancing the class distribution before fitting the Random Forest, we are greatly reducing the bias towards the 'Non-Injury' class. From the confusion matrix above we can see that the balanced data suffered a decrease in accuracy (73% ->54%) in order to achieve a higher Sensitivity (29% -> 45%) and Specificity (79% -> 82%). After extended experimentation we were unable to fit a RF with reliable accuracy (90%+). I have observed that after the class balancing, minor and serious crashes were being predicted with higher accuracy, however the RF was still unable to predict fatal crashes reliably.
(Confusion matrix only partially shown, full confusion matrix code can be found in appendix.)

```
# Variable importance sorted from most important to least important
rev(sort(head(importance(forest2),8)))
```

```
##  carStationWagon       pedestrian       motorcycle          bicycle
##       377.701077       223.690330       195.043239       128.418386
##            moped              bus otherVehicleType        schoolBus
##        40.808554        28.607903        17.312673         4.088929
```

According to the result from our balanced RF, these are the 8 most influential features that decide a crash's severity. From this we can infer that vehicle type has stronger correlation with crash severity when compared to other factors such as weather, lighting etc.

```
gb_imb$table # Imbalanced XGBoost Confusion Matrix
```

XGBoost

```
##                  Reference
## Prediction         Non-Injury Crash Minor Crash Serious Crash Fatal Crash
##   Non-Injury Crash           101060       26857          5472        1031
##   Minor Crash                  1877        6804          2786         324
##   Serious Crash                   0           0             0           0
##   Fatal Crash                     0           0             0           0
```

```
gb_bal$table # Balanced XGBoost Confusion Matrix
```

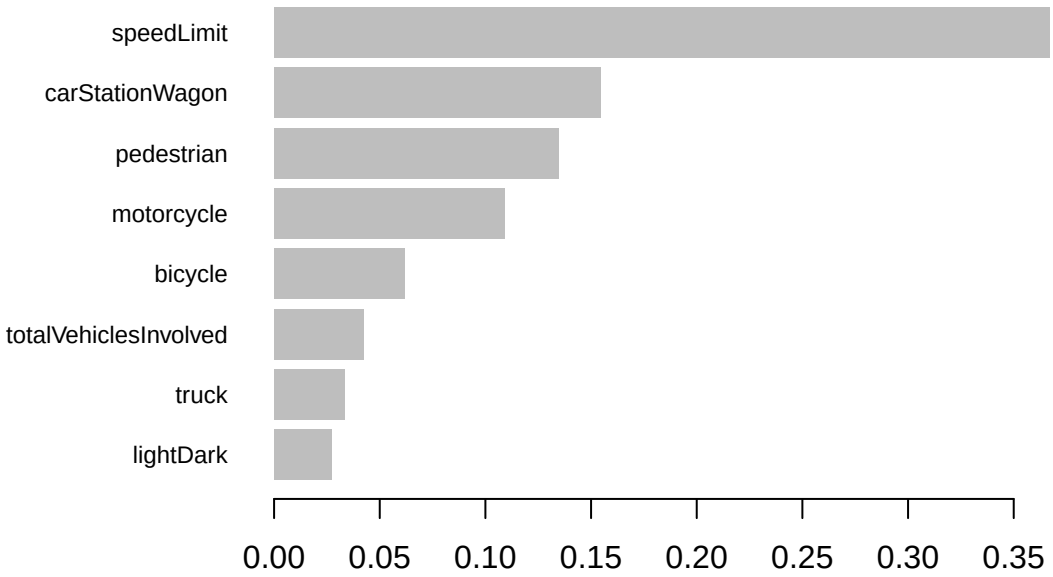
```
##                  Reference
## Prediction         Non-Injury Crash Minor Crash Serious Crash Fatal Crash
##   Non-Injury Crash            70322       13564          1796         200
##   Minor Crash                  3855        3702           746          54
##   Serious Crash                3845        5954          2377         181
##   Fatal Crash                 24915       10441          3339         920
```

```
gb_bal$overall # Balanced XGBoost Overall Accuracy
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.5288316      0.1675773      0.5262690      0.5313931      0.7040305
## AccuracyPValue  McnemarPValue
##      1.0000000      0.0000000
```

The same experimental design is carried out for XGBoost, with the addition of parameter tuning via the use of **xgb.cv**. After the parameters have been decided it was found that XGB had a very simillar performance compared to RF, however the extra setup and parameter tuning effort made it an inefficient process. For our specific dataset, we concluded that boosting methods did not offer a clear advantage over bagging as both methods generated virtually identical results.
(Detailed code in Appendix)

```
xgb.plot.importance(importance_matrix = imp2, top_n = 8)
```



This is a visualization of the 8 most important variables according to XGBoost. carStationWagon, pedestrian, and motorcycle were agreed to be in the top 4 most influential features. Bicycle and truck were also present in both lists however ranked in different orders.

```
kable(conc2)
```

Conclusion:

model	imp1	imp2	imp3	imp4	imp5	imp6	imp7	imp8
RF Balanced	car	pedestrian	motorcycle	truck	bicycle	suv	moped	bus
XGB Balanced	speedLimit	car	pedestrian	motorcycle	bicycle	totalVehicles	truck	lightDark

According to my models, a combination of speed limit, lighting condition, and vehicle types were the most influential features of crash severity. This result indicates that the speed & type of vehicle were more correlated with crash severity when compared to secondary features such as weather, road finishing etc.

```
kable(conc)
```

metric	RF_imbal	RF_bal	RF_diff	XGB_imbal	XGB_bal	XGB_diff
Accuracy	73.20%	53.97%	-19.23%	73.77%	52.88%	-20.86%
Sensitivity	28.75%	42.68%	+13.93%	29.54%	44.28%	+14.74%
Specificity	78.55%	80.69%	+2.14%	79.61%	81.50%	+1.89%

As described above, balancing the dataset seemed to reduce overall accuracy by a significant margin, but

increase Sensitivitity (True positive rate) and Specificity (True negative rate) at a smaller rate. This meant that the model was getting more of the minority classes right when compared with the model trained on the imbalanced dataset, however neither versions of the models were able to predict crash severity with a good enough confidence to be usable in a real-life scenario.

Discussion

The strength of our data was that it was well-organized and tidy, with a large amount of samples for models to train with. The large amount of samples meant we could also drop chunks of unreliable data without impacting the overall model by a significant margin.

The context as well as the feature names were able to be easily understood & interpreted by us as humans. This meant that the presentation of our findings could be easily communicated with stakeholders in a relatable fashion.

Though the data retrieved from NZTA's public repository comes handy in a tidy format, it was difficult to utilize this data with tree-based models to predict the crash severity. The main data issue discovered were the effects of the heavily imbalanced dataset.

Fatal crashes occupy only a small portion of the CAS dataset. Additionally, the crash data is known to be weighted heavily against fatal crashes. As advised by NZTA, this repository only stores information that were officially reported to law enforcement authorities, therefore a siginificant portion of minor accidents and civil disputes that were able to be settled independently of NZ Police / NZTA would not appear in the dataset, further skewing the distribution of our classes.

This imbalance in class proportions had a significant impact across all of our models, techniques such as hyperparameter tuning, manual class weighting, as well as down/upsampling were extensively experimented with and an appropriate solution was unable to be reached (My goal was minimum 95% accuracy predictions).

The data also contained a mixture of numerical and categorical data, which required additional effort to transform the training data before XGBoost was able to function correctly.

The overall results suggest that the downsample RF would be the most appropriate model to use out of all the proposed methods, as it was both fast to train and had decent Sensitivity/Specificity metrics. For our proposed problem and dataset there is no tangible benefit to spend extra time tuning XGBoost just to find virtually identical results as the Random Forest.

In conclusion, this project did more to make me gain familiarity with tree-based classification model best practices as opposed to producing an usable machine learning model. I was able to get hand-on experience with RF & XGB parameter tuning, class weighting, and down/upsampling techniques which lead to a deeper understanding of these common data science practices. In the future I would also like to experiment with sentiment-analysis based projects as I do not have as much experience in those types of projects.

Appendix

Appendix 1.1

```
set.seed(1234)
library(knitr)
library(dplyr)
library(tidyr)
library(caret)
library(ggplot2)
library(egg)
library(janitor)
library(rpart)
library(ranger)
keep_cols <- as.vector(c('crashSeverity', 'fatalCount', 'minorInjuryCount', 'seriousInjuryCount',

                         'bicycle', 'bus', 'carStationWagon', 'moped', 'motorcycle',
                         'otherVehicleType', 'pedestrian', 'schoolBus', 'suv', 'taxi',
                         'train', 'truck', 'unknownVehicleType', 'vanOrUtility',           #SET NA TO 0

                         'crashLocation1', 'crashSHDescription',
                         'region', 'roadSurface', 'flatHill',
                         'speedLimit',

                         'crashYear', 'holiday',

                         'light', 'streetLight', 'weatherA', 'weatherB'
                         ))

df <- read.csv("Crash_Analysis_System_(CAS)_Data.csv")  %>%
  select(all_of(keep_cols)) %>%
  as.data.frame()

dfDimBefore <-  dim(df)
dfDimBefore
```

Appendix 1.2:

```
# Check for existence of NA values in our dataframe - we can see a lot of columns with missing data.
naBef <- sapply(df, function(x) sum(is.na(x)))

# fatalCount, minorInjuryCount, seriousInjuryCount - 136 missing rows where all three values are missin
df <- drop_na(df, 'fatalCount')

# vehicle information - 5 rows where all vehicle information is NA
df <- drop_na(df, 'bicycle')

# Drop 382 rows where no speed limit has been recorded
```

```r
df <- drop_na(df, 'speedLimit')

# If no number of pedestrian/train specified, assume 0
df$pedestrian[is.na(df$pedestrian)] <- 0
df$train[is.na(df$train)] <- 0

#Check if our dataframe still contains any NA values - we can see no columns contain any NA values
naAft <- sapply(df, function(x) sum(is.na(x)))

# Normalize all 'Null' or '' values to 'None'
df[df=="Null"] <- 'None'
df$holiday[df$holiday==""] <- 'None'


# Removing low-quality rows where there are no weather/light/hill/roadsurface/etc information
df <- df[df$weatherA !="None", ]
df <- df[df$light !="Unknown", ]
df <- df[df$flatHill !="None", ]
df <- df[df$roadSurface !="None", ]
df <- df[df$region !="", ]
df <- df[df$crashSHDescription !="Unknown", ]

# Removing low quality rows where speed limit has been recorded inaccurately
df <- filter(df, !(speedLimit %in% c(2, 5, 6, 15, 51, 61) ))
```

Appendix 1.3:

```r


# Create row calculating the sum of all vehicle types involved
vehicles <- c('bicycle', 'bus', 'carStationWagon', 'moped', 'motorcycle',
              'otherVehicleType', 'pedestrian', 'schoolBus', 'suv', 'taxi',
              'train', 'truck', 'unknownVehicleType', 'vanOrUtility')

# Create new column that is a sum of all vehicles involved
df <- mutate(df, totalVehiclesInvolved = rowSums(df[vehicles]))

# Drop all rows where there were 0 vehicles involved
df <- df[df$totalVehiclesInvolved !=0, ]

# Transforming crashSeverity into numerical representation
df$crashSeverity[df$crashSeverity=="Non-Injury Crash"] <- 1
df$crashSeverity[df$crashSeverity=="Minor Crash"] <- 2
df$crashSeverity[df$crashSeverity=="Serious Crash"] <- 3
df$crashSeverity[df$crashSeverity=="Fatal Crash"] <- 4

dfDimAfter <- dim(df)

# Comparison of dimensions before and after data cleaning
dfDimBefore
dfDimAfter
```

Appendix 1.4:

```
# Randomly select 20% of data to be the test set.
test.index <- createDataPartition(df$crashSeverity, p = 0.2, list = FALSE)
test  <- df[test.index,]

# Select 80% of remaining data to be the training set.
train <- df[-test.index,]

# Create balanced training via downsampling.
train_ds<- downSample(x = train,
                      y = train$crashSeverity)
train_ds <- within(train_ds, rm(Class))

# Manual table creation to be used in the report - visualizing downsampling
balance <- data.frame(description =   c('Training Set','Downsampled Training'),
                      NonInjury =   c('411746','5420'),
                      Minor =       c('134643','5420'),
                      Serious =     c('33032','5420'),
                      Fatal =       c('5420','5420')
                      )
```

Appendix 1.5:

```
library(ggplot2)
library(egg)

plot1 <- df %>%
  ggplot(aes(x = speedLimit, fill = crashSeverity)) +
  geom_bar() +
  labs(title = 'Speed Limit vs Crash Severity proportion')

plot2 <- group_by(df, crashSeverity) %>%
  summarise(averageSpeedLimit = mean(speedLimit)) %>%
  arrange(desc(averageSpeedLimit)) %>%
  ggplot(aes(x=crashSeverity, y=averageSpeedLimit )) +
  geom_bar(fill='darkgreen', alpha=0.7, stat='identity') +
  labs(title = 'Crash Severity vs Average Speed limit')

library(janitor)
lightSeverity <- tabyl(df, light, crashSeverity)%>%
  adorn_percentages("row")%>%
  adorn_pct_formatting()

weatherSeverity <- tabyl(df, weatherA, crashSeverity)%>%
  adorn_percentages("row")%>%
  adorn_pct_formatting()
```

Appendix: Analytical Plan:

```r
# Fitting Decision Tree
tree <- rpart(crashSeverity ~ bicycle + bus + carStationWagon + moped + motorcycle +
                          otherVehicleType + pedestrian + schoolBus + suv + taxi +
                          train + truck + unknownVehicleType + vanOrUtility, data = train)

# Predicting with DT
pred <- predict(tree, test, type = 'class')

# DT Confusion Matrix
tree_matrix <- confusionMatrix( pred,test$crashSeverity)


# Fitting two Random Forests
forest <-ranger(crashSeverity ~ .-crashLocation1-Class-minorInjuryCount-seriousInjuryCount-fatalCount-c

forest2 <-ranger(crashSeverity ~ .-crashLocation1-Class-minorInjuryCount-seriousInjuryCount-fatalCount-

# Forest Feature Importance
forest_importance <- sort(importance(forest))
forest2_importance <- sort(importance(forest2))

# RF Predictions
old_pred <- predict(forest, test)
new_pred <- predict(forest2, test)

old_pred_factor <- factor(old_pred$predictions, levels = c("Non-Injury Crash", "Minor Crash", "Serious
new_pred_factor <- factor(new_pred$predictions, levels = c("Non-Injury Crash", "Minor Crash", "Serious

# RF Confusion Matrix
forest1_matrix <- confusionMatrix(old_pred_factor, test$crashSeverity)
forest2_matrix <- confusionMatrix(new_pred_factor, test$crashSeverity)



library(xgboost)
library(Matrix)


# XGB Preprocessing
sparse_matrix <- sparse.model.matrix(crashSeverity ~ .-crashLocation1-minorInjuryCount-seriousInjuryCou
sparse_matrix2 <- sparse.model.matrix(crashSeverity ~ .-crashLocation1-minorInjuryCount-seriousInjuryCo
sparse_test <- sparse.model.matrix(crashSeverity ~ .-crashLocation1-minorInjuryCount-seriousInjuryCount-

# Convert classes to integers for xgboost
temp2 <- as.numeric(train$crashSeverity)-1
temp3 <- as.numeric(train_ds$crashSeverity)-1


# Parameter Tuning
xgb.cv(data=sparse_matrix,label = temp2, booster = 'gbtree', num_class = 4, max_depth = 2, eta = 0.3, n

xgb.cv(data=sparse_matrix2,label = temp3, booster = 'gbtree', num_class = 4, max_depth = 2, eta = 0.3,
```

```r
# Fitting XGBoost models
xgb <- xgboost(data=sparse_matrix,label=temp2, booster = 'gbtree', num_class = 4, max_depth = 2, eta = 

xgb2 <- xgboost(data=sparse_matrix2,label=temp3, booster = 'gbtree', num_class = 4, max_depth = 2, eta =

# XGB Predictions
yhat_xgb <- predict(xgb, newdata=sparse_test, reshape = T)
yhat_xgb2 <- predict(xgb2, newdata=sparse_test, reshape = T)

library(plyr)
revalued <- mapvalues(yhat_xgb, from = c(0,1), to = c("Non-Injury Crash","Minor Crash"))
revalued2 <- mapvalues(yhat_xgb2, from = c(0,1,2,3), to = c("Non-Injury Crash", "Minor Crash", "Serious

revalued <- factor(revalued, levels = c("Non-Injury Crash", "Minor Crash", "Serious Crash","Fatal Crash
revalued2 <- factor(revalued2, levels = c("Non-Injury Crash", "Minor Crash", "Serious Crash","Fatal Cras

# XGB Feature Importance
imp1 <- xgb.importance(colnames(sparse_matrix), model = xgb)
imp2 <- xgb.importance(colnames(sparse_matrix2), model = xgb2)

# XGB Confusion Matrix
gb_imb <- confusionMatrix(revalued,   test$crashSeverity)
gb_bal <- confusionMatrix(revalued2,  test$crashSeverity)

# Custom tables to present accuracy changes in report
conc <- data.frame(metric = c('Accuracy','Sensitivity','Specificity' ),
                   RF_imbal =  c('73.20%','28.75%','78.55%' ),
                   RF_bal =    c('53.97%','42.68%','80.69%' ),
                   RF_diff =   c('-19.23%','+13.93%','+2.14%' ),
                   XGB_imbal = c('73.77%','29.54%','79.61%' ),
                   XGB_bal =   c('52.88%','44.28%','81.50%' ),
                   XGB_diff =  c('-20.86%','+14.74%','+1.89%' )
                   )

# Custom table of top 8 significant features
conc2 <- data.frame(model = c('RF Balanced', 'XGB Balanced'),
                    imp1 = c('car','speedLimit'),
                    imp2 = c('pedestrian','car'),
                    imp3 = c('motorcycle','pedestrian'),
                    imp4 = c('truck','motorcycle'),
                    imp5 = c('bicycle','bicycle'),
                    imp6 = c('suv','totalVehicles'),
                    imp7 = c('moped','truck'),
                    imp8 = c('bus','lightDark'))
```

EOF: