Code

Project 2 - Australian Road Fatalities Modelling

J. Barrett 22683565 and D. Chegwidden 21282744

CITS4009 Sem 2

Introduction

This analysis will explore Australian road fatatilites from 1989 - 2019, utilising data sets provided by the Australian Government's Australian Road Deaths Database (ARDD).

There are two datasets, one where each observation is an individual fatality, and one where each observation is a crash involving one or more fatalities. For the purposes of our modelling, we will focus on the 'Fatalities' data set, and draw observations from the 'Crashes' data set if necessary.

In this analysis, we aim to uncover patterns and relationships in the data, and form models that allow us to better understand variables of interest. This notebook will be divided into two core sections:

1. Classification

For our classification task, we aim to develop models that help us explain the gender of a fatality on Australian roads. The modelling techniques used are:

- Single variable models
- Binary logistic regression
- Decision trees

2. Clustering

Clustering analysis is an unsupervised method that helps us to see relationships between observations in our data. For this task, we use the Heirarchical Clustering method to explore similarities and differences between differently aged road users of differing types (eg. Motorcyclist, Car Passenger), according to the speed and time of the accident.

Data Import / Transformation

This data is publicy available from https://www.data.gov.au/ (https://www.data.gov.au/), and updated monthly:

Hide

```
fatalities <- read_csv("https://data.gov.au/data/dataset/5b530fb8-526e-4fbf-b0f6-aa24e84e4277/resource/fd646fdc-7788-4bea-a736-e4aeb0dd09a8/download/ardd_fatalities.csv", na = c("-9", ""))

crashes <- read_csv("https://data.gov.au/data/dataset/5b530fb8-526e-4fbf-b0f6-aa24e84e4277/resource/d54f7465-74b8-4fff-8653-37e724d0ebbb/download/bitre_ardd_fatal_crashes.csv", na = c("-9", ""))
```

Before we begin modelling, some care needs to be taken with the handling of different variables within the data. From previous exploratory data analysis performed on this data, we know that some variables contain large proportions of missing values, providing inconsistent and limited value. We will drop these variables, and filter out the small number of observations where:

- `Gender` is not equal to Male or Female,
- A value for `Age` is missing,
- A value for `Time` is missing,
- A value for `Bus Involvement` is missing,

as there are so few observations meeting these criteria and they are key data points for our modelling.

Hide

```
drop_col <- c("National Remoteness Areas", "SA4 Name 2016",
              "National LGA Name 2017", "National Road Type")

df <- fatalities %>%
    select(-drop_col) %>%
    filter(Gender != "Unspecified") %>%
    filter(!is.na(Age), !is.na(Time), !is.na(`Bus Involvement`)) %>%
    mutate_all(funs(replace_na(., "Missing"))) %>%
    rename(H.Truck.Involvement = `Heavy Rigid Truck Involvement`,
           A.Truck.Involvement = `Articulated Truck Involvement`) %>%
    as.data.frame()

colnames(df) <- str_replace_all(colnames(df), c(" " = "."), ("\n" = "."))  # stan
dardise column names
df$Road.User <- revalue(df$Road.User, c("Other/-9" = "Other"))

df <- droplevels.data.frame(df) # dropping vacant levels
```

We also notice that for about 40% of our data, we don't have entries for `H.Truck.Involvement` :

Hide

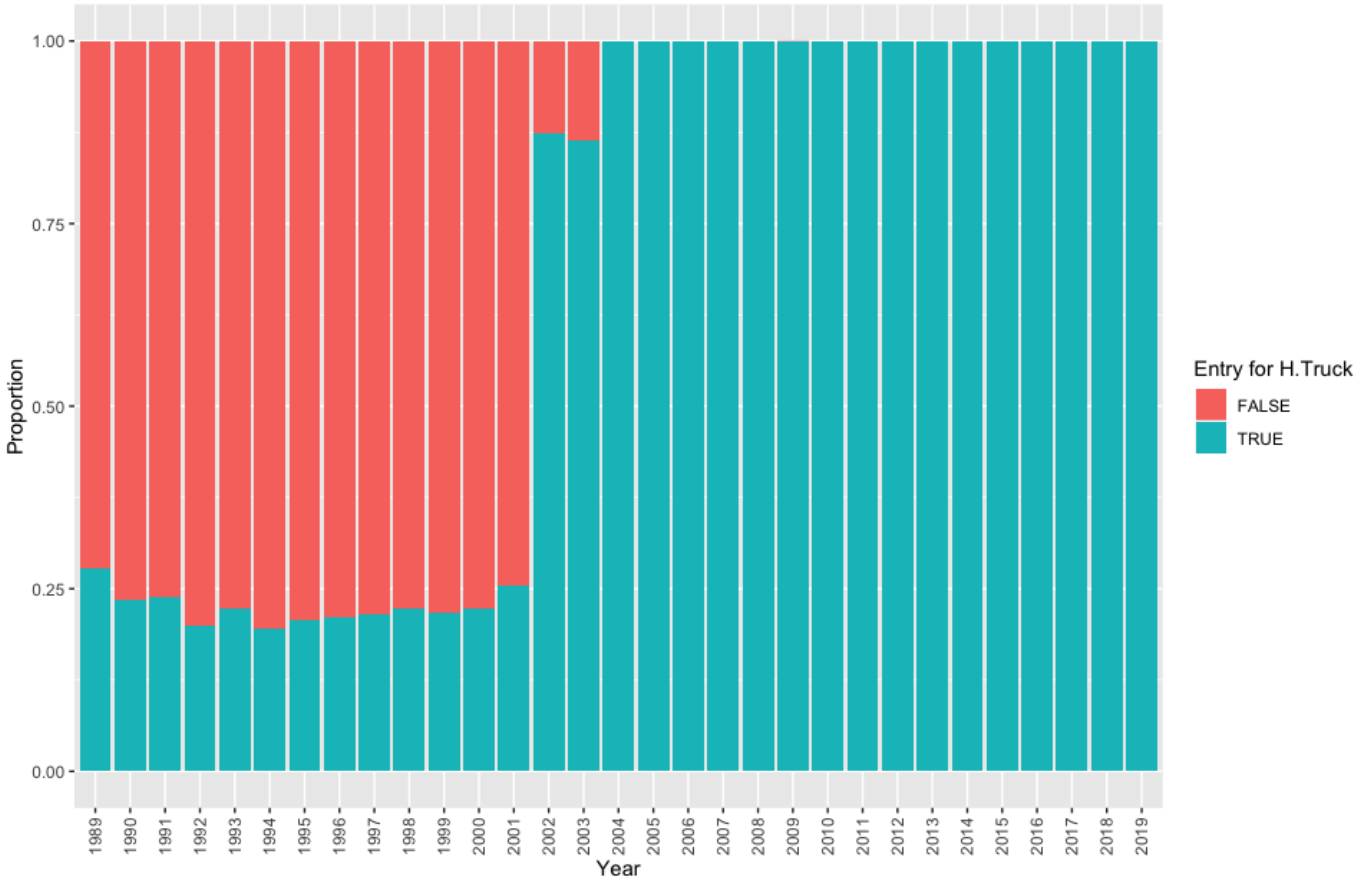
```
paste0(round(sum(df$H.Truck.Involvement == "Missing") / nrow(df) * 100, 2), "%")
```

```
[1] "40.45%"
```

We will take a look to see how data for this variable is distributed across the years the data was recorded:

Hide

```
with(df, table(Year, H.Truck.Involvement != "Missing")) %>%
    data.frame() %>%
    ggplot(aes(x = Year, y = Freq, fill = Var2)) +
        geom_bar(stat="identity", position = "fill") +
        theme(axis.text.x = element_text(angle = 90, vjust = 0.5)) +
        labs(fill = "Entry for H.Truck", y = "Proportion")
```



We see that records for `H.Truck.Involvement` were only taken consistently from 2002-2004 onwards. This strong relationship between `H.Truck.Involvement` and `Year` could introduce multicollinearity in our models moving forward, especially if the year of a crash ends up being a good predictor of `Gender`. One way to decide how to handle this variable, is to see if this variable individually has any relationship with the gender of a fatility, and if not, we will not include it in any of our modelling. To accomplish this we can perform a Chi-Squared test of homogeneity:

Hide

```
df %>%
  filter(H.Truck.Involvement != "Missing") %>%
  droplevels() %>%
  with(table(Gender, H.Truck.Involvement)) %>%
  chisq.test()
```

```

    Pearson's Chi-squared test with Yates' continuity correction

data:  .
X-squared = 0.037987, df = 1, p-value = 0.8455
```

This high p-value of 0.8455 leads us to fail to reject the null hypothesis of homogeneity between the proportion of male/female fatalities and the proportion of fatalities involving a heavy rigid truck. We conclude that there is no evidence to suggest that the involvement of a heavy rigid truck has any relationship to the gender of a fatility, and as such, will not include this variable in any of our modelling.

Hide

```
df <- df %>%
  select(-H.Truck.Involvement)
```

Given that the exact time of a crash may not be as of as much interest to us as the more general time period the crash took place in, we create a new categorical variable `Time.Period` for later use:

Hide

```
df <- within(df, {
  Time.Period <- NA
  Time.Period[Time >= hms(hours = 0) & Time <= hms(hours = 3, minutes = 59)] <-
"Late Night"
  Time.Period[Time >= hms(hours = 4) & Time <= hms(hours = 7, minutes = 59)] <-
"Early Morning"
  Time.Period[Time >= hms(hours = 8) & Time <= hms(hours = 11, minutes = 59)] <-
"Mid Morning"
  Time.Period[Time >= hms(hours = 12) & Time <= hms(hours = 15, minutes = 59)] <-
"Early Afternoon"
  Time.Period[Time >= hms(hours = 16) & Time <= hms(hours = 19, minutes = 59)] <-
"Afternoon / Evening"
  Time.Period[Time >= hms(hours = 20) & Time <= hms(hours = 23, minutes = 59)] <-
"Night"
})

df$Time.Period <- factor(df$Time.Period,
    levels = c("Early Morning", "Mid Morning", "Early Afternoon",
               "Afternoon / Evening", "Night", "Late Night"))

df$Time <- as.numeric(df$Time)
```

Lastly, we trim down some of the longer category names in our `Road.User` variable:

Hide

```
df <- df %>%
    mutate(Road.User = factor(str_replace(Road.User, "Motorcycle pillion passenge
r", "MC. Pass.")),
           Road.User = factor(str_replace(Road.User, "Motorcycle rider", "MC. Rid
er")),
           Road.User = factor(str_replace(Road.User, "Pedal cyclist", "Cyclist"
)))
```

---

1. Classification

In this section we aim to create models that explain the gender of an Australian road fatality.

1.1 Classification Transformation

Our first step will be to split our data into Training, Calibration and Test data sets for our models. Our initial split will be:

- Train: 90%

- Test: 10%

Hide

```
df_ <- df %>%
  select(-Crash.ID) # removing ID from any modelling data sets

set.seed(4009)
smpl_size <- floor(0.9 * nrow(df_))
train_idx <- sample(seq_len(nrow(df_)), size = smpl_size)

trainAll <- df_[train_idx, ]
test <- df_[-train_idx, ]
```

We then set our response variable as `Gender` and the positive outcome for it as “Female”, then further split our training data into:

- Train: 90%
- Calibration: 10%

This allows us to perform cross-validation during our single variable analaysis.

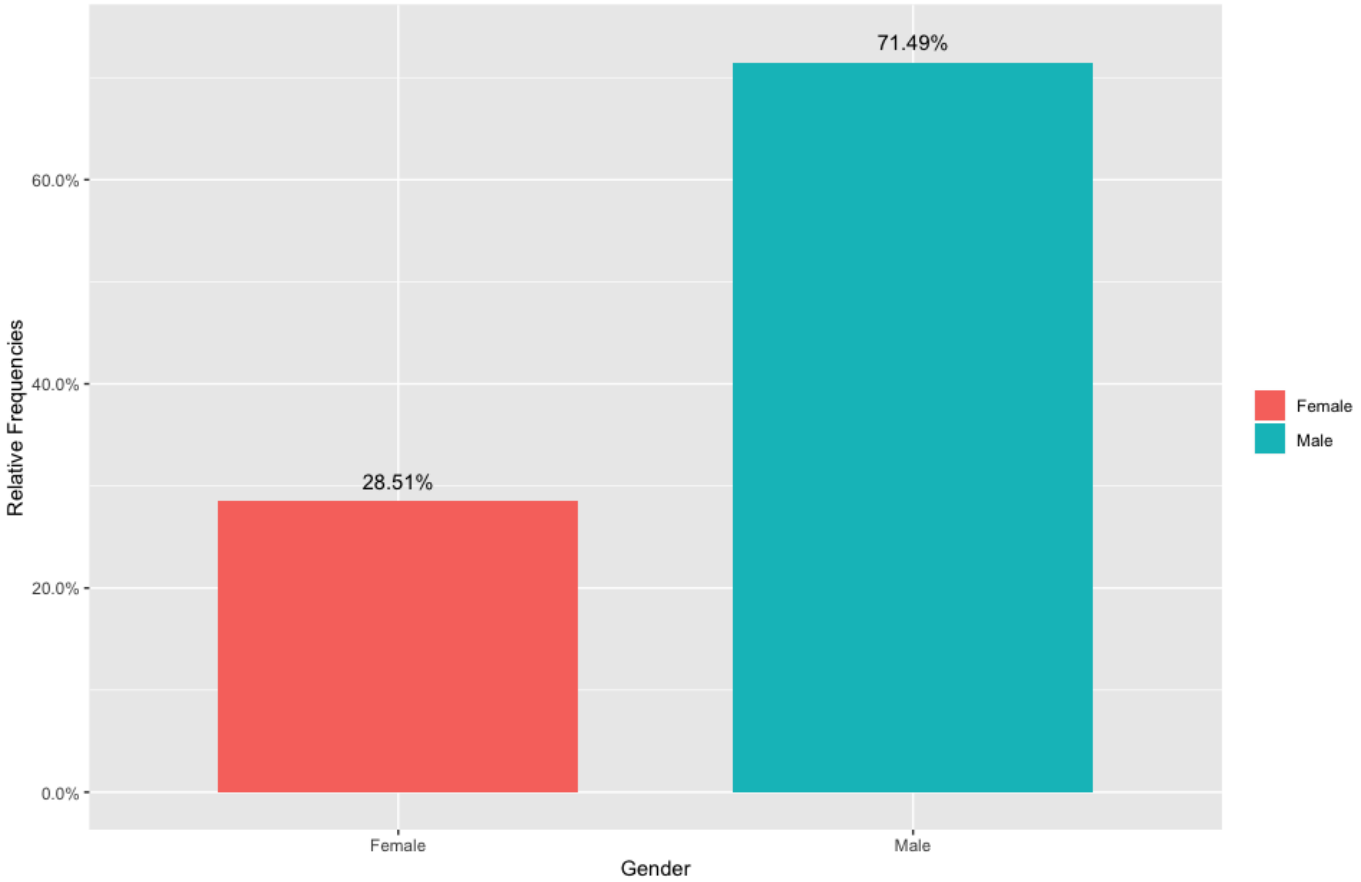
Hide

```
outcomes = c("Gender")
vars <- setdiff(colnames(trainAll), c(outcomes))

outcome <- "Gender"
pos <- "Female"

cal_idx <- rbinom(n = nrow(trainAll), size = 1, prob = 0.1) > 0
cal <- subset(trainAll, cal_idx)
train <- subset(trainAll, !cal_idx)
```

Before we begin any modelling tasks, it is valuable for us to discern what the majority class classifier (null model) is for our response variable in the data:

Hide

```
gndr.tab <- prop.table(table(df$Gender))
f <- round(gndr.tab[1], 4)
m <- round(gndr.tab[2], 4)

ggplot(df, aes(x = Gender, fill = Gender)) +
    geom_bar(aes(y = (..count..) / sum(..count..)), width = 0.7) +
    scale_y_continuous(labels = percent_format()) +
    annotate("text", x = "Female", y = f + 0.02, label = paste0(f * 100, "%")) +
    annotate("text", x = "Male", y = m + 0.02, label = paste0(m * 100, "%")) +
    ylab("Relative Frequencies") +
    theme(legend.title = element_blank())
```



We see here that over 7/10 of the fatalities in our data are male, meaning our null model performs at a classification accuracy of 71.5%. This could cause some issues moving forward, as the models we train will have access to much more data for males than females, potentially leading to imbalanced predictions.
A potential solution to this problem is a technique called Synthetic Oversampling, which generates a random set of minority class observations to shift the classifier learning bias towards minority class. In the article referenced below, this technique outperforms both simple undersampling and simple oversampling as ways to deal with inbalanced classifications.

https://www.analyticsvidhya.com/blog/2016/03/practical-guide-deal-imbalanced-classification-problems/ (https://www.analyticsvidhya.com/blog/2016/03/practical-guide-deal-imbalanced-classification-problems/)

We will only use synthetic oversampling on the training data, as we want our test and calibration data to reflect real observations that our model accuracies will be measured on.

To perform this synthetic oversampling, we use the package `ROSE`, which stands for Random Over Sampling Examples. This package requires that all variables be transformed to a factor prior to synthetic sample generation:

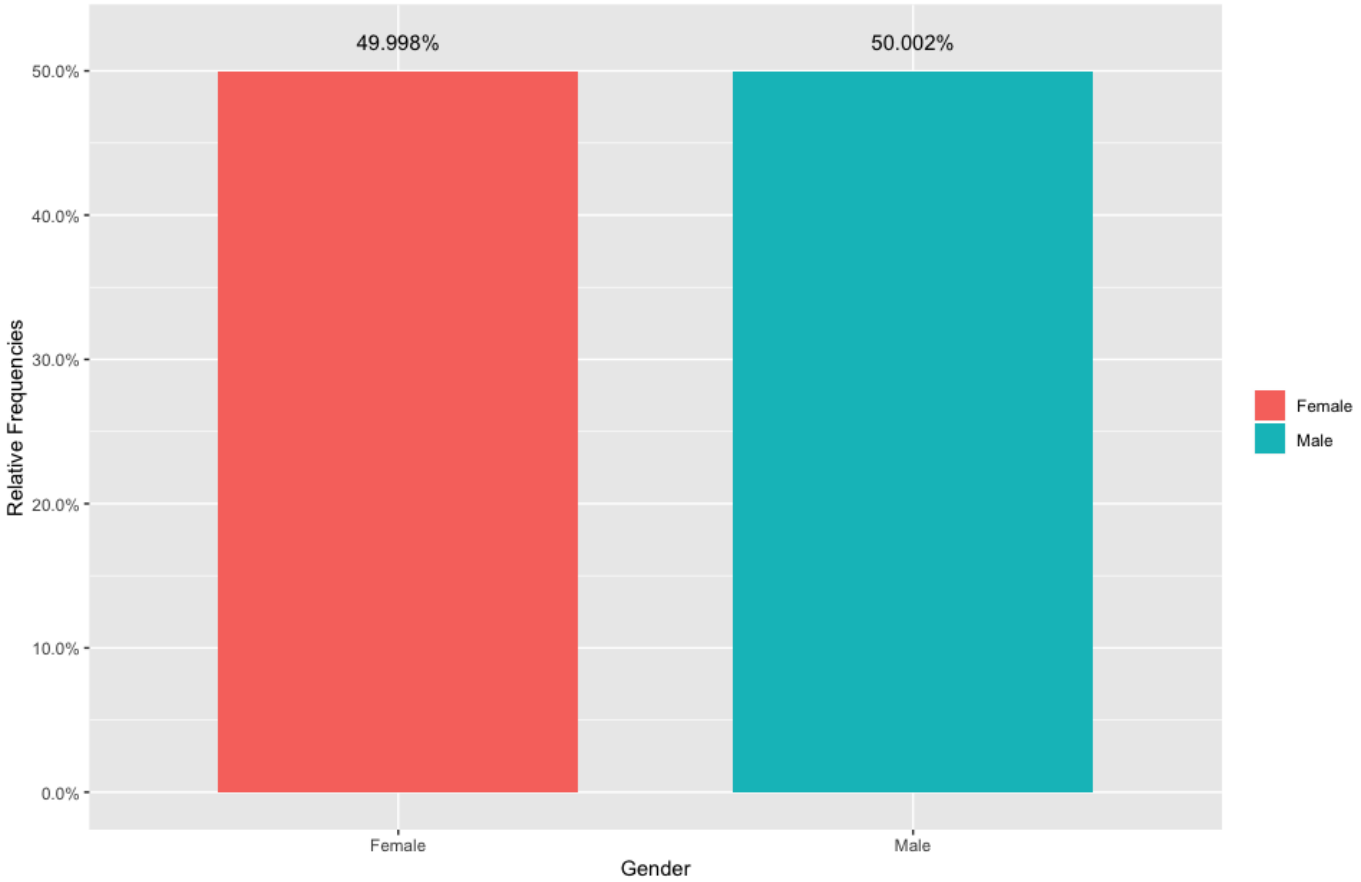
Hide

```
rose.form <- Gender ~ State + Month + Year + Dayweek + Time + Crash.Type +
    Bus.Involvement + A.Truck.Involvement + Speed.Limit + Road.User + Age +
    Christmas.Period + Easter.Period + Day.of.week + Time.of.day + Time.Period +
Age.Group

rose.train <- ROSE(rose.form, data = mutate_all(train, as.factor), seed = 4009)$d
ata
```

To check if this synthetic sampling worked:

Hide

```
rose.train$Gender <- relevel(rose.train$Gender, ref = "Female")
rose.gndr.tab <- prop.table(table(rose.train$Gender))
f <- round(rose.gndr.tab[1], 5)
m <- round(rose.gndr.tab[2], 5)

ggplot(rose.train, aes(x = Gender, fill = Gender)) +
    geom_bar(aes(y = (..count..) / sum(..count..)), width=0.7) +
    scale_y_continuous(labels = percent_format()) +
    annotate("text", x = "Female", y = f + 0.02, label = paste0(f * 100, "%")) +
    annotate("text", x = "Male", y = m + 0.02, label = paste0(m * 100, "%")) +
    ylab("Relative Frequencies") +
    theme(legend.title = element_blank())
```



Hide

```
rose.train$Gender <- relevel(rose.train$Gender, ref = "Male")
```

Now that we have a balanced data set for training, we'd like to convert all of our variables back to their correct class, which the following function achieves. We will also run our test and calibration data sets through this function to ensure the three datasets are in the exact same format.

Hide

```
numReverse <- function (df) {
    numV <- c("Month", "Year", "Time", "Speed.Limit", "Age")
    for (var in numV) {
        df[[var]] <- df[[var]] %>%
            as.character() %>%
            as.numeric
    }
    return(df)
}

rose.train <- numReverse(rose.train)
train <- numReverse(rose.train)
test <- numReverse(test)
cal <- numReverse(cal)
```

We can now create vectors of both categorical and numerical variables to be used in different single-variable modelling processes:

Hide

```
catVars <- vars[sapply(train[, vars], class) %in% c('factor', 'character')]
numVars <- vars[sapply(train[, vars], class) %in% c('numeric', 'integer')]
```

1.2 Single Variable Modelling

For our multi-variable modelling to be most effective, it can be valuable to begin with single-variable modelling to identify which variables on their own have a significant relationship with our response variable, `Gender`. We start with creating two functions, one for categorical variables and one for numeric variables. These functions take their respective variable, and create another column in our dataframe that is a numerical representation of the original variable, with a ‘pred’ prefix in the variable name.

Hide

```
mkPredC <- function(outCol, varCol, appCol) {
    pPos <- sum(outCol == pos) / length(outCol)
    naTab <- table(as.factor(outCol[is.na(varCol)]))
    pPosWna <- (naTab / sum(naTab))[pos]
    vTab <- table(as.factor(outCol), varCol)
    pPosWv <- (vTab[pos,] + 1.0e-3 * pPos) / (colSums(vTab) + 1.0e-3)
    pred <- pPosWv[appCol]
    pred[is.na(appCol)] <- pPosWna
    pred[is.na(pred)] <- pPos
    pred
}

mkPredN <- function(outCol, varCol, appCol) {
    cuts <- unique(as.numeric(quantile(varCol, probs = seq(0, 1, 0.1), na.rm = TR
UE)))
    varC <- cut(varCol,cuts)
    appC <- cut(appCol,cuts)
    mkPredC(outCol,varC,appC)
}
```

We also create an AUC function that calculates the area under the ROC curve for a given variable so we are able to compare across variables:

Hide

```
calcAUC <- function(predcol, outcol) {
    perf <- performance(prediction(predcol, outcol == pos), 'auc')
    as.numeric(perf@y.values)
}
```

We now use our predictions functions and our AUC function to cycle through all of our categorical and numerical variables, to identify if any score greater than a given threshold. If the AUC is above the threshold, we print both the training data AUC and the calibration data AUC for comparison. If the training and calibration AUC's are close, it means that our training data is representative of our full data set and is appropriate to be run against the test data. These functions highlight overfitting, where models can potentially "remember" too much from the training data and then fail to generalise, underperforming against the test data set.

Hide

```
thresh = 0.5

for(var in catVars) {
    pred <- paste0("pred", var)

    rose.train[, pred] <- mkPredC(rose.train[, outcome], rose.train[, var], rose.
train[, var])
    cal[, pred] <- mkPredC(rose.train[, outcome], rose.train[, var], cal[, var])
    test[, pred] <- mkPredC(rose.train[, outcome], rose.train[, var], test[, va
r])

    aucTrain <- calcAUC(rose.train[, pred], rose.train[, outcome])

    if(aucTrain >= thresh) {
        aucCal <- calcAUC(cal[, pred], cal[, outcome])
        print(sprintf("%s, trainAUC: %4.3f calibrationAUC: %4.3f", pred, aucTrai
n, aucCal))
}}
```

```
[1] "predState, trainAUC: 0.513 calibrationAUC: 0.493"
[1] "predDayweek, trainAUC: 0.533 calibrationAUC: 0.531"
[1] "predCrash.Type, trainAUC: 0.554 calibrationAUC: 0.551"
[1] "predBus.Involvement, trainAUC: 0.505 calibrationAUC: 0.503"
[1] "predA.Truck.Involvement, trainAUC: 0.508 calibrationAUC: 0.508"
[1] "predRoad.User, trainAUC: 0.674 calibrationAUC: 0.680"
[1] "predChristmas.Period, trainAUC: 0.502 calibrationAUC: 0.503"
[1] "predEaster.Period, trainAUC: 0.500 calibrationAUC: 0.500"
[1] "predAge.Group, trainAUC: 0.592 calibrationAUC: 0.593"
[1] "predDay.of.week, trainAUC: 0.534 calibrationAUC: 0.531"
[1] "predTime.of.day, trainAUC: 0.575 calibrationAUC: 0.579"
[1] "predTime.Period, trainAUC: 0.592 calibrationAUC: 0.596"
```

Hide

```
for(var in numVars) {
    pred <- paste0("pred", var)

    rose.train[, pred] <- mkPredN(rose.train[, outcome], rose.train[, var], rose.
train[, var])
    test[, pred] <- mkPredN(rose.train[, outcome], rose.train[, var], test[, va
r])
    cal[, pred] <- mkPredN(rose.train[, outcome], rose.train[, var], cal[, var])

    aucTrain <- calcAUC(rose.train[, pred], rose.train[, outcome])

    if(aucTrain >= thresh) {
        aucCal <- calcAUC(cal[, pred], cal[, outcome])
        print(sprintf("%s, trainAUC: %4.3f calibrationAUC: %4.3f", pred, aucTrai
n, aucCal))
}}
```

```
[1] "predMonth, trainAUC: 0.510 calibrationAUC: 0.493"
[1] "predYear, trainAUC: 0.522 calibrationAUC: 0.518"
[1] "predTime, trainAUC: 0.596 calibrationAUC: 0.597"
[1] "predSpeed.Limit, trainAUC: 0.514 calibrationAUC: 0.518"
[1] "predAge, trainAUC: 0.598 calibrationAUC: 0.608"
```

For our variables that have an AUC above the threshold, we see that the training AUC and calibration AUC are very similar, indicating that the training data is representative of our full data set.

Our next step to confirm that our training data is appropriate, is cross validation. This function, which is replicated 100 times, recalculates the training/calibration data split and then runs the prediction function again on a specific variable. The average (mean) of these 100 repetitions should be comparable to the AUC calculated for that variable above, which is especially important as the above calculation uses ROSE data with synthetic samples, whilst the cross-validation does not. We will take the variable `Road.User` as an example:

Hide

```
fCross <- function(var) {
    useForCalRep <- rbinom(n = nrow(trainAll), size = 1, prob = 0.1 ) > 0
    predRep <- mkPredC(trainAll[!useForCalRep, outcome],
                       trainAll[!useForCalRep, var],
                       trainAll[useForCalRep, var])
    calcAUC(predRep, trainAll[useForCalRep, outcome])
}

aucs <- replicate(100, fCross("Road.User"))

print(sprintf("cross-validation mean: %4.3f, cross-validation standard deviation:
%4.3f", mean(aucs), sd(aucs)))
```

```
[1] "cross-validation mean: 0.675, cross-validation standard deviation: 0.008"
```

We see that for the `Road User` variable, the cross-validation mean is almost identical to the above AUC calculation (0.675 vs 0.674 respectively), with a small standard deviation. Running this function with the other variables that met our threshold gives similar results, confirming that our training data is ready to be used in further modelling.

1.3 Logistic Regression

Our response variable is binary (male / female), which suits a binary logistic regression model. But before fitting a regression model, we want to remove variables that are too highly correlated to each other, to avoid feeding the model duplicate information (causing multicollinearity). Examples of this duplicated information can be found in the variables:

- Dayweek / Day.of.week
- Time.of.day / Time.Period / Time
- Age / Age.Group

We will take the variable `Dayweek` over `Day.of.week`, `Age` over `Age_Group`, and the variable `Time` over its counterparts, as these variables provide more detailed infromation about the time of the observation.

Hide

```
cat.remove <- c("Day.of.week", "Time.of.day", "Time.Period", "Age.Group")
catVars.rm <- catVars[! catVars %in% cat.remove]

fV <- paste0(outcome," ~ ", paste(c(catVars.rm, numVars), collapse = " + "))

tVars <- paste0('pred', c(catVars.rm, numVars))
fV2 <- paste0(outcome," ~ ", paste(tVars, collapse = " + "))
```

We can now fit our logistic regression model and calculate the AUC values for the model against on our

Hide

```
gender.log.regr <- glm(as.formula(fV2), data = rose.train, family = binomial(link
= 'logit'))

log.pred <- c(
    calcAUC(predict(gender.log.regr, newdata = rose.train), rose.train[, outcom
e]),
    calcAUC(predict(gender.log.regr, newdata = cal), cal[, outcome]),
    calcAUC(predict(gender.log.regr, newdata = test), test[, outcome]))

log.pred
```

```
[1] 0.7266602 0.7294604 0.7187536
```

Thankfully, we can see that our model performs similarly on the three datasets. We will now take a look at the summary of the model:

Hide

```
summary(gender.log.regr)
```

```
Call:
glm(formula = as.formula(fV2), family = binomial(link = "logit"),
    data = rose.train)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-2.059  -1.061  -0.296   1.034   2.422

Coefficients:
                         Estimate Std. Error z value Pr(>|z|)
(Intercept)             -20.17625    3.22879  -6.249 4.13e-10 ***
predState                 3.64605    0.85659   4.256 2.08e-05 ***
predDayweek               2.73797    0.36104   7.584 3.36e-14 ***
predCrash.Type            3.76958    0.21197  17.784  < 2e-16 ***
predBus.Involvement       0.39011    0.66666   0.585  0.55843
predA.Truck.Involvement   8.54320    0.80005  10.678  < 2e-16 ***
predRoad.User             4.87266    0.07794  62.520  < 2e-16 ***
predChristmas.Period      3.13945    2.02038   1.554  0.12021
predEaster.Period         2.28420    5.83582   0.391  0.69549
predMonth                 2.98044    1.20960   2.464  0.01374 *
predYear                  0.37291    0.56400   0.661  0.50850
predTime                  4.02249    0.13311  30.219  < 2e-16 ***
predSpeed.Limit           2.61153    0.79414   3.289  0.00101 **
predAge                   0.93080    0.13363   6.966 3.27e-12 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 56686  on 40889  degrees of freedom
Residual deviance: 49475  on 40876  degrees of freedom
AIC: 49503

Number of Fisher Scoring iterations: 4
```

We notice here that we have a number of variables that fail the Wald’s test, failing to reject the null hypothesis that the variable’s coefficient is equal to zero. We will now perform backwards elimination on the model, until we are left only with variables that have a significant relationship with our response:

Hide

```
log.regr.be <- update(gender.log.regr, .~.-predEaster.Period)
summary(log.regr.be)
```

```
Call:
glm(formula = Gender ~ predState + predDayweek + predCrash.Type +
    predBus.Involvement + predA.Truck.Involvement + predRoad.User +
    predChristmas.Period + predMonth + predYear + predTime +
    predSpeed.Limit + predAge, family = binomial(link = "logit"),
    data = rose.train)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-2.059  -1.061  -0.296   1.034   2.423

Coefficients:
                         Estimate Std. Error z value Pr(>|z|)
(Intercept)             -19.04484    1.43823 -13.242  < 2e-16 ***
predState                 3.64786    0.85657   4.259 2.06e-05 ***
predDayweek               2.73914    0.36102   7.587 3.27e-14 ***
predCrash.Type            3.76920    0.21196  17.782  < 2e-16 ***
predBus.Involvement       0.39114    0.66667   0.587  0.55740
predA.Truck.Involvement   8.53765    0.79991  10.673  < 2e-16 ***
predRoad.User             4.87244    0.07793  62.520  < 2e-16 ***
predChristmas.Period      3.14966    2.02020   1.559  0.11898
predMonth                 2.97831    1.20959   2.462  0.01381 *
predYear                  0.38926    0.56245   0.692  0.48888
predTime                  4.02275    0.13311  30.221  < 2e-16 ***
predSpeed.Limit           2.61021    0.79413   3.287  0.00101 **
predAge                   0.93085    0.13363   6.966 3.26e-12 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 56686  on 40889  degrees of freedom
Residual deviance: 49475  on 40877  degrees of freedom
AIC: 49501

Number of Fisher Scoring iterations: 4
```

Hide

```
log.regr.be <- update(log.regr.be, .~.-predBus.Involvement)
summary(log.regr.be)
```

```
Call:
glm(formula = Gender ~ predState + predDayweek + predCrash.Type +
    predA.Truck.Involvement + predRoad.User + predChristmas.Period +
    predMonth + predYear + predTime + predSpeed.Limit + predAge,
    family = binomial(link = "logit"), data = rose.train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.0603  -1.0608  -0.2959   1.0342   2.4227

Coefficients:
                         Estimate Std. Error z value Pr(>|z|)
(Intercept)             -18.85373    1.40082 -13.459  < 2e-16 ***
predState                 3.65193    0.85654   4.264 2.01e-05 ***
predDayweek               2.74340    0.36094   7.601 2.95e-14 ***
predCrash.Type            3.77871    0.21134  17.879  < 2e-16 ***
predA.Truck.Involvement   8.55336    0.79945  10.699  < 2e-16 ***
predRoad.User             4.87439    0.07787  62.596  < 2e-16 ***
predChristmas.Period      3.13165    2.02000   1.550  0.12106
predMonth                 2.97176    1.20952   2.457  0.01401 *
predYear                  0.38555    0.56240   0.686  0.49300
predTime                  4.02478    0.13307  30.246  < 2e-16 ***
predSpeed.Limit           2.61001    0.79414   3.287  0.00101 **
predAge                   0.93065    0.13363   6.965 3.29e-12 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 56686  on 40889  degrees of freedom
Residual deviance: 49475  on 40878  degrees of freedom
AIC: 49499

Number of Fisher Scoring iterations: 4
```

Hide

```
log.regr.be <- update(log.regr.be, .~.-predYear)
summary(log.regr.be)
```

```
Call:
glm(formula = Gender ~ predState + predDayweek + predCrash.Type +
    predA.Truck.Involvement + predRoad.User + predChristmas.Period +
    predMonth + predTime + predSpeed.Limit + predAge, family = binomial(link = "l
ogit"),
    data = rose.train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.0577  -1.0615  -0.2949   1.0347   2.4220

Coefficients:
                         Estimate Std. Error z value Pr(>|z|)
(Intercept)             -18.68370    1.37872 -13.552  < 2e-16 ***
predState                 3.66170    0.85642   4.276 1.91e-05 ***
predDayweek               2.73961    0.36089   7.591 3.17e-14 ***
predCrash.Type            3.78767    0.21094  17.956  < 2e-16 ***
predA.Truck.Involvement   8.56381    0.79935  10.713  < 2e-16 ***
predRoad.User             4.87938    0.07754  62.929  < 2e-16 ***
predChristmas.Period      3.13305    2.02021   1.551 0.120936
predMonth                 2.96844    1.20950   2.454 0.014117 *
predTime                  4.02504    0.13307  30.248  < 2e-16 ***
predSpeed.Limit           2.63138    0.79353   3.316 0.000913 ***
predAge                   0.92600    0.13345   6.939 3.96e-12 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 56686  on 40889  degrees of freedom
Residual deviance: 49476  on 40879  degrees of freedom
AIC: 49498

Number of Fisher Scoring iterations: 4
```

Hide

```
log.regr.be <- update(log.regr.be, .~.-predChristmas.Period)
summary(log.regr.be)
```

```
Call:
glm(formula = Gender ~ predState + predDayweek + predCrash.Type +
    predA.Truck.Involvement + predRoad.User + predMonth + predTime +
    predSpeed.Limit + predAge, family = binomial(link = "logit"),
    data = rose.train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.0593  -1.0617  -0.2952   1.0354   2.4216

Coefficients:
                         Estimate Std. Error z value Pr(>|z|)
(Intercept)             -17.19218    0.98717 -17.416  < 2e-16 ***
predState                 3.68146    0.85629   4.299 1.71e-05 ***
predDayweek               2.76290    0.36057   7.663 1.82e-14 ***
predCrash.Type            3.77705    0.21081  17.916  < 2e-16 ***
predA.Truck.Involvement   8.59506    0.79908  10.756  < 2e-16 ***
predRoad.User             4.88106    0.07753  62.954  < 2e-16 ***
predMonth                 3.04418    1.20853   2.519 0.011772 *
predTime                  4.02345    0.13306  30.239  < 2e-16 ***
predSpeed.Limit           2.64355    0.79341   3.332 0.000863 ***
predAge                   0.92415    0.13345   6.925 4.35e-12 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 56686  on 40889  degrees of freedom
Residual deviance: 49478  on 40880  degrees of freedom
AIC: 49498

Number of Fisher Scoring iterations: 4
```

Before settling on this model, we will perform a Drop In Deviance test between it and our unconstrained model (before backwards elimination). The null hypothesis of this test is that the additional explanatory variables in the unconstrained model do not explain more of the change in our response variable than the contrained model, ie. that the constrained model is adequate:

Hide

```
anova(log.regr.be, gender.log.regr, test = "Chisq")
```

```
Analysis of Deviance Table

Model 1: Gender ~ predState + predDayweek + predCrash.Type + predA.Truck.Involvem
ent +
    predRoad.User + predMonth + predTime + predSpeed.Limit +
    predAge
Model 2: Gender ~ predState + predDayweek + predCrash.Type + predBus.Involvement
+
    predA.Truck.Involvement + predRoad.User + predChristmas.Period +
    predEaster.Period + predMonth + predYear + predTime + predSpeed.Limit +
    predAge
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1     40880      49478
2     40876      49475  4   3.3779   0.4967
```

The p-value of 0.4967 leads us to fail to reject the null hypothesis, and we can conclude that the constrained model, obtained from backwards selection, is preferable.

A valuable metric to calculate the quality of our model is the pseudo R-squared, which will help us measure how much deviance is explained by our explanatory variables:

Hide

```
ps.r.sq <- log.regr.be$deviance / log.regr.be$null.deviance
sprintf("Pseudo R-Squared: %.4f", ps.r.sq)
```

```
[1] "Pseudo R-Squared: 0.8729"
```

This figure tells us that our model explains 87.29% of the variation in Gender.

1.4 Decision Trees

The second classification technique we will use decision trees. We will leverage the same formulas that were extablished in the logistic regression, with the first tree using the raw variables, the second tree using the 'pred' versions of these variables, the the third tree use the raw variables again but controlling for aspects of the tree such as depth and splits.

Hide

```
fV <- paste0(outcome," == 'Female' ~ ", paste(c(catVars.rm, numVars), collapse =
" + "))
rose.tmodel1 <- rpart(fV, data = rose.train)

Rose.Tree.1 <- c(
    calcAUC(predict(rose.tmodel1, newdata = rose.train), rose.train[, outcome]),
    calcAUC(predict(rose.tmodel1, newdata = cal), cal[, outcome]),
    calcAUC(predict(rose.tmodel1, newdata = test), test[, outcome]))
Rose.Tree.1
```

```
[1] 0.6887919 0.6973400 0.6912615
```

Hide

```
tVars <- paste0('pred', c(catVars.rm, numVars))
fV2 <- paste0(outcome," == 'Female' ~ ", paste(tVars, collapse = " + "))
rose.tmodel2 <- rpart(fV2, data = rose.train)

Rose.Tree.2 <- c(
    calcAUC(predict(rose.tmodel2, newdata = rose.train), rose.train[, outcome]),
    calcAUC(predict(rose.tmodel2, newdata = cal), cal[, outcome]),
    calcAUC(predict(rose.tmodel2, newdata = test), test[, outcome]))
Rose.Tree.2
```

```
[1] 0.6935812 0.7014220 0.6960501
```

Hide

```
rose.tmodel3 <- rpart(fV, data = rose.train,
                control = rpart.control(cp = 0.001, minsplit = 1000,
                                        minbucket = 1000, maxdepth = 5))

Rose.Tree.3 <- c(
    calcAUC(predict(rose.tmodel3, newdata = rose.train), rose.train[, outcome]),
    calcAUC(predict(rose.tmodel3, newdata = cal), cal[, outcome]),
    calcAUC(predict(rose.tmodel3, newdata = test), test[, outcome]))
Rose.Tree.3
```

```
[1] 0.7174499 0.7205072 0.7088934
```

Hide

```
dt.matrix <- matrix(c(Rose.Tree.1, Rose.Tree.2, Rose.Tree.3),
                    nrow = 3, ncol = 3, byrow = FALSE,
                    dimnames = list(c("Train", "Cal", "Test"),
                                    c("Tree.1", "Tree.2", "Tree.3")))
dt.matrix
```

```
        Tree.1    Tree.2    Tree.3
Train 0.6887919 0.6935812 0.7174499
Cal   0.6973400 0.7014220 0.7205072
Test  0.6912615 0.6960501 0.7088934
```

Looking at our three decision tree results, we see very comprable numbers within each tree for training, calibration and testing. This is likely due to our dilligence in the single-variable stage, as we ensured that the training data was fit for modelling. For the in depth analysis of the decision tree and the associated visual representation, we will use the third tree where we controlled for aspects as it gives us the best test results:

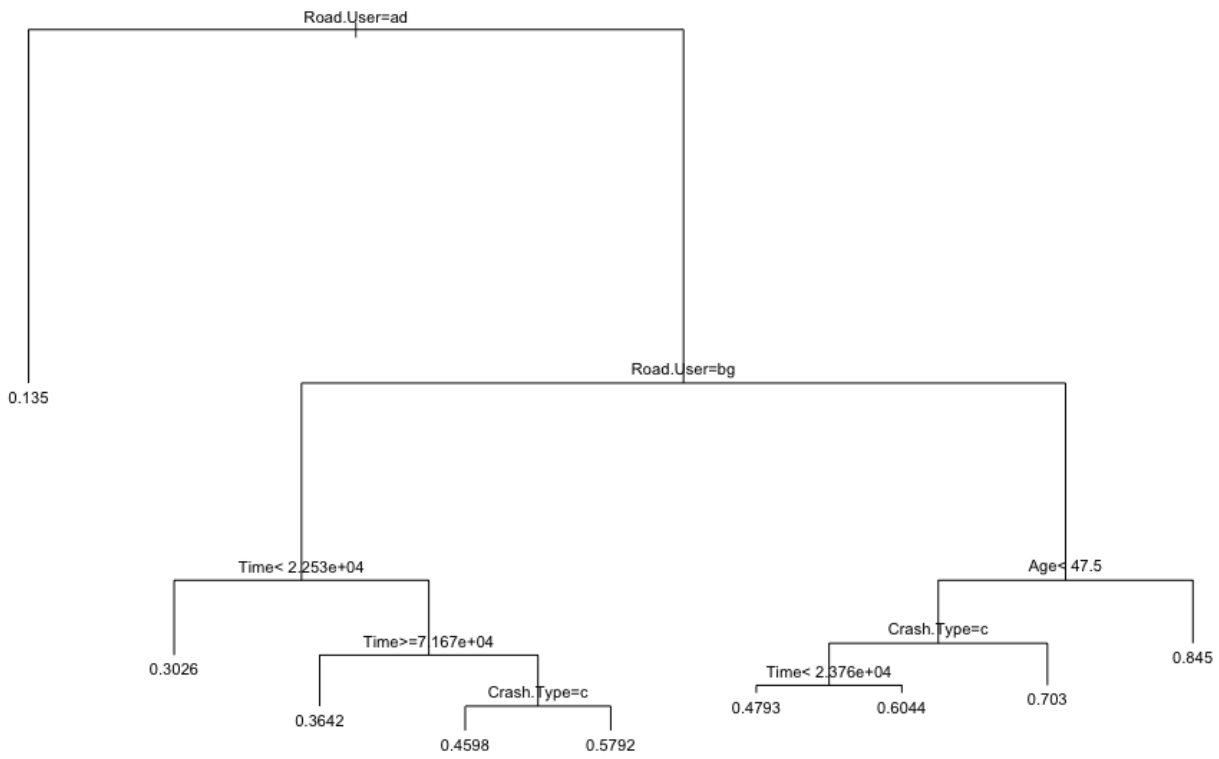
Hide

```
print(rose.tmodel3)
```

```
n= 40890

node), split, n, deviance, yval
      * denotes terminal node

 1) root 40890 10222.5000 0.4999755
   2) Road.User=Cyclist,MC. Rider 4673   545.7954 0.1350310 *
   3) Road.User=Driver,MC. Pass.,Other,Passenger,Pedestrian 36217  8974.0300 0.54
70635
     6) Road.User=Driver,Pedestrian 24470  6102.1940 0.4749898
      12) Time< 22530 4157   877.3014 0.3026221 *
      13) Time>=22530 20313  5076.1100 0.5102644
        26) Time>=71670 3850   891.4535 0.3641558 *
        27) Time< 71670 16463  4083.2470 0.5444330
          54) Crash.Type=Single 4796  1191.2330 0.4597581 *
          55) Crash.Type=Multiple,Pedestrian 11667  2843.4920 0.5792406 *
     7) Road.User=MC. Pass.,Other,Passenger 11747  2479.9380 0.6971993
      14) Age< 47.5 7895  1850.2340 0.6250792
        28) Crash.Type=Single 4488  1102.4780 0.5659537
          56) Time< 23760 1379   344.1610 0.4793328 *
          57) Time>=23760 3109   743.3805 0.6043744 *
        29) Crash.Type=Multiple,Pedestrian 3407   711.4001 0.7029645 *
      15) Age>=47.5 3852   504.4743 0.8450156 *
```

Hide

```
par(cex = 0.7)
plot(rose.tmodel3)
text(rose.tmodel3)
```



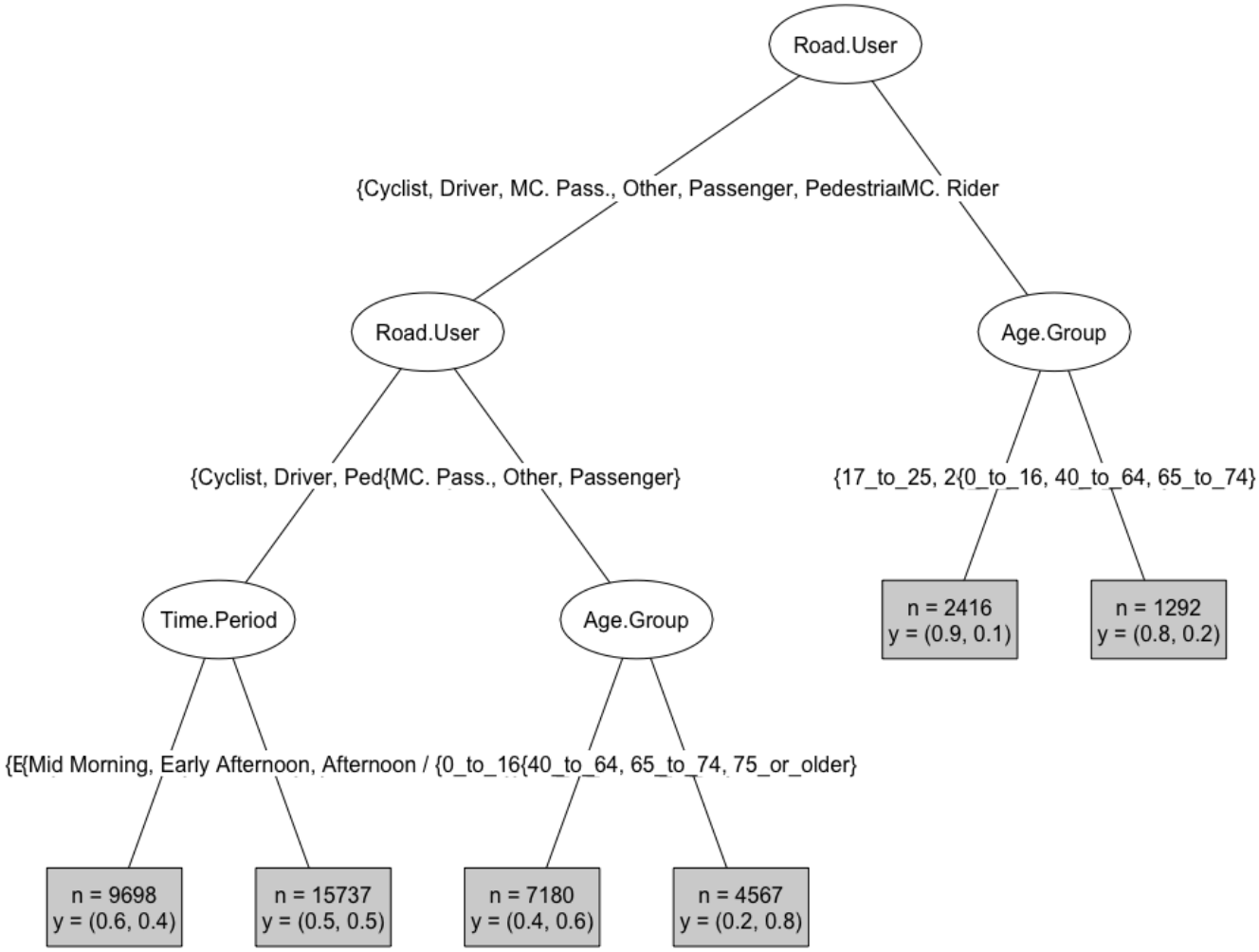
Looking at both the textual and visual display of our decision tree, we look to see where our ‘yval’ is substanitally above or below 0.5 as that is where the odds of being male are equivalent to being female for the respective node. Two points really stand out in this regard:

- The first is 0.135 - This node has a very low probability of the fatality being female where the ‘Road User’ is equal to Cyclist or Motorcycle Rider. Going back to the original data, we see that the number of Female & Cyclist is around 14% so that is in line, but the number of Female and Motocycle Rise is only 4%. This discrepancy shows that even though there are substantially more male motorcycle riders, our decision tree is predicting almost triple the number of female fatalities that we would expect.
- The second is 0.845 - This node on the far right has a very high probability of the fatality being female, where ‘Road User’ is equal to Motorcycle Passenger, Passenger or Other and the Age is greater or equal to 47.5. Again, going back to the original data, we see that only 68% of fatalities in the category are female. This reveals that at that node, the model is prediciting a much higher probability of being Female than our data would suggest.

Running a different decision tree package but with the same inputs as the third decision tree from above, we see similar variables playing a part in the model. Road User, Time Period and Age Group are all part of branches in our above tree visual. To note for the tree below though, it is resticted to only 3 levels deep due to the ability to read and interpret the visual semi-easily.

Hide

```
r.dt<- ctree(formula = rose.form, data = rose.train,
           controls = ctree_control(minsplit = 1000, minbucket = 1000, maxdepth =
3))

plot(r.dt, type = "simple",
     inner_panel = node_inner(r.dt, pval = FALSE, id = FALSE),
     terminal_panel = node_terminal(r.dt, digits = 1, id = FALSE))
```



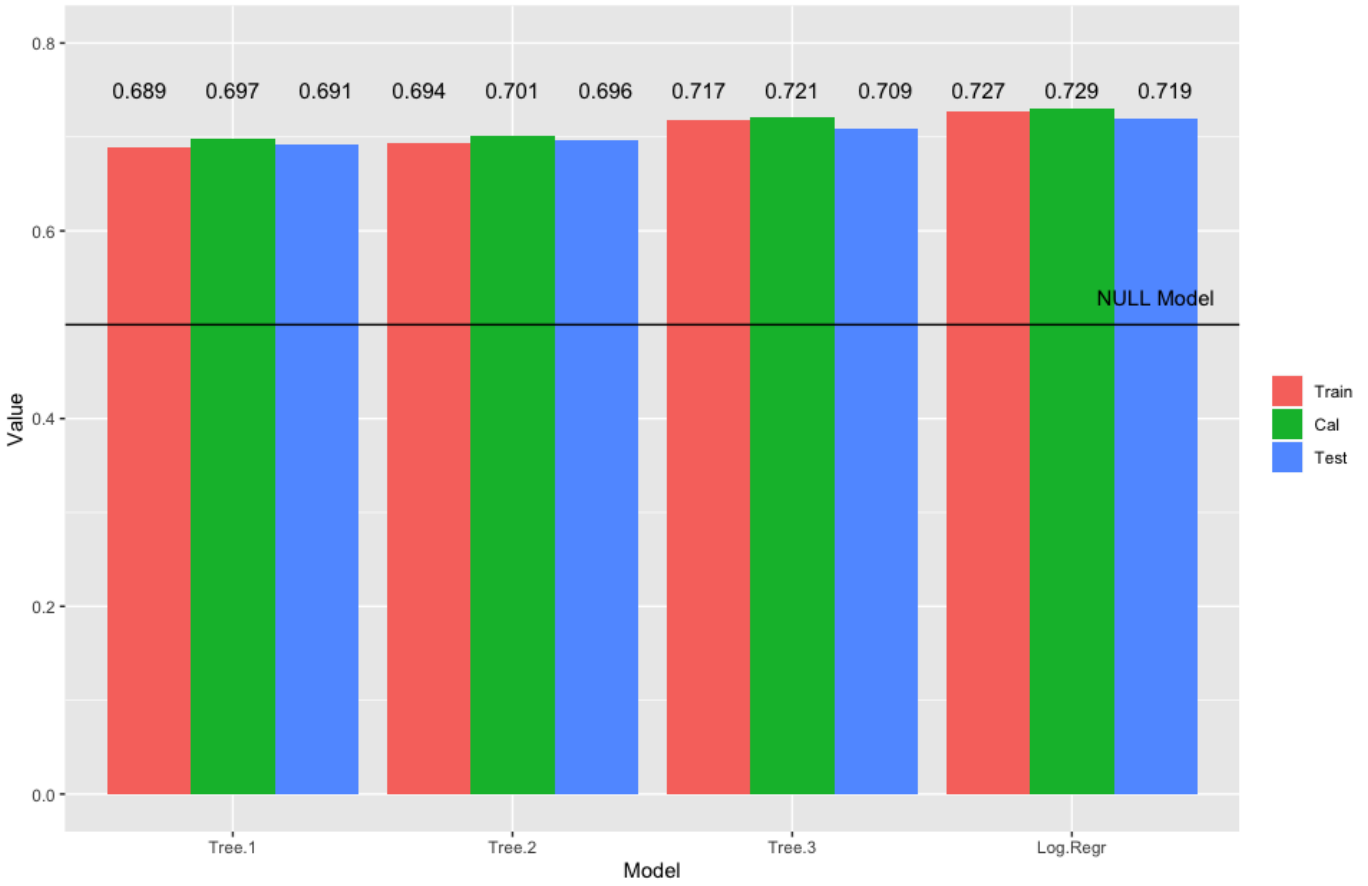
At each leaf of the tree, we see the total number of fatalities and the Male / Female split between them. What stands out here is that on the left of the tree the leaves are fairly gender balanced, but on the right, where Road.User = Motorcycle Rider, we see a significant skew towards the fatalities being male. We also see the inverse of this in the branch for Motorcycle Passenger, Passenger and Other, with the majority of the fatalities here being female. This modelling would infer that fatalities of riders of motorcycles are predominantly male, whilst passenger fatalities are more often female.

1.5 Classification Model Comparisons / Evaluation

Having modelled our logistic regression and our three decision trees, we create the visual to compare them against each other and the null model:

Hide

```
dtl.df <- data.frame(Model = rep(c("Tree.1", "Tree.2", "Tree.3", "Log.Regr"), ea
ch = 3),
                     Data = rep(c("Train", "Cal", "Test"), times = 4),
                     Value = c(Rose.Tree.1, Rose.Tree.2, Rose.Tree.3, log.pred))

ggplot(dtl.df, aes(x = factor(Model, levels = unique(Model)), y = Value,
                   fill = factor(Data, levels = unique(Data)))) +
    geom_bar(stat = "identity", position = "dodge") +
    geom_text(aes(y = 0.75, label = round(Value, digits = 3)), position = positio
n_dodge(width = 1), size = 4) +
    geom_hline(yintercept = 0.5, color = "black") +
    annotate(geom = "text", x = 4.3, y = 0.53, label = "NULL Model", colour = "b
lack") +
    ylim(0, 0.8) +
    ylab("Value") +
    xlab("Model") +
    theme(legend.title = element_blank())
```



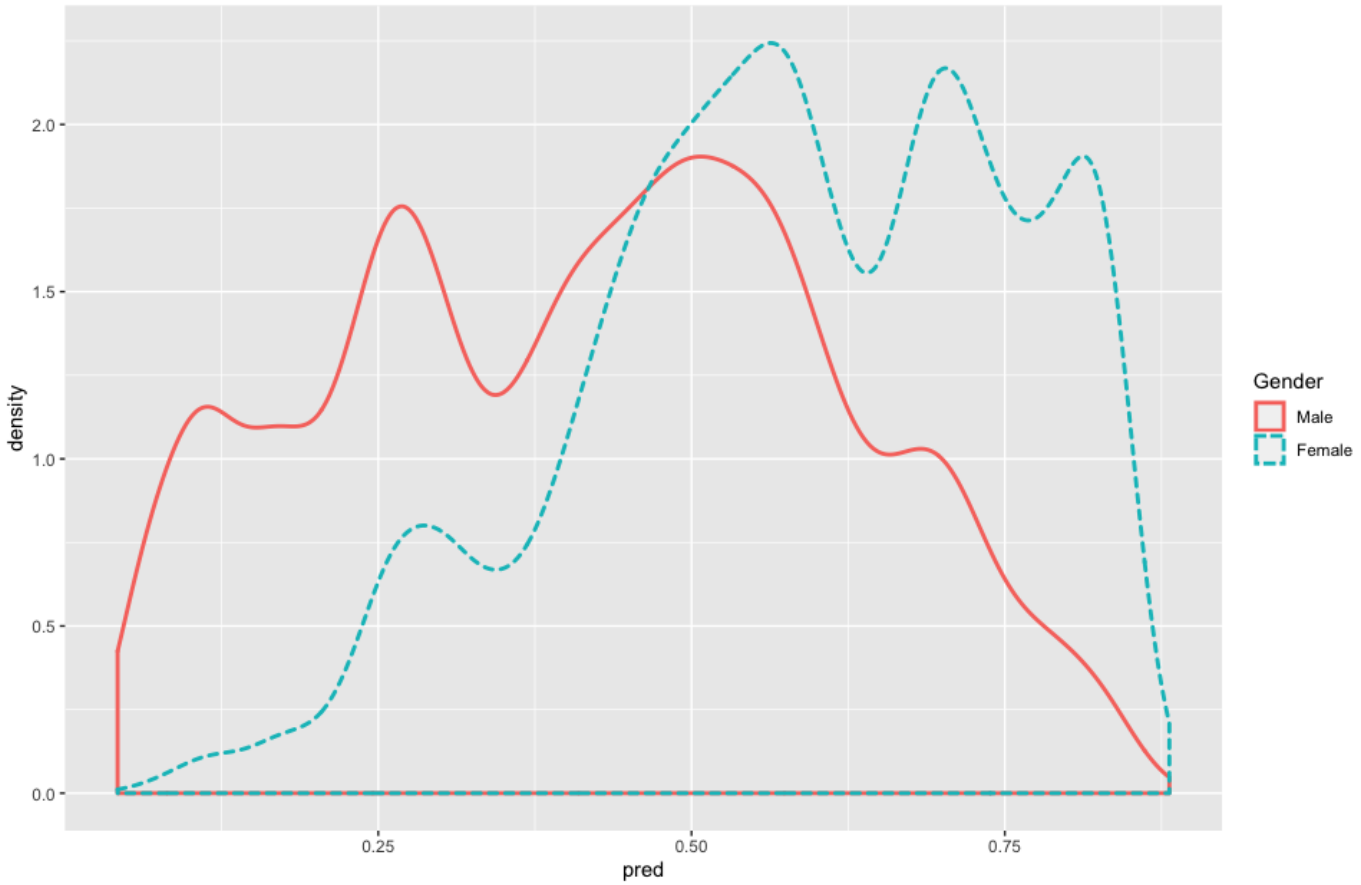
This plot shows that all of our models outperformed the null model, and that our logistic regression narrowly outperformed the decision tree models. Selecting the logistic regression as our preferred model, we will perform some additional analysis into it's prediction quality. Firstly, we can use the model to predict the gender (a value between 0 and 1) of fatalities in our dataset:

Hide

```
rose.train$pred <- predict(log.regr.be, newdata = rose.train, type = "response")
test$pred <- predict(log.regr.be, newdata = test, type = "response")

rose.train <- rose.train %>% mutate(Female = ifelse(Gender == "Female", TRUE, FAL
SE))
test <- test %>% mutate(Female = ifelse(Gender == "Female", TRUE, FALSE))
```

Hide

```
ggplot(rose.train, aes(x = pred, colour = Gender, linetype = Gender)) +
  geom_density(size = 1)
```



This plot shows the distribution of predictions when the true label is either Male or Female. Ideally, we'd like to see the two distributions completely separated, which isn't the case here, suggesting that our we cannot build a classifier from our model that simultaneously achieves good recall and good precision.

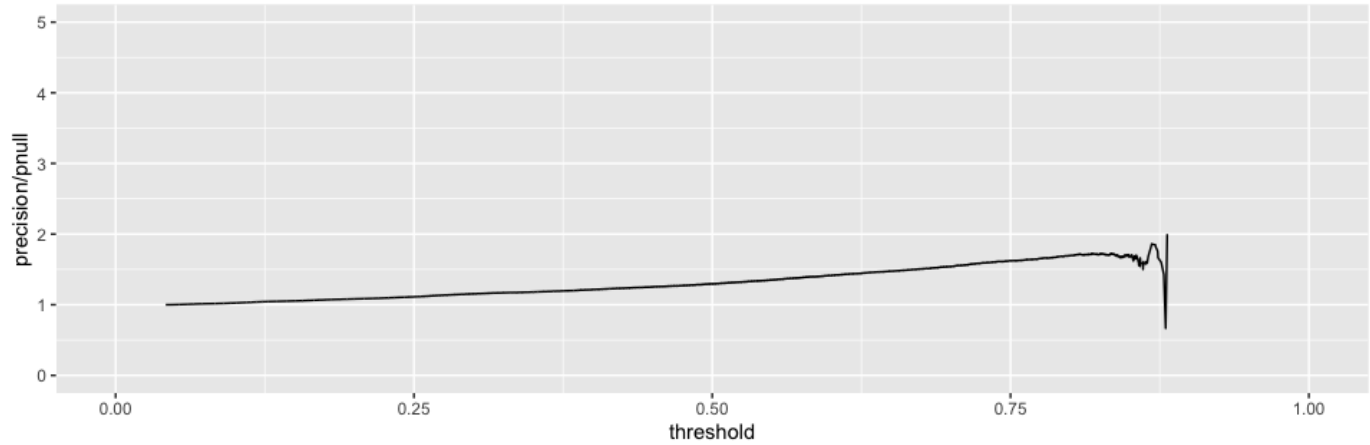
From a plot of our model's recall, F1 score and (precision / null model precision) over a range of thresholds, we aim to identify a comprimising classification threshold:

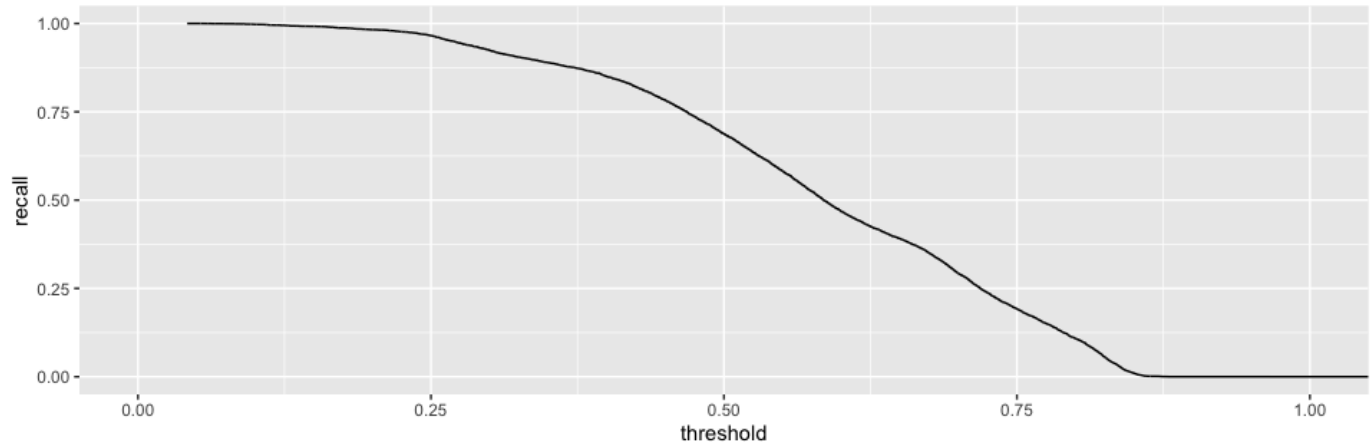
Hide

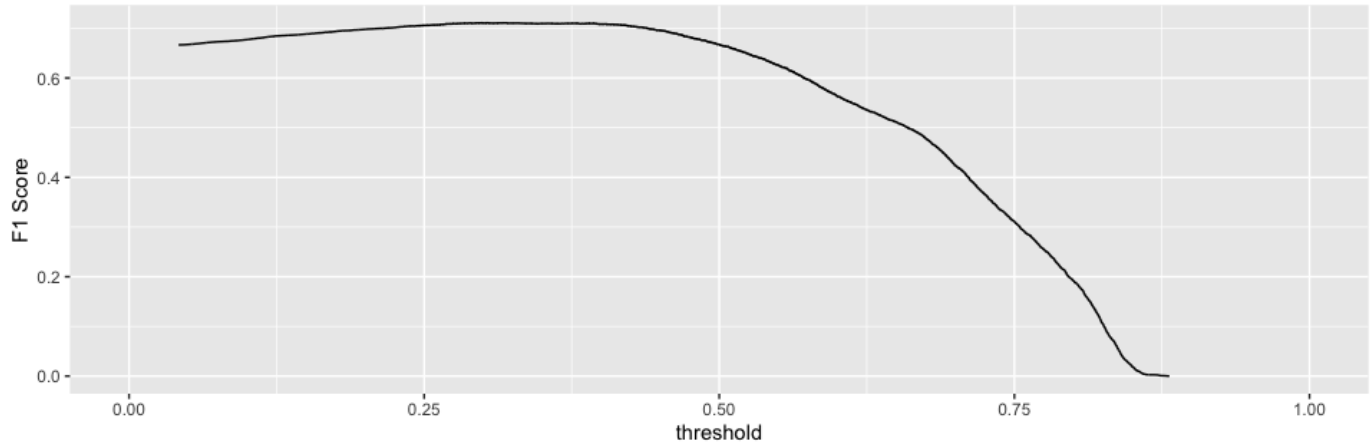
```
predObj <- prediction(rose.train$pred, rose.train$Female)
precObj <- performance(predObj, measure = "prec")
recObj <- performance(predObj, measure = "rec")
precision <- (precObj@y.values)[[1]]
prec.x <- (precObj@x.values)[[1]]
recall <- (recObj@y.values)[[1]]
rocFrame <- data.frame(threshold = prec.x,
                       precision = precision,
                       recall = recall)

pnull <- mean(as.numeric(rose.train$Female))
```

Hide

```
p <- ggplot(rocFrame, aes(x = threshold))
p1 <- p +
    geom_line(aes(y = precision / pnull)) +
    coord_cartesian(xlim = c(0, 1), ylim = c(0, 5))

p2 <- p +
    geom_line(aes(y = recall)) +
    coord_cartesian(xlim = c(0, 1))

p3 <- p + geom_line(aes(y = 2 * ((precision * recall) / (precision + recall)))) +
    ylab("F1 Score") +
    coord_cartesian(xlim = c(0, 1))

grid.arrange(p1, p2, p3, nrow = 3)
```







We can also calculate our precision, recall, enrichment rate and F1 score for our logistic regression model applied to our test data. We will look at these over a range of classification thresholds to help us select one for use:

Hide

```
thresh.eval <- function(df, thresh) {
    ctab <- table(pred = df$pred > thresh, Female = df$Female)

    precision <- ctab[2, 2] / sum(ctab[2, ])
    recall <- ctab[2, 2] / sum(ctab[, 2])
    enrich <- precision / mean(as.numeric(df$Female))

    F1 <- 2 * ((precision * recall) / (precision + recall))

    sprintf("Thresh: %4.2f, F1: %4.4f, Precision: %4.4f, Recall: %4.4f, Enrich: %
4.4f",
            thresh, F1, precision, recall, enrich)
}
```

Hide

```
for (i in seq(0.05, 0.85, by = 0.05)) {
  print(thresh.eval(test, i))
}
```

```
[1] "Thresh: 0.05, F1: 0.4505, Precision: 0.2907, Recall: 1.0000, Enrich: 1.0024"
[1] "Thresh: 0.10, F1: 0.4617, Precision: 0.3005, Recall: 0.9959, Enrich: 1.0361"
[1] "Thresh: 0.15, F1: 0.4765, Precision: 0.3135, Recall: 0.9918, Enrich: 1.0811"
[1] "Thresh: 0.20, F1: 0.4911, Precision: 0.3269, Recall: 0.9863, Enrich: 1.1272"
[1] "Thresh: 0.25, F1: 0.5072, Precision: 0.3437, Recall: 0.9679, Enrich: 1.1850"
[1] "Thresh: 0.30, F1: 0.5127, Precision: 0.3554, Recall: 0.9194, Enrich: 1.2255"
[1] "Thresh: 0.35, F1: 0.5213, Precision: 0.3683, Recall: 0.8914, Enrich: 1.2700"
[1] "Thresh: 0.40, F1: 0.5279, Precision: 0.3824, Recall: 0.8525, Enrich: 1.3184"
[1] "Thresh: 0.45, F1: 0.5305, Precision: 0.4013, Recall: 0.7821, Enrich: 1.3838"
[1] "Thresh: 0.50, F1: 0.5201, Precision: 0.4176, Recall: 0.6892, Enrich: 1.4400"
[1] "Thresh: 0.55, F1: 0.4978, Precision: 0.4432, Recall: 0.5676, Enrich: 1.5282"
[1] "Thresh: 0.60, F1: 0.4735, Precision: 0.4843, Recall: 0.4631, Enrich: 1.6699"
[1] "Thresh: 0.65, F1: 0.4406, Precision: 0.5220, Recall: 0.3811, Enrich: 1.7998"
[1] "Thresh: 0.70, F1: 0.3873, Precision: 0.5609, Recall: 0.2958, Enrich: 1.9340"
[1] "Thresh: 0.75, F1: 0.2975, Precision: 0.6305, Recall: 0.1947, Enrich: 2.1741"
[1] "Thresh: 0.80, F1: 0.1695, Precision: 0.6698, Recall: 0.0970, Enrich: 2.3096"
[1] "Thresh: 0.85, F1: 0.0122, Precision: 0.7500, Recall: 0.0061, Enrich: 2.5861"
```

Here, we might select 0.45 as the threshold, as it has the maximum F1 score on our test data. The F1 score is a measure of our model's accuracy that takes into account both precision and recall.

---

2. Clustering

For our clustering analysis, we have a new task: we want to explore relationships between differently aged road users of differing types, according to the speed and time of the accident. To accomplish this, and considering our data is primarily categorical, we need to create a new dataset from our original data, with numerical representations of the variables of interest. We will create a new dataframe, with each observation being a different age group and road user type (for example: "Middle Aged Motorcycle Rider"), and variables decribing the speed limit and time of the crash.

2.1 Clustering Transformations

The clustering transformations begin with the same dataframe that was used for classification, without synthetic oversampling. We then join with the crashes data set to get the number of fatalities that were involved in the crash to be associated with each of our individual fatality oberservations.

Hide

```
crashes.join <- crashes %>%
  select(c("Crash ID", "Number Fatalities"))

colnames(crashes.join) <- str_replace_all(colnames(crashes.join), c(" " = "."))
crashes.join$Crash.ID <- as.factor(crashes.join$Crash.ID)

c.df <- df %>%
    left_join(crashes.join, by = "Crash.ID") %>%
    as.data.frame() %>%
    mutate(Month = as.numeric(Month), Year = as.numeric(Year),
           Age = as.numeric(Age), Speed.Limit = as.numeric(Speed.Limit)) %>%
    mutate_if(is.character, as.factor)

c.df$Road.User <- revalue(c.df$Road.User, c("Other/-9" = "Other"))
```

We then provide more meaningful names to our Age.Group variable:

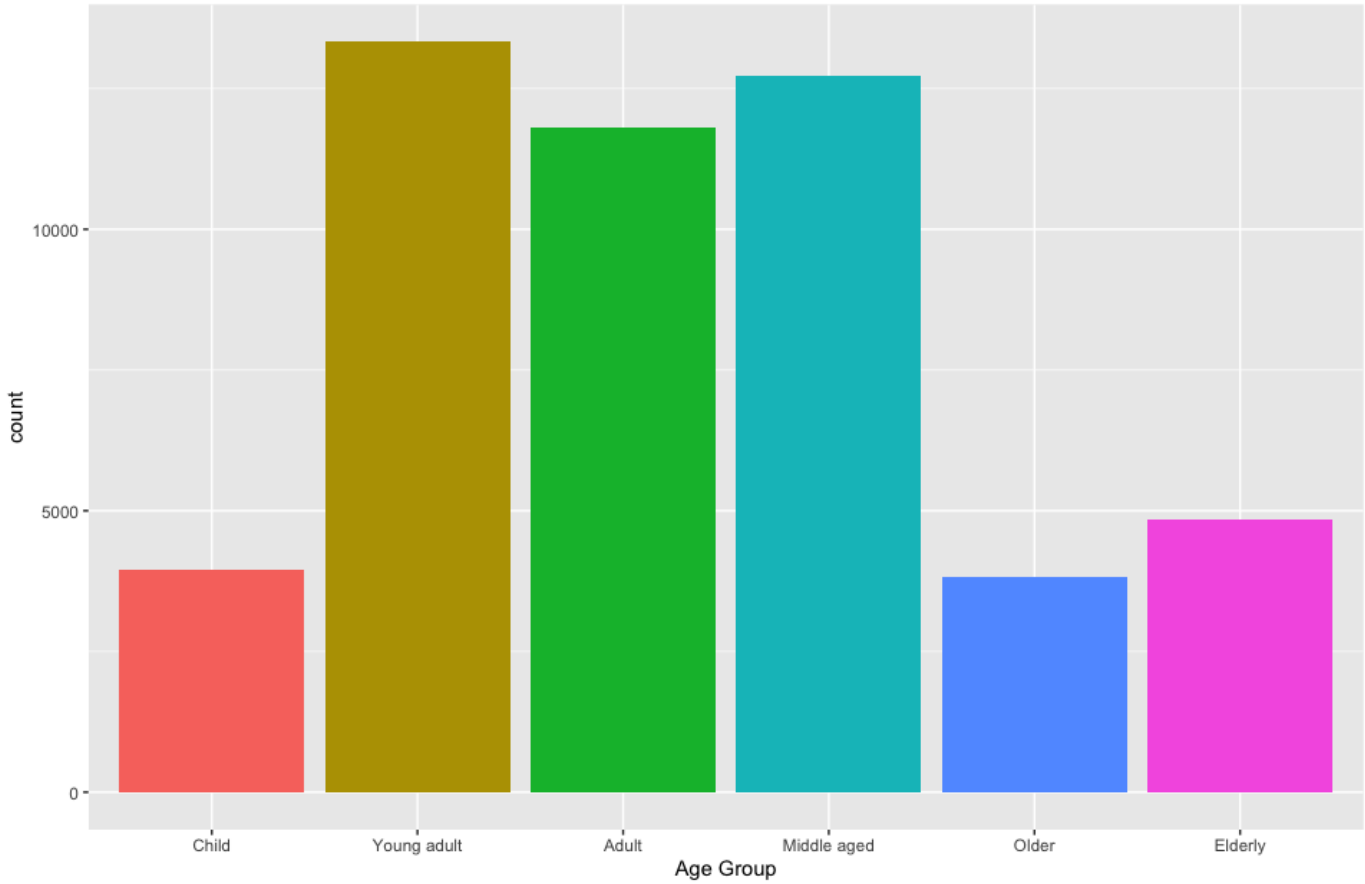
Hide

```
c.df <- within(c.df, {
  Age.Grp.Name <- NA
  Age.Grp.Name[Age.Group == "0_to_16"] <- "Child"
  Age.Grp.Name[Age.Group == "17_to_25"] <- "Young adult"
  Age.Grp.Name[Age.Group == "26_to_39"] <- "Adult"
  Age.Grp.Name[Age.Group == "40_to_64"] <- "Middle aged"
  Age.Grp.Name[Age.Group == "65_to_74"] <- "Older"
  Age.Grp.Name[Age.Group == "75_or_older"] <- "Elderly"
})

c.df$Age.Grp.Name <- factor(c.df$Age.Grp.Name,
                            levels = c("Child", "Young adult",
                                       "Adult", "Middle aged",
                                       "Older", "Elderly"))
```

and look at the distribution of fatal crashes between these age groups:

Hide

```
c.df %>%
    ggplot(aes(x = Age.Grp.Name, fill = Age.Grp.Name)) +
        geom_bar() +
        xlab("Age Group") +
        theme(legend.position = "")
```



Now we will create our new dataframe, where each observation is a different age group and road user. We also add a summary column containing the average `Speed.Limit` a fatal crash occurs in in each group, and a column for each `Time.Period` during the day, with each value being the percentage of crashes in that group occurring in that time period:

Hide

```
c.df <- c.df %>%
    mutate(Road.User.Age = factor(paste0(Age.Grp.Name, ".", Road.User)))

clust.df <- c.df %>%
    group_by(Road.User.Age) %>%
    filter(Speed.Limit != "Missing") %>%
    dplyr::summarise(avgSpeed = round(mean(as.numeric(Speed.Limit))))
```

Hide

```
n <- length(levels(c.df$Road.User.Age))
t <- round(prop.table(with(c.df, table(Road.User.Age, Time.Period)), 1), 4) * 100
split.tab <- split(t, ceiling(seq_along(t)/n))

clust.df$percEarlyMorn <- split.tab$`1`
clust.df$percMidMorn <- split.tab$`2`
clust.df$percEarlyAft <- split.tab$`3`
clust.df$percAftEve <- split.tab$`4`
clust.df$percNight <- split.tab$`5`
clust.df$percLateNight <- split.tab$`6`

head(clust.df, 10)
```

Road.User.Age <fctr>	avgSp... <dbl>	percEarlyMorn <dbl>	percMidM... <dbl>	percEarlyAft <dbl>	percAftEve <dbl>	p
Adult.Cyclist	74	26.41	13.85	19.48	22.94	
Adult.Driver	90	14.32	12.19	16.07	20.83	
Adult.MC. Pass.	81	4.65	11.63	24.42	29.07	
Adult.MC. Rider	76	10.51	13.86	23.73	28.58	
Adult.Other	93	12.50	6.25	31.25	25.00	
Adult.Passenger	90	10.84	11.57	18.13	22.46	
Adult.Pedestrian	74	11.57	5.90	6.19	21.11	
Child.Cyclist	71	6.13	16.87	25.15	46.63	
Child.Driver	86	12.90	10.60	19.35	15.67	
Child.MC. Pass.	77	2.50	7.50	22.50	45.00	

1-10 of 10 rows

2.2 Finding k

We are using the euclidean distance for our clustering as it is the straight line distance between our points that we are interested in:

Hide

```
vars.to.use <- colnames(clust.df)[-1]
pmatrix <- scale(clust.df[, vars.to.use])

pcenter <- attr(pmatrix, "scaled:center")
pscale <- attr(pmatrix, "scaled:scale")

d <- dist(pmatrix, method = "euclidean")
```

Before performing our clustering, we need to find the optimal value for `k` to use. `k` will be the number of clusters that we model against, and it can be derived through the within sum of squares (the average distance from each point to the centre of its cluster) and the between sum of squares (the average distance between clusters):

Hide

```
sqr_edist <- function(x, y) {
  sum((x - y) ^ 2)
}

wss.cluster <- function(clustermat) {
  c0 <- apply(clustermat, 2, FUN = mean)
  sum(apply(clustermat, 1,
            FUN = function(row) {sqr_edist(row, c0)}))
}

wss.total <- function(dmatrix, labels) {
  wsstot <- 0
  k <- length(unique(labels))
  for(i in 1:k){
    wsstot <- wsstot +
      wss.cluster(subset(dmatrix, labels == i))
  }
  wsstot
}

totss <- function(dmatrix) {
  grandmean <- apply(dmatrix, 2, FUN = mean)
  sum(apply(dmatrix, 1,
            FUN=function(row) {
              sqr_edist(row, grandmean)
              }
            )
      )
}
```

The Calinski-Harabasz (CH) index is a popular metric for selecting the optimum value for `k` . The CH index is defined as the ratio of the between-cluster variance to the total within-cluster variance. The following function computes this CH index:

Hide

```
rm(var)
ch_criterion <- function(dmatrix, kmax, method = "kmeans") {
  if(!(method %in% c("kmeans", "hclust"))){
    stop("method must be one of c('kmeans', 'hclust')")
  }
  npts <- nrow(dmatrix)
  totss <- totss(dmatrix)
  wss <- numeric(kmax)
  crit <- numeric(kmax)
  wss[1] <- (npts - 1) * sum(apply(dmatrix, 2, FUN = var))

  for(k in 2:kmax) {
    if(method == "kmeans") {
      clustering <- kmeans(dmatrix, k, nstart = 10, iter.max = 100)
      wss[k] <- clustering$tot.withinss
    }else {
      d <- dist(dmatrix, method = "euclidean")
      pfit <- hclust(d, method = "ward")
      labels <- cutree(pfit, k = k)
      wss[k] <- wss.total(dmatrix, labels)
    }
  }
  bss <- totss - wss
  crit.num <- bss / (0:(kmax - 1))
  crit.denom <- wss/(npts - 1:kmax)
  list(crit = crit.num / crit.denom, wss = wss, bss = bss, totss = totss)
}
```

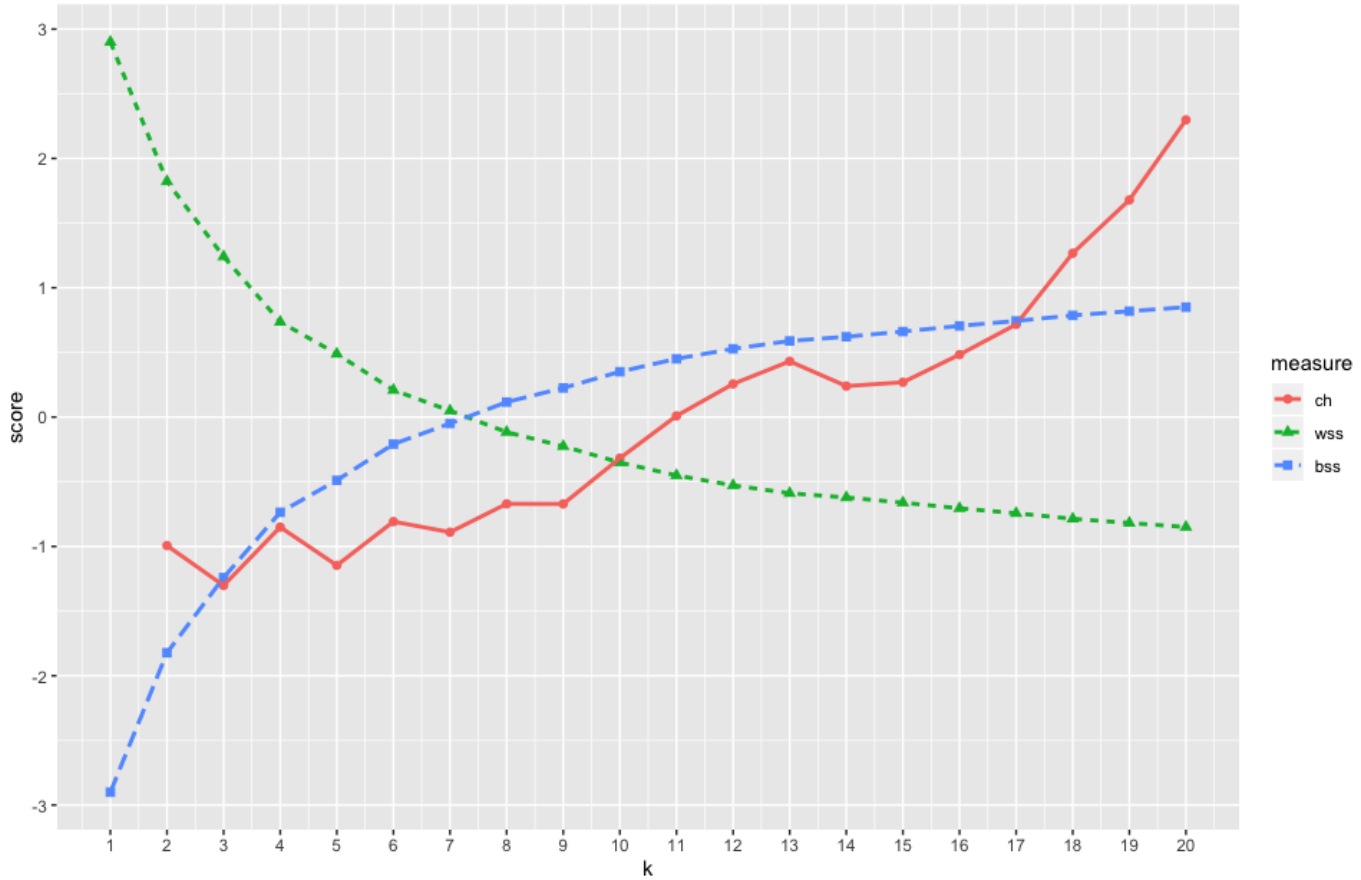
Typically, the CH index is then plotted over a range of different values for `k`, and if a clear “elbow” is present (where the benefit of an increased number of clusters slows down), that value is selected as `k`.

Hide

```
clustcrit <- ch_criterion(pmatrix, 20, method = "hclust")
critframe <- data.frame(k = 1:20, ch = scale(clustcrit$crit), wss = scale(clustcr
it$wss), bss = scale(clustcrit$bss))
critframe <- melt(critframe, id.vars = c("k"), variable.name = "measure", value.n
ame = "score")
```

Hide

```
ggplot(critframe, aes(x = k, y = score, color = measure)) +
    geom_point(aes(shape = measure), size = 2) +
    geom_line(aes(linetype = measure), size = 1) +
    scale_x_continuous(breaks = 1:20, labels = 1:20)
```



As our CH index simply increases, it is difficult to interpret from it a good value for `k`. We will instead make use of an `R` function named `NbClust`, which finds the optimum value for `k` as reported from over 30 different clustering indexes, then tallies these results:

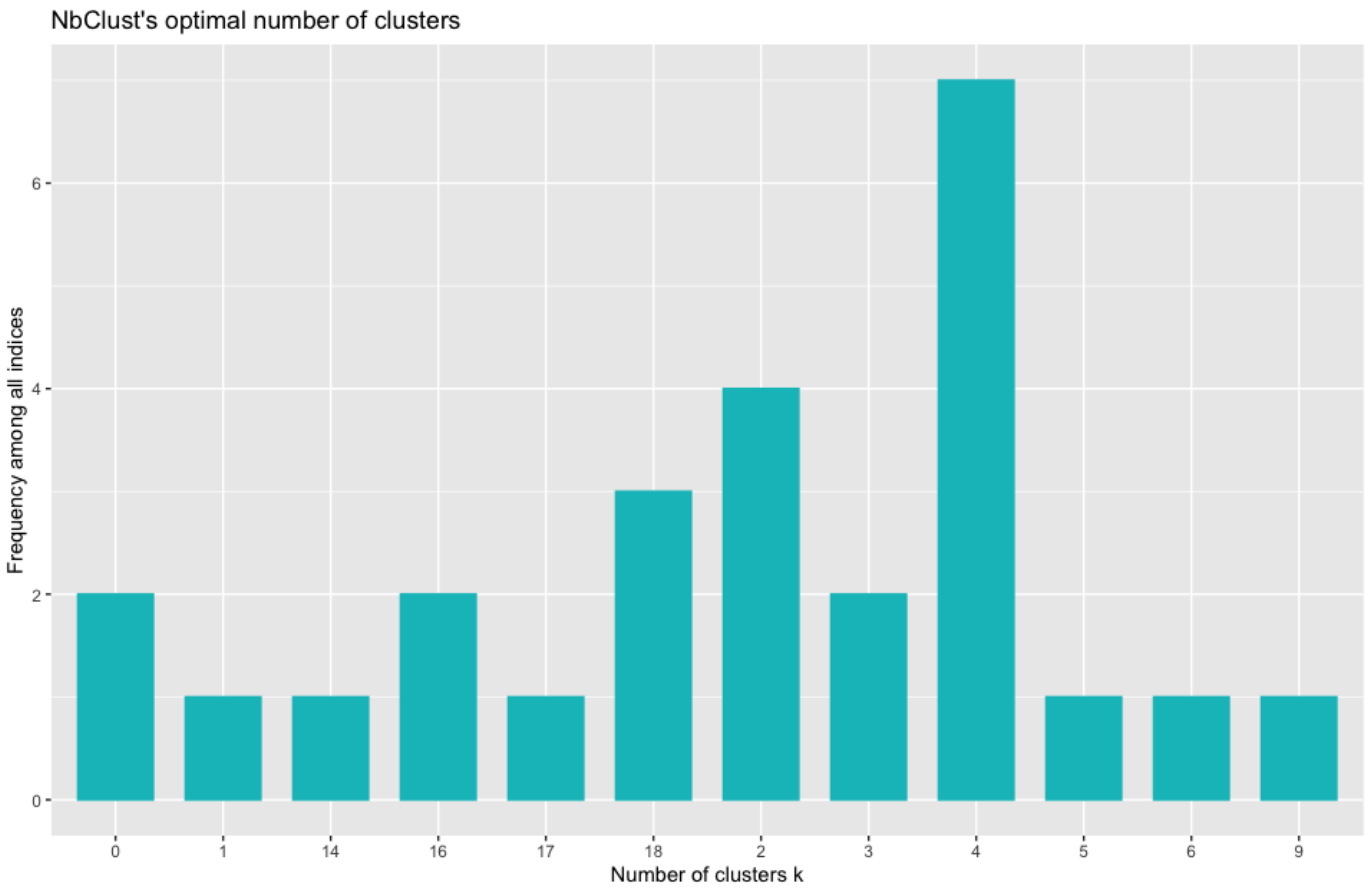
Hide

```
library(NbClust)
library(factoextra)
k.select <- clust.df %>%
    select(-Road.User.Age) %>%
    NbClust(method = "ward.D2", index = "all", min.nc = 2, max.nc = 18)
```

Hide

```
fviz_nbclust(k.select, barfill = "#00BFC4", barcolor = "#00BFC4") +
    theme_gray() +
    ggtitle("NbClust's optimal number of clusters")
```

```
Among all indices:
===================
* 2 proposed  0 as the best number of clusters
* 1 proposed  1 as the best number of clusters
* 4 proposed  2 as the best number of clusters
* 2 proposed  3 as the best number of clusters
* 7 proposed  4 as the best number of clusters
* 1 proposed  5 as the best number of clusters
* 1 proposed  6 as the best number of clusters
* 1 proposed  9 as the best number of clusters
* 1 proposed  14 as the best number of clusters
* 2 proposed  16 as the best number of clusters
* 1 proposed  17 as the best number of clusters
* 3 proposed  18 as the best number of clusters

Conclusion
=========================
* According to the majority rule, the best number of clusters is  4 .
```



From this, we can see that the majority of clustering indexes selected 4 as the optimum value for `k`. Using this value, we use the clusterboot function to see the stability of these clusters:

Hide

```
kbest.p <- 4
cboot.hclust <- clusterboot(pmatrix,
                            clustermethod = hclustCBI,
                            method = "ward.D",
                            k = kbest.p,
                            count=FALSE)

groups <- cboot.hclust$result$partition

1 - cboot.hclust$bootbrd / 100
```

```
[1] 0.67 0.88 0.70 0.90
```

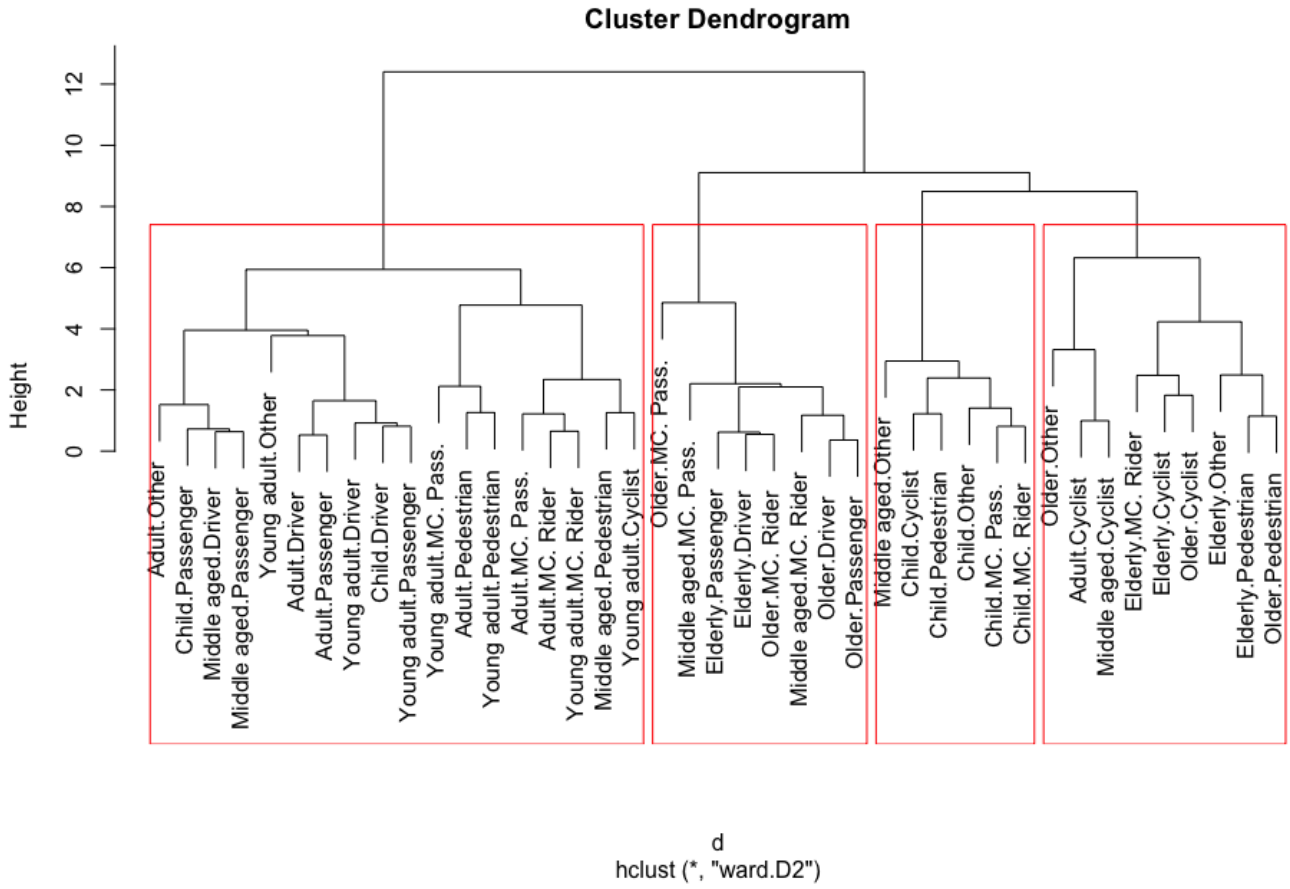
Using a value of roughly 0.70 to represent stability, we see that each cluster is roughly stable. This confirms 4 as our value for `k`.

2.3 Hierarchical Clustering

The clustering methodology we will use is hierarchical clustering. The below dendrogram displays each of our 42 'Road User Age' groups in their respective cluster:

Hide

```
pfit <- hclust(d, method="ward.D2")
plot(pfit, labels = clust.df$Road.User.Age)
rect.hclust(pfit, k = kbest.p)
```



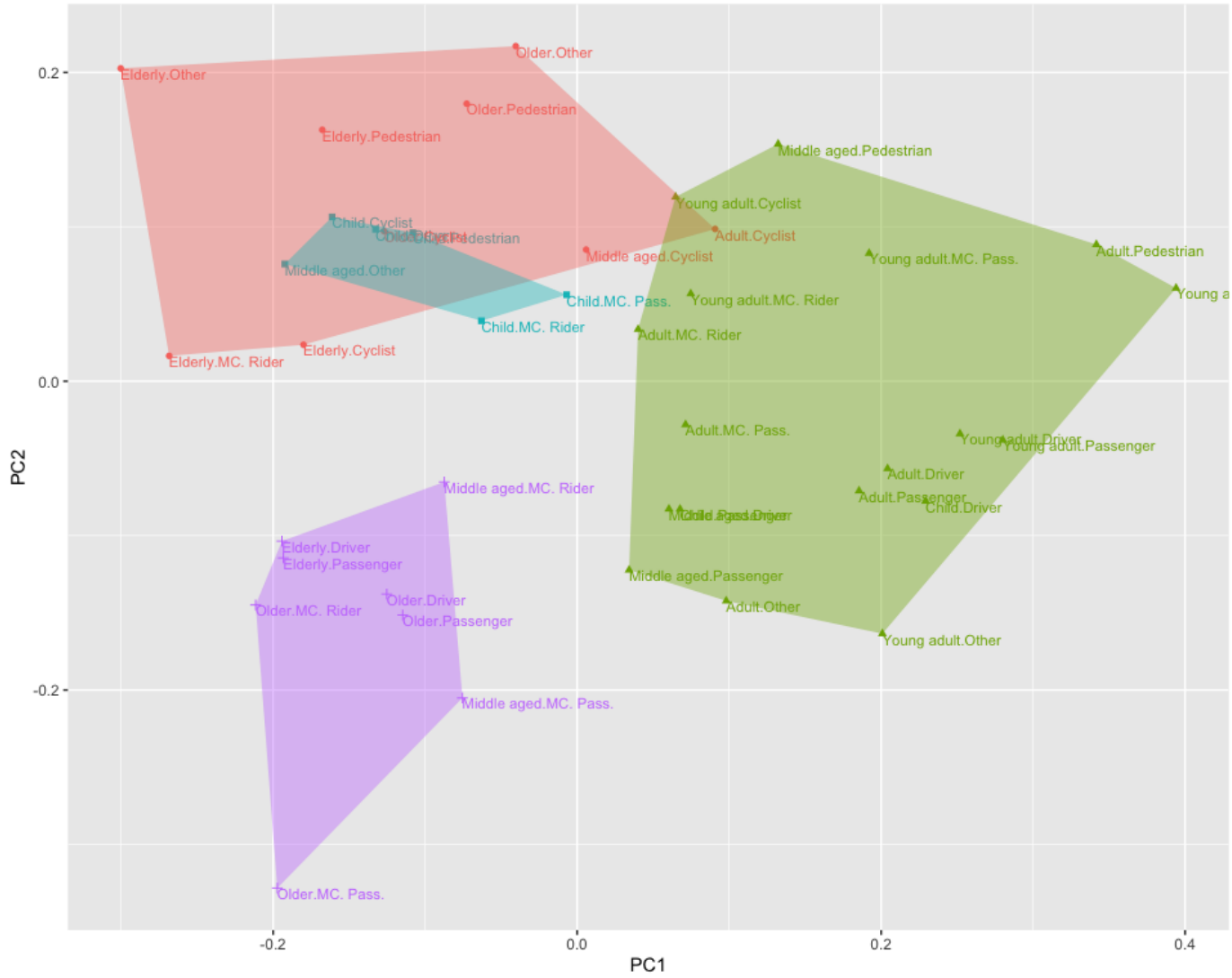
Using principal component transformation, we will plot the clusters for a clear visual respresentation:

Hide

```
groups <- cutree(pfit, k = kbest.p)
princ <- prcomp(pmatrix)
nComp <- 2
project <- as.data.frame(predict(princ, newdata=pmatrix)[,1:nComp])
project.plus <- cbind(project, cluster=as.factor(groups), Group=clust.df$Road.Use
r.Age)

h <- do.call(
    rbind,
    lapply(
        unique(groups),
        function(c) {
            f <- subset(project.plus, cluster == c);
            f[chull(f), ]
        }
    )
)
```

Hide

```
ggplot(project.plus, aes(x = PC1, y = PC2)) +
    geom_point(aes(shape = cluster, color = cluster)) +
    geom_text(aes(label = Group, color = cluster),
              hjust = 0, vjust = 1, size = 3) +
    geom_polygon(data = h, aes(group = cluster, fill = as.factor(cluster)),
                 alpha = 0.4, linetype = 0) +
    theme(legend.position = "")
```



We see distinction between our clusters in the plot with minor overlap towards the edges. Perhaps the most interesting observation from this visualisation is that the various age groups have to some degree clustered together, with the red and purple clusters consisting of almost entirely “Older” and “Elderly” groups, the green containing mostly “Adult” and “Young Adult”, and the blue being almost entirely comprised of “Child” observations. This suggests that a relationship does exist between these groups and the speed and time of the crash.

Conclusion

In this exploration and analysis of Australian road fatalities, we faced a number of challenges:

- Many variables with missing data
- Imbalanced data for classification
- Similar / duplicate information between variables (multicollinearity)
- Difficult data for clustering analysis

These challenges reinforce the need for care to be taken when handling and modelling data, especially if inference from these models will ultimately affect government or business decisions. Despite these challenges, our analysis did uncover some fascinating and potentially valuable relationships in the Australian Road Deaths Database. Many variables showed a significant relationship with the gender of a

fatality on Australian roads, for example age, type of road user, posted speed limit and time of day. Also, through clustering we found that when compared using the speed and time of a crash, many age groups of fatalities were naturally similar.

Could these insights aid in the development of targeted advertising? Or perhaps provide valuable insight into education and training opportunities?

How could this information be used to reduce the number of deaths on Australian roads?