CITS4009 Project 2 - Modelling

Jia Min HO (23337561) 50% and Amy HUNG (23702699) 50%

Introduction

The data that we are analyzing in this project is an extract from the US Accident Injury dataset, which can be obtained from Data.gov, published by US Department of Labour. The dataset contains data on all accidents, injuries and illnesses reported by US mine operators and contractors. The entire dataset spans across 15 years (2000 to 2015) and has a total of 202,814 observations. The dataset analysed in this project covers only 2,000 observations selected from the entire dataset.

In this project, we aim to help mining companies to determine whether the mining accidents will result in work being affected or not. Mining accidents may lead to under-staff issues and slow down the work progress. Hence, we hope to help mining companies to pin point these issues early and act fast on it. This will potentially help companies to achieve smooth work progress and save millions of dollars.

Data loading, overview and set up

Loading all the necessary libraries

```
library(ggplot2)
library(gridExtra)
library(ggthemes)
library(MASS)
library(dplyr)
library(usmap)
library(timeDate)
library(chron)
library(WVPlots)
library(scales)
library(treemapify)
library(lubridate)
library(tidyverse)
library(tidyr)
library(ggrepel)
library(ape)
library(knitr)
library(ROCR)
library(rpart)
library(vtreat)
library(caret)
library(ROCit)
library(lime)
library(dendextend)
library(ggradar)
library(ggmap)
library(fpc)
library(factoextra)
library(cluster)
library(pander)
library(rpart)
library(rpart.plot)
library(grDevices)
library(patchwork)
```

Setting up plotting theme

Setting up a plotting theme to ensure all graphs look coherent for the whole report.

```
cits4009_theme <- theme_minimal() +
                  theme(panel.grid.major.x = element_blank(),
                        panel.grid.minor.x = element_blank()) +
                  theme(plot.title = element_text(color = "darkred", face="bold"))
+
                  theme(axis.title = element_text(face="bold")) +
                  theme(legend.title = element_text(face="bold"))
```

Loading the dataset in R and data cleaning

Loading the dataset, which is in csv format, into R and assign the dataframe to “accident”.

```
accident <- read.csv("us_data_2000.csv", header = T, sep = ",", stringsAsFactors =
TRUE)
```

In Project 1, we have performed preliminary data cleaning on the dataset, such as (i) assigning factors to categorical features, (ii) ensuring appropriate data type for every features and (iii) assigning invalid inputs and missing values to NAs. In Project 2, we will leverage the “cleaned” dataset from Project 1, “accident_clean_v2”.

After data cleaning, we add new columns and variables that might by useful for further analysis in later stage.

```
accident_clean_v2 <- within(accident_clean_v2, {

  # Adding new variables
  worktime_before_accident <- ifelse(
    as.numeric(difftime(accident_time_format, shift_begin_time_format, units = "ho
urs")) < 0,
    as.numeric(difftime(accident_time_format, shift_begin_time_format, units = "ho
urs")+24),
    as.numeric(difftime(accident_time_format, shift_begin_time_format, units = "ho
urs")))

  tot_exp_years <- NA
  for (i in 0:5){
    tot_exp_years[tot_exper >= i*10 & tot_exper < (i+1)*10] <- paste(i*10, " - ",
(i+1)*10 , " years")
  }

  days_affected <- days_lost + days_restrict + schedule_charge
  days_affected_binary <- ifelse(days_affected, 0, 1)

  days_to_resume_work <- difftime(return_to_work_dt, accident_dt, tz , units = "da
ys")

  # Convert to factor from string type
  tot_exp_years <- as.factor(tot_exp_years)
  days_affected_binary <- as.factor(days_affected_binary)
})
```

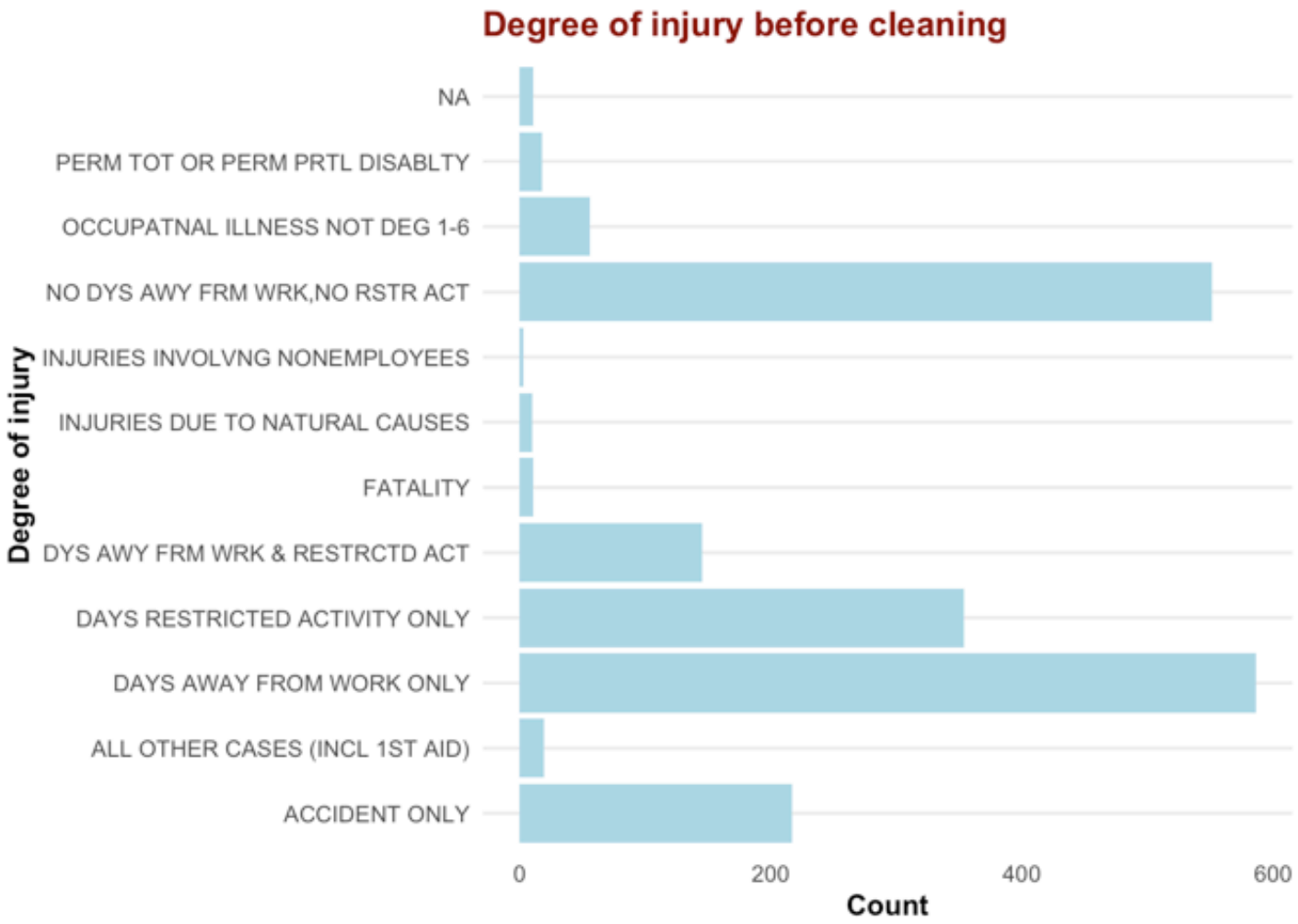
Part 1 - Classification

1.1 Choosing the response (target) variable

To achieve our aim in helping mining companies to determine if the accident affects work or not, we added 1 continuous variable “days_affected” (total of days_lost, days_restrict and schedule_charge) and 1 binary categorical variable “days_affected_binary” (0 = No days affected, 1 = Have days affected).

Understanding that degree_injury and days_affected are closely related, we then further explore these columns using graphical displays, such as bar chart and donut chart, to determine if days_affected_binary is appropriate to be selected as our target variable for classification models.

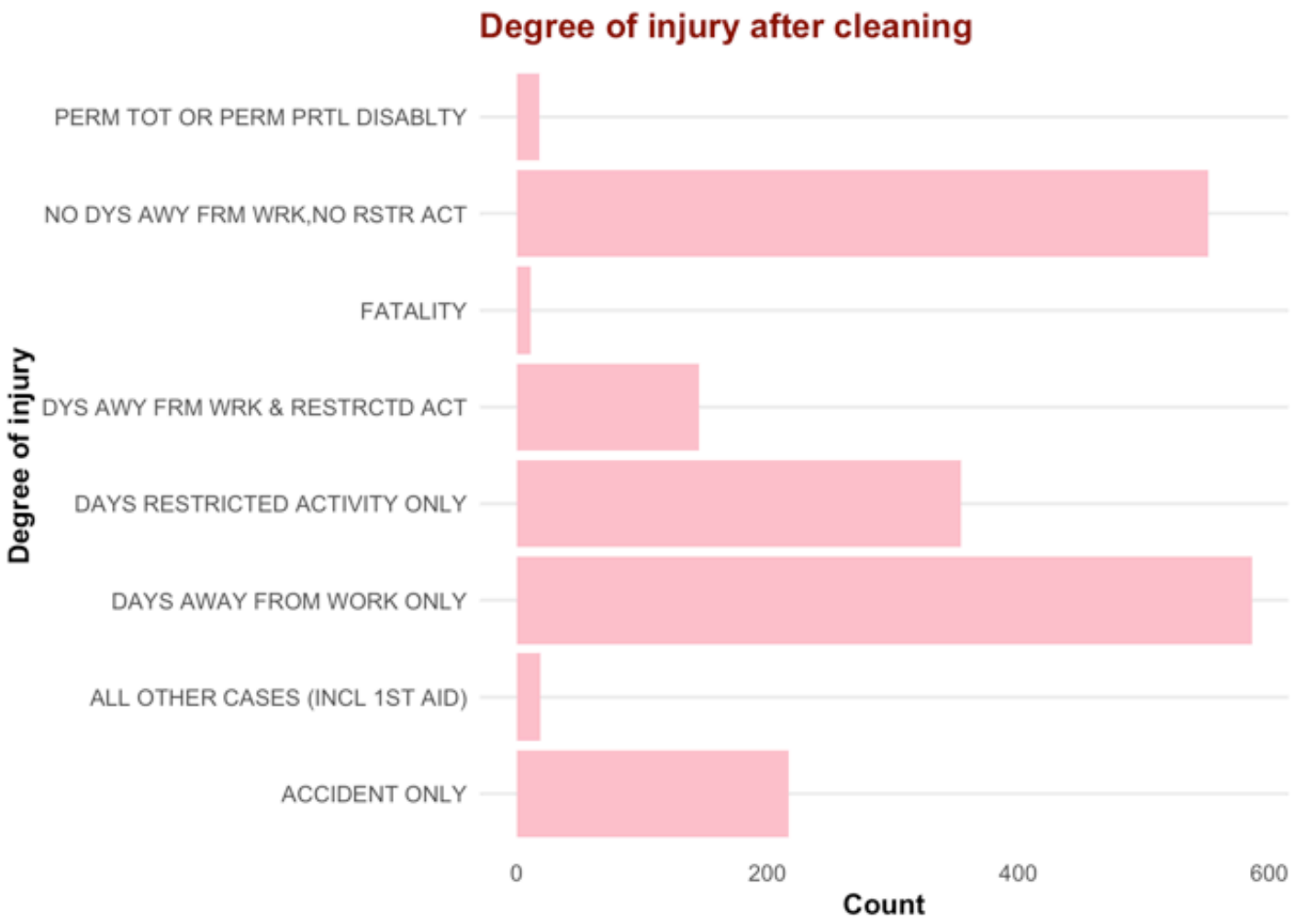
```
ggplot(data = accident_clean_v2, aes(x = degree_injury)) + geom_bar(fill = "lightb
lue") + labs(title = "Degree of injury before cleaning", x = "Degree of injury", y
= "Count") + coord_flip() + cits4009_theme
```



From the above bar chart, we can see some of the levels in degree_injury, such as “INJURIES DUE TO NATURAL CAUSES”, “OCCUPATNAL ILLNESS NOT DEG 1-6” and “INJURIES INVOLVNG NONEMPLOYEES”, are not relevant to our analysis in predicting the impact of work progress caused by injured workers. Thus, we have removed these 3 degree_injury levels and NAs from our dataset as follows.

```
accident_clean_v3 <- subset(accident_clean_v2,
                            accident_clean_v2$degree_injury !=
                              "INJURIES DUE TO NATURAL CAUSES" &
                              accident_clean_v2$degree_injury !=
                              "OCCUPATNAL ILLNESS NOT DEG 1-6" &
                              accident_clean_v2$degree_injury !=
                              "INJURIES INVOLVNG NONEMPLOYEES",)
accident_clean_v3 <- accident_clean_v3[!is.na(accident_clean_v3$degree_injury),]

ggplot(data = accident_clean_v3, aes(x = degree_injury)) + geom_bar(fill = "pink")
+ labs(title = "Degree of injury after cleaning", x = "Degree of injury", y = "Cou
nt") + coord_flip() + cits4009_theme
```




```
nrow(accident_clean_v3)
```

```
## [1] 1901
```

We have a total of 1901 observations after removing irrelevant categories in degree_injury.

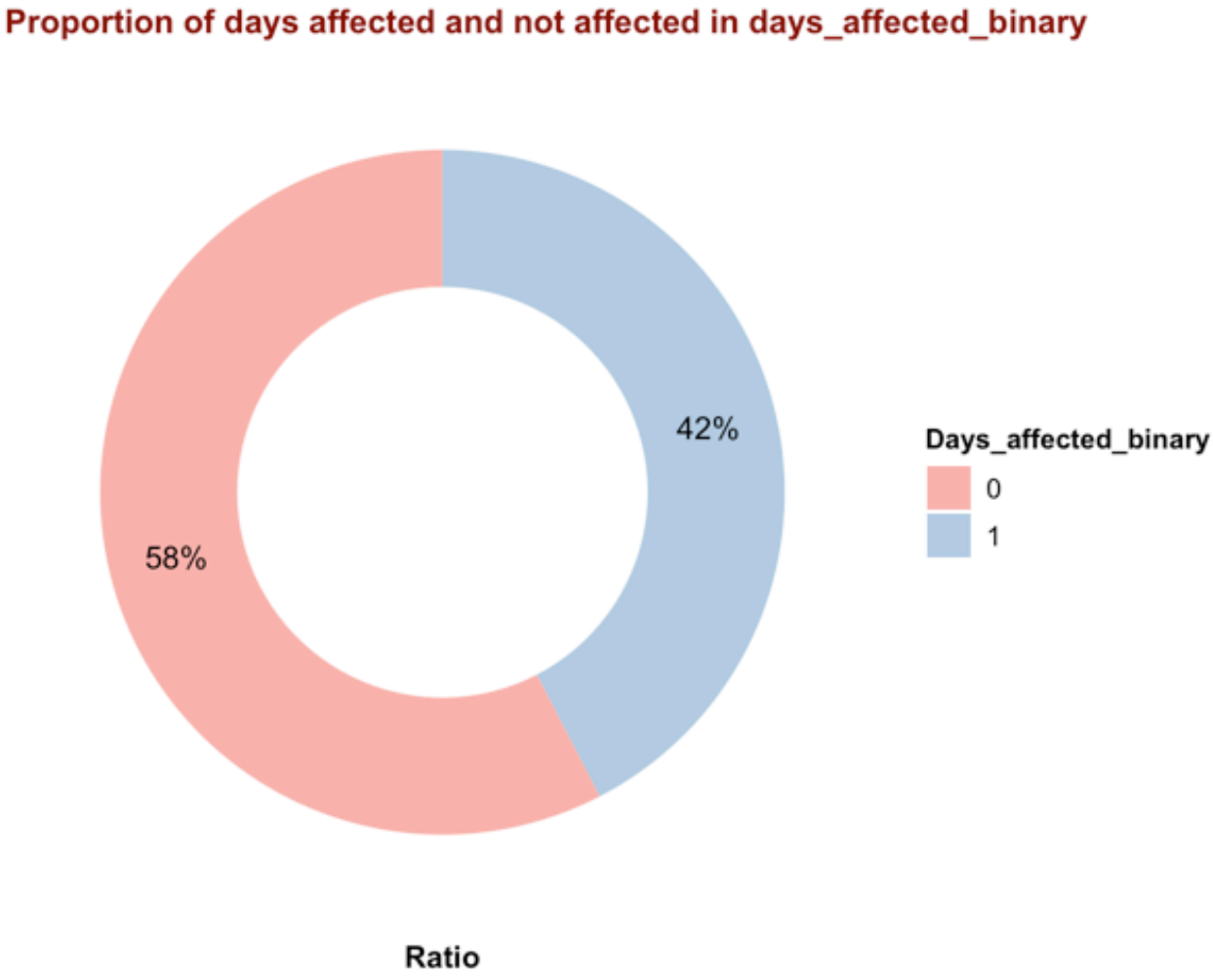
```
days_affected_binary_df <- as.data.frame(accident_clean_v3$days_affected_binary)
colnames(days_affected_binary_df) <- "days_affected"

days_affected_binary_df <- days_affected_binary_df %>%
  group_by(days_affected) %>%
  count() %>%
  ungroup() %>%
  mutate(perc = `n` / sum(`n`)) %>%
  arrange(perc) %>%
  mutate(labels = scales::percent(perc))

hsize <- 2
days_affected_binary_df <- days_affected_binary_df %>% mutate(x = hsize)

days_affected_donut <- ggplot(days_affected_binary_df,
                              aes(x= hsize, y = perc, fill=days_affected)) + geom_
col() +
  labs(title = "Proportion of days affected and not affected in days_affected_bina
ry", fill = "Days_affected_binary",
       y = "Ratio", x = "") + coord_polar("y") + xlim(c(0.2, hsize + 0.5)) +
  scale_fill_brewer(palette="Pastel1") + cits4009_theme +
  theme(plot.title = element_text(size = 12), legend.position = "right",
        legend.text = element_text(size = 10), legend.title = element_text(size =
10),
        axis.text = element_blank(), panel.grid.major = element_blank(),
        panel.grid.minor = element_blank()) + geom_text(aes(label = labels),
        position = position_stack(vjust = 0.5), size = 4)

days_affected_donut
```



The donut plot above shows the proportion of the binary variable - days affected. 58% of accidents do not affect work and 42% of accidents affected work. Not affected to affected ratio was roughly 60 : 40, which indicates a balanced proportion on both levels. Therefore, we will choose days_affected_binary as our response variable for our classification model.

1.2 Cleaning and keeping relevant feature variables

Based on Project 1, we have discovered some of the columns are irrelevant to our analysis in Project 2, thus we are cleaning for those columns.

```
accident_clean_v4 <- subset(accident_clean_v3, select = -c(
  # Removing columns which are related to response/target variable
  degree_injury, degree_injury_cd, schedule_charge, days_restrict, days_lost, days
_affected,
  # Removing columns with unique values in each observation
  document_no, narrative, closed_doc_no, accident_dt,
  # Removing columns with too many unique levels
  mine_id, controller_id, operator_id,
  # Removing columns with over 85% are invalid values and missing values
  equip_mfr_cd, equip_model_no,
  # Removing duplicated columns with same meanings
  controller_name, operator_name, subunit, ug_location, ug_mining_method, mining_e
quip,
  equip_mfr_name, classification, accident_type, occupation, activity, injury_sour
ce,
  nature_injury, inj_body_part, immed_notify, fiscal_yr, fiscal_qtr, accident_time
,
  shift_begin_time,
  # Removing columns with no relation and unrelated to our chosen target/response
variable
  invest_begin_dt, immed_notify_cd, trans_term, return_to_work_dt, days_to_resume_
work, i
  ))
```

1.3 Splitting the data

Splitting the dataset into train-calibration-test sets

```
# Performing 90/10 split to form the training and test sets
set.seed(319901)
accident_clean_v4$rgroup <- runif(dim(accident_clean_v4)[1])
dTrainAll <- subset(accident_clean_v4, rgroup <= 0.9)
dTest <- subset(accident_clean_v4, rgroup >0.9)
outcome <- 'days_affected_binary'
pos <- '1'

# Identifying names of columns that are categorical type and numerical type
vars <- setdiff(colnames(dTrainAll), c(outcome, 'rgroup'))
catVars <- vars[sapply(dTrainAll[,vars], class) %in%
                  c("factor","character")]
numericVars <- vars[sapply(dTrainAll[,vars], class) %in%
                      c("numeric", "integer")]

# Splitting dTrainAll into a training set and a calibration set
useForCal <- rbinom(n=dim(dTrainAll)[1], size=1, prob=0.1) > 0
dCal <- subset(dTrainAll, useForCal)
dTrain <- subset(dTrainAll, !useForCal)
```

Comparing the number of observation between (i) Training, (ii) Calibration and (iii) Test set

```
Total <- sum(nrow(dTrain), nrow(dCal), nrow(dTest))
tab <- matrix(c(nrow(dTrain), nrow(dCal), nrow(dTest), Total), byrow=TRUE)
colnames(tab) <- "No of observation"
rownames(tab) <- c('Training set','Calibration set',
                   'Testing set', 'Total')
tab <- as.table(tab)
train_prop <- paste(round(nrow(dTrain)/Total*100), "%")
cal_prop <- paste(round(nrow(dCal)/Total*100), "%")
test_prop <- paste(round(nrow(dTest)/Total*100), "%")
Proportion <- c(train_prop, cal_prop, test_prop, "100 %")
tab <- cbind(tab, Proportion)
kable(tab, align = "rr")
```

	No of observation	Proportion
Training set	1559	82 %
Calibration set	170	9 %
Testing set	172	9 %
Total	1901	100 %

Printing all categorical variables

```
catVars
```

```
##  [1] "contractor_id"       "subunit_cd"          "cal_yr"
##  [4] "cal_qtr"             "fips_state_cd"       "ug_location_cd"
##  [7] "ug_mining_method_cd" "mining_equip_cd"     "classification_cd"
## [10] "accident_type_cd"    "occupation_cd"       "activity_cd"
## [13] "injury_source_cd"    "nature_injury_cd"    "inj_body_part_cd"
## [16] "coal_metal_ind"      "accident_hour"       "shift_begin_hour"
## [19] "tot_exp_years"
```

Printing all numerical variables

```
numericVars
```

```
## [1] "no_injuries"              "tot_exper"
## [3] "mine_exper"               "job_exper"
## [5] "worktime_before_accident"
```

1.3 Processing the variables

1.3.1 Categorical Variables

Preforming prediction on train-calibration-test sets for categorical variables

```
# Creating function for Single variable predictions for categorical variables
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
}
```

```
for(v in catVars) {
  pi <- paste('pred',v,sep='')
  dTrain[,pi] <- mkPredC(dTrain[,outcome], dTrain[,v], dTrain[,v])
  dCal[,pi] <- mkPredC(dTrain[,outcome], dTrain[,v], dCal[,v])
  dTest[,pi] <- mkPredC(dTrain[,outcome], dTrain[,v], dTest[,v])
}
```

Evaluating training and calibration AUC for the single-variable models (categorical)

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

```
for(v in catVars) {
  pi <- paste('pred',v,sep='')
  aucTrain <- calcAUC(dTrain[,pi],dTrain[,outcome])
  if(aucTrain>=0.65) { # cut off at 0.65 to display only top performing variables
    aucCal <- calcAUC(dCal[,pi],dCal[,outcome])
    print(sprintf(
      "%s, trainAUC: %5.3f calibrationAUC: %5.3f",
      pi, aucTrain, aucCal))
  }
}
```

```
## [1] "predclassification_cd, trainAUC: 0.717 calibrationAUC: 0.732"
## [1] "predaccident_type_cd, trainAUC: 0.808 calibrationAUC: 0.766"
## [1] "predoccupation_cd, trainAUC: 0.748 calibrationAUC: 0.735"
## [1] "predactivity_cd, trainAUC: 0.756 calibrationAUC: 0.723"
## [1] "predinjury_source_cd, trainAUC: 0.781 calibrationAUC: 0.733"
## [1] "prednature_injury_cd, trainAUC: 0.827 calibrationAUC: 0.856"
## [1] "predinj_body_part_cd, trainAUC: 0.820 calibrationAUC: 0.819"
```

From the above list of single categorical variables (threshold training AUC > 0.65), there are few promising variables noted, such as (i) nature_injury_cd, (ii) inj_body_cd and (iii) accident_type_cd, having both calibration and training AUCs scored above 0.75. Moreover, we can see the AUC scores on training set are higher than that of calibration set in most of the categorical variables, except for (i) classification_cd and (ii) nature_injury_cd.

We then further perform 100-fold cross-validation to validate if the AUCs obtained above are accurate for the above 4 categorical variables.

```
# Creating a function for performing 100-fold cross-validation
calcAUC_kfold <- function(vars, training, fold = 100 ,size = 1 ,prob = 0.1,
                          do.print = TRUE) {
  list_auc <- vector()

  for (var in vars) {
    aucs <- rep(0,fold)
    for (rep in 1:length(aucs)) {
      useForCalRep <- rbinom(n=nrow(training), size = size, prob = prob) > 0
      predRep <- mkPredC(training[!useForCalRep, outcome],
                         training[!useForCalRep, var],
                         training[useForCalRep, var])
      aucs[rep] <- calcAUC(predRep, training[useForCalRep, outcome])
      }

    if(do.print == TRUE){
      print(sprintf("%s: mean: %5.3f; sd: %5.3f", var, mean(aucs), sd(aucs)))
    }

    list_auc <- append(list_auc, mean(aucs))
  }
  if(do.print == FALSE){
  print(list_auc)}
}
```

```
catVars_to_invest <- c("nature_injury_cd", "inj_body_part_cd", "accident_type_cd",
"classification_cd")
calcAUC_kfold(catVars_to_invest, dTrainAll)
```

```
## [1] "nature_injury_cd: mean: 0.821; sd: 0.030"
## [1] "inj_body_part_cd: mean: 0.802; sd: 0.033"
## [1] "accident_type_cd: mean: 0.785; sd: 0.035"
## [1] "classification_cd: mean: 0.716; sd: 0.041"
```

Based on the result from 100-fold cross-validation, we can see that:

Promising variables:

- “nature_injury_cd”: The 100-fold repeated estimates of the AUC give a mean value similar to the original AUC estimate. The original AUC estimate is very good.

- “inj_body_part_cd” & “accident_type_cd”: The 100-fold repeated estimates of the AUC give a mean value lower than the original AUC estimate. The original AUC estimate is a bit high, which may be by chance. However, the 100-fold repeated estimated AUC is still very promising.

Variables having calibration AUC > training AUC:

- “classification_cd” & “nature_injury_cd”: The 100-fold repeated estimates of the AUC give a mean value lower than the original calibration AUC. The original calibration AUC is a bit high, which may simply by chance.

Since 100-fold cross-validation result indicate the original AUC of some variables may be overstated, we decided to use 100-fold mean AUC value in the further analysis for more accurate results.

```
calcAUC_kfold(catVars, dTrainAll)
```

```
## [1] "contractor_id: mean: 0.493; sd: 0.021"
## [1] "subunit_cd: mean: 0.586; sd: 0.040"
## [1] "cal_yr: mean: 0.522; sd: 0.042"
## [1] "cal_qtr: mean: 0.506; sd: 0.037"
## [1] "fips_state_cd: mean: 0.566; sd: 0.046"
## [1] "ug_location_cd: mean: 0.591; sd: 0.035"
## [1] "ug_mining_method_cd: mean: 0.561; sd: 0.034"
## [1] "mining_equip_cd: mean: 0.564; sd: 0.037"
## [1] "classification_cd: mean: 0.710; sd: 0.037"
## [1] "accident_type_cd: mean: 0.791; sd: 0.034"
## [1] "occupation_cd: mean: 0.677; sd: 0.039"
## [1] "activity_cd: mean: 0.695; sd: 0.039"
## [1] "injury_source_cd: mean: 0.712; sd: 0.037"
## [1] "nature_injury_cd: mean: 0.817; sd: 0.033"
## [1] "inj_body_part_cd: mean: 0.802; sd: 0.031"
## [1] "coal_metal_ind: mean: 0.537; sd: 0.035"
## [1] "accident_hour: mean: 0.530; sd: 0.037"
## [1] "shift_begin_hour: mean: 0.517; sd: 0.042"
## [1] "tot_exp_years: mean: 0.638; sd: 0.040"
```

From the above list, we can see the top 3 promising categorical variables, which having highest mean AUC values, are (i) inj_body_part_cd, (ii) nature_injury_cd and (iii) accident_type_cd.

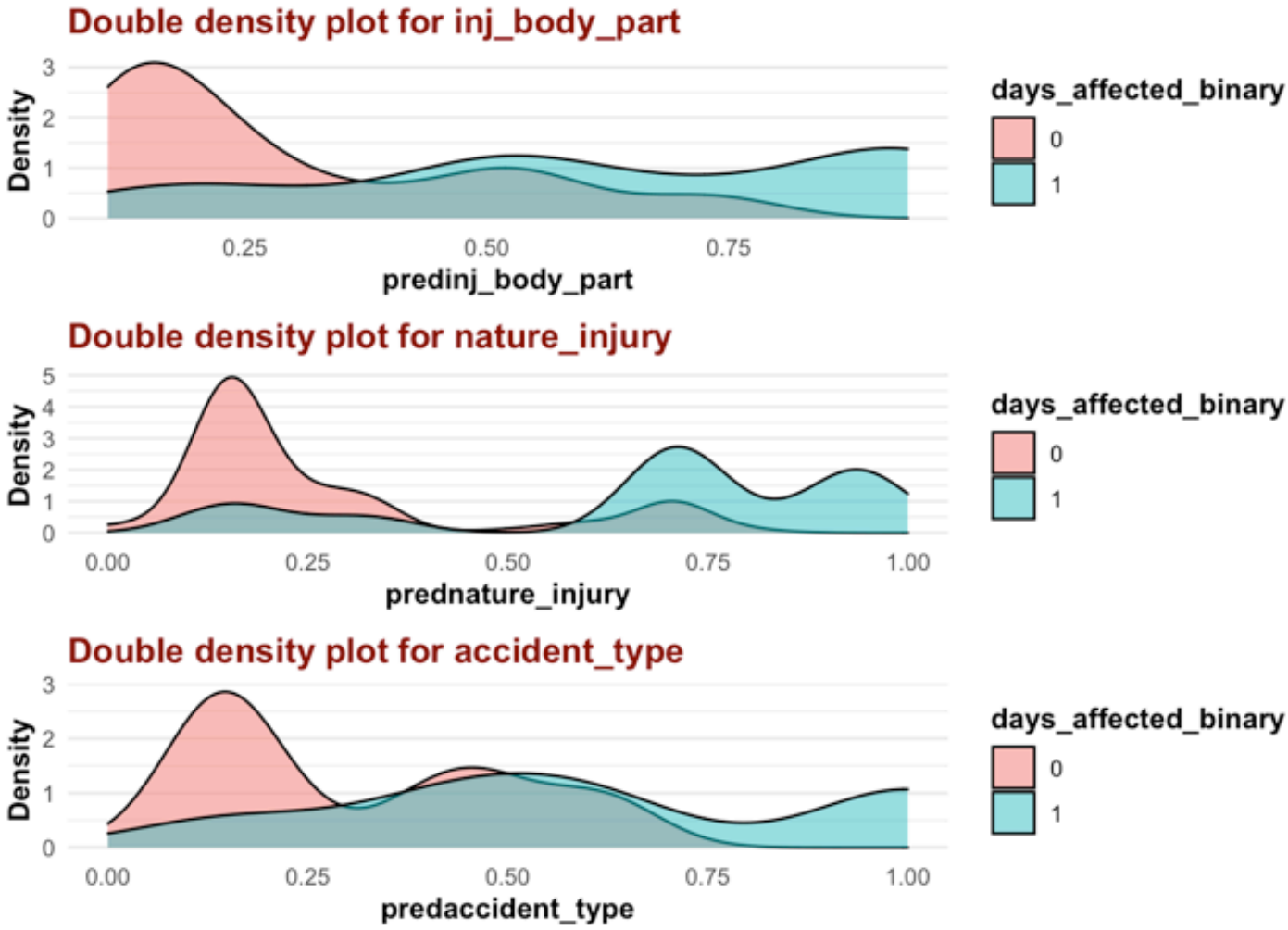
Double density plots are then plotted for each top 3 promising categorical variables to inspect the predicted probabilities on these variables.

```
fig1 <- ggplot(dCal, aes(x = predinj_body_part_cd, fill = days_affected_binary)) +
  geom_density(alpha = 0.5) +
  labs(title = "Double density plot for inj_body_part", x = "predinj_body_part", y
= "Density") +
  cits4009_theme

fig2 <- ggplot(dCal, aes(x = prednature_injury_cd, fill = days_affected_binary)) +
  geom_density(alpha = 0.5) +
  labs(title = "Double density plot for nature_injury", x = "prednature_injury", y
= "Density") +
  cits4009_theme

fig3 <- ggplot(dCal, aes(x = predaccident_type_cd, fill = days_affected_binary)) +
  geom_density(alpha = 0.5) +
  labs(title = "Double density plot for accident_type", x = "predaccident_type", y
= "Density") +
  cits4009_theme

grid.arrange(fig1,fig2,fig3,ncol=1)
```



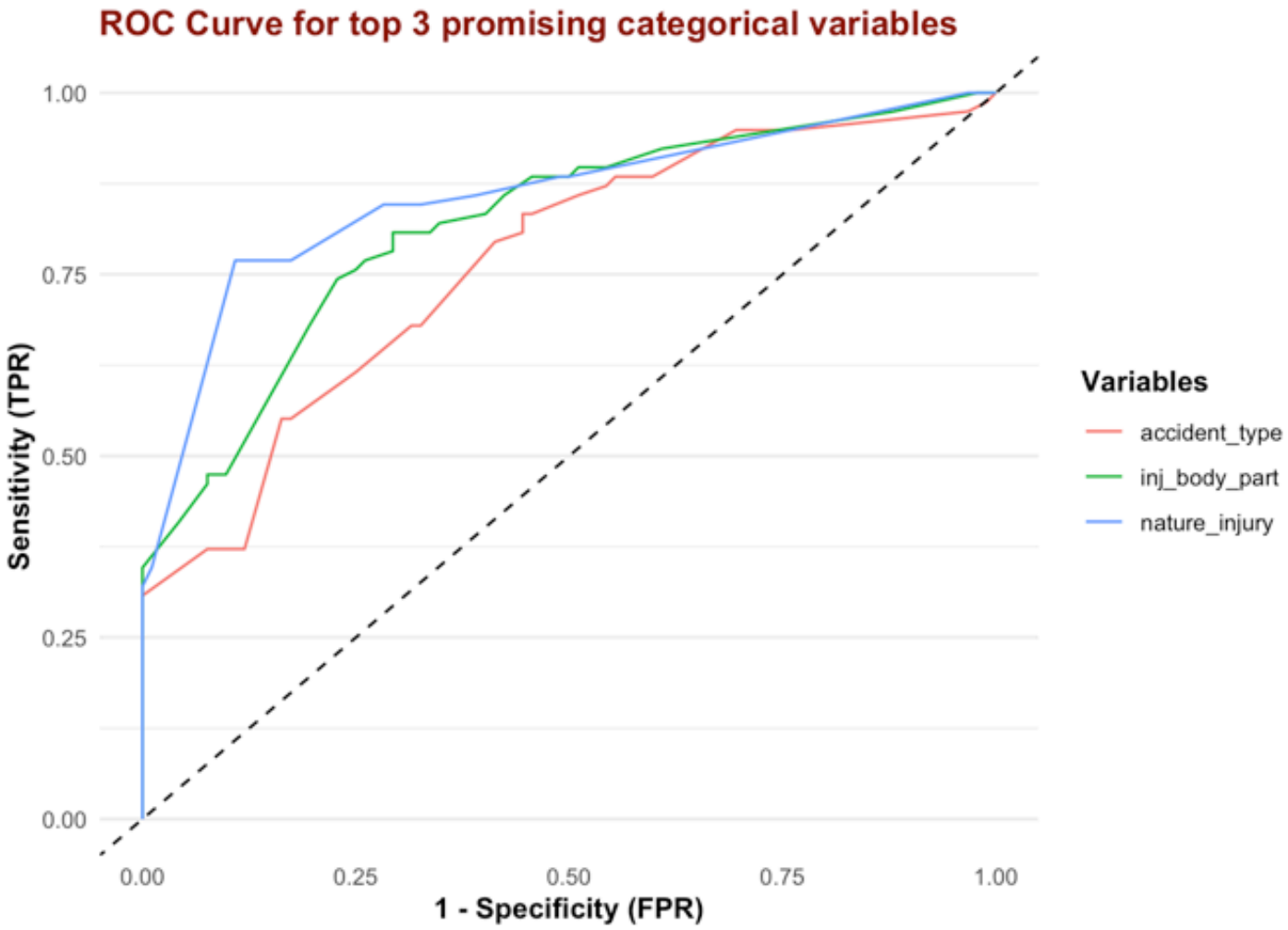
From all the above plots, we can see the distributions of no days affected (days_affected_binary = 0) and having days affected (days_affected_binary = 1) are dominating low probability region and high probability region respectively, which indicate satisfying predictions from these single variable models.

Next, ROC curves are plotted for the top 3 promising categorical variables as follows:

```
predinj_body_part_cd_roc <- prediction(dCal$predinj_body_part_cd, dCal[,outcome])
prednature_injury_cd_roc <- prediction(dCal$prednature_injury_cd, dCal[,outcome])
predaccident_type_cd_roc <- prediction(dCal$predaccident_type_cd, dCal[,outcome])

preds <- list(predinj_body_part_cd_roc, prednature_injury_cd_roc, predaccident_typ
e_cd_roc)
var.names <- c('inj_body_part', 'nature_injury', 'accident_type')
roc.df <- data.frame()
for(i in 1:length(var.names)){
  pred <- preds[[i]]
  var.name <- var.names[i]
  perf <- performance( pred, "tpr", "fpr" )
  roc.df <- rbind(roc.df, data.frame(
              'fpr'=unlist(perf@x.values),
              'tpr'=unlist(perf@y.values),
              'threshold'=unlist(perf@alpha.values),
              'Variables'=var.name))
}

ggplot(roc.df, aes(x=fpr, y=tpr, group=Variables, color=Variables))+
  geom_line() +
  geom_abline(intercept = 0, slope = 1, linetype='dashed') +
  cits4009_theme +
  labs(title = "ROC Curve for top 3 promising categorical variables", x = "1 - Spe
cificity (FPR)", y = "Sensitivity (TPR)")
```



From the ROC curve, we can observe that nature_injury has the best performance as compared to inj_body_part, followed by accident_type. This plot further supports the AUC values calculated above.

1.3.2 Numerical Variables

Performing prediction on train-calibration-test sets and evaluating training and calibration AUC for the single-variable models for numerical variables. We will use quantile( ) and cut( ) commands to bin the numeric features into a number of ranges and then use the range labels as a new categorical variable.

```
# Creating a function for Single variable predictions for numerical variables
mkPredN <- function(outCol,varCol,appCol) {
  cuts <- unique(as.numeric(quantile(varCol, probs=seq(0, 1, 0.1), na.rm=T)))
  varC <- cut(varCol, cuts)
  appC <- cut(appCol, cuts)
  mkPredC(outCol, varC, appC)
}

for(v in numericVars) {
  pi <-paste('pred', v,sep='')
  dTrain[,pi] <- mkPredN(dTrain[,outcome], dTrain[,v], dTrain[,v])
  dTest[,pi] <- mkPredN(dTrain[,outcome], dTrain[,v], dTest[,v])
  dCal[,pi] <- mkPredN(dTrain[,outcome], dTrain[,v], dCal[,v])
  aucTrain <- calcAUC(dTrain[,pi],dTrain[,outcome])

  if(aucTrain>=0.5) { # cut off at 0.5 to display only top performing variables
    aucCal<-calcAUC(dCal[,pi],dCal[,outcome])
    print(sprintf(
      "%s, trainAUC: %5.3f calibrationAUC: %5.3f",
      pi,aucTrain,aucCal))
  }
}
```

```
## [1] "predno_injuries, trainAUC: 0.635 calibrationAUC: 0.661"
## [1] "predtot_exper, trainAUC: 0.657 calibrationAUC: 0.589"
## [1] "predmine_exper, trainAUC: 0.646 calibrationAUC: 0.661"
## [1] "predjob_exper, trainAUC: 0.654 calibrationAUC: 0.609"
## [1] "predworktime_before_accident, trainAUC: 0.551 calibrationAUC: 0.528"
```

From the above list, we can see 2 numerical variables (i) no_injuries and (ii) mine_exper are having higher Calibration AUC than Training AUC. 100-fold cross-validation is then performed to validate if the higher Calibration AUC is simply by chance.

```
calcAUC_kfold(numericVars, dTrainAll)
```

```
## [1] "no_injuries: mean: 0.512; sd: 0.022"
## [1] "tot_exper: mean: 0.594; sd: 0.042"
## [1] "mine_exper: mean: 0.622; sd: 0.041"
## [1] "job_exper: mean: 0.614; sd: 0.043"
## [1] "worktime_before_accident: mean: 0.500; sd: 0.044"
```

From the result above, the 100-fold repeated estimates of the AUC give a mean value lower than the original calibration AUC for both (i) no_injuries and (ii) mine_exper. The original calibration AUC are a bit high, which may simply by chance.

Similar to categorical variables, since 100-fold cross-validation result indicate the original AUC of some variables may be overstated, we decided to use 100-fold mean AUC value in the further analysis for more accurate results.

1.3.3 Feature Selection: Select the top performing categorical and numerical variables

Calculating the 100-fold mean AUC for all variables (categorical & numerical)

```
all.vars <- c(catVars, numericVars)
df <- data.frame(model.type = 'univariate', model.name = all.vars,
                 mean_auc = calcAUC_kfold(all.vars, dTrain, 100, 1, 0.1, do.print
= FALSE))
```

```
df %>% arrange(-mean_auc)
```

model.type <chr>	model.name <chr>	mean_auc <dbl>
univariate	nature_injury_cd	0.8137945
univariate	inj_body_part_cd	0.7996297
univariate	accident_type_cd	0.7921924
univariate	injury_source_cd	0.7161354
univariate	classification_cd	0.7043203
univariate	activity_cd	0.7015865
univariate	occupation_cd	0.6655347
univariate	tot_exp_years	0.6410629
univariate	mine_exper	0.6157828
univariate	job_exper	0.6052633

1-10 of 24 rows Previous **1** 2 3 Next

From the above table, we can see the top 3 performing variables are (i) nature_injury_cd, (ii) inj_body_part_cd and (iii) accident_type_cd, having mean AUC > 0.79.

In this project, we will perform feature selection based on Drop-in-Deviance Test. Top performing variables will be selected if they give a significant drop in deviance (Deviance = -2 * logLikelihood) when compare against that of the Null model.

```
# Creating a function to compute log likelihood so as to compute the deviance
logLikelihood <- function(ytrue, ypred, epsilon=1e-6) {
sum(ifelse(ytrue==pos, log(ypred+epsilon), log(1-ypred-epsilon)), na.rm=T)
}
```

Computing the log likelihood of the Null model

```
logNull <- logLikelihood(
dTrain[,outcome], sum(dTrain[,outcome]==pos)/nrow(dTrain)
)
cat("The log likelihood of the Null model is:", logNull)
```

```
## The log likelihood of the Null model is: -1059.668
```

After getting the log likelihood of the Null model, we compare it (in fact deviance) with the deviance of all variables (categorical & numerical) to compute the drop in deviance.

Running through all categorical variables to select top performers

```
# selCatVars is a vector that keeps the names of the top performing categorical va
riables.
selCatVars <- c()
minDrop <- 150     # threshold on selecting top performers

# Computing the drop in deviance for all categorical variables, only selecting tho
se having drop > threshold
for (v in catVars) {
pi <- paste('pred', v, sep='')
devDrop <- 2*(logLikelihood(dTrain[,outcome], dTrain[,pi]) - logNull)
if (devDrop >= minDrop) {
print(sprintf("%s, deviance reduction: %g", pi, devDrop))
selCatVars <- c(selCatVars, pi)
}
}
```

```
## [1] "predclassification_cd, deviance reduction: 290.012"
## [1] "predaccident_type_cd, deviance reduction: 579.157"
## [1] "predoccupation_cd, deviance reduction: 422.279"
## [1] "predactivity_cd, deviance reduction: 405.731"
## [1] "predinjury_source_cd, deviance reduction: 463.534"
## [1] "predinj_body_part_cd, deviance reduction: 562.488"
## [1] "predtot_exp_years, deviance reduction: 165.115"
```

From the above, we can see the selected categorical variables with largest reduction in deviance are having the highest AUC values among all the variables too.

Running through all numerical variables to select top performers

```
# selNumVars is a vector that keeps the names of the top performing numerical vari
ables.
selNumVars <- c()
minDrop <- 150       # threshold on selecting top performers

# Computing the drop in deviance for all numerical variables, only selecting those
having drop > threshold
for (v in numericVars) {
pi <- paste('pred', v, sep='')
devDrop <- 2*(logLikelihood(dTrain[,outcome], dTrain[,pi]) - logNull)
if (devDrop >= minDrop) {
print(sprintf("%s, deviance reduction: %g", pi, devDrop))
selNumVars <- c(selNumVars, pi)
}
}
```

```
## [1] "predno_injuries, deviance reduction: 333.604"
## [1] "predtot_exper, deviance reduction: 164.522"
## [1] "predmine_exper, deviance reduction: 174.589"
## [1] "predjob_exper, deviance reduction: 164.07"
```

Based on the above results, 12 top performing features are selected as follows:

```
(selVars <- c(selCatVars, selNumVars))
```

```
##  [1] "predclassification_cd" "predaccident_type_cd"  "predoccupation_cd"
##  [4] "predactivity_cd"       "predinjury_source_cd"  "prednature_injury_cd"
##  [7] "predinj_body_part_cd"  "predtot_exp_years"     "predno_injuries"
## [10] "predtot_exper"         "predmine_exper"        "predjob_exper"
```

```
print(sprintf("%d variables selected", length(selVars)))
```

```
## [1] "12 variables selected"
```

After selecting the top performing single variables, we evaluate the test AUC on these variables.

```
cat("Performance of the top performing single variables on the test set:\n")
```

```
## Performance of the top performing single variables on the test set:
```

```
for (v in selVars) {
  # retrieve the original variable name (after "pred")
  orig_v <- substring(v, 5)
  cat(sprintf("Variable %6s: test_AUC = %g\n", orig_v, calcAUC(dTest[,v], dTest[,o
utcome])))
}
```

```
## Variable classification_cd: test_AUC = 0.689762
## Variable accident_type_cd: test_AUC = 0.809759
## Variable occupation_cd: test_AUC = 0.676398
## Variable activity_cd: test_AUC = 0.738624
## Variable injury_source_cd: test_AUC = 0.728002
## Variable nature_injury_cd: test_AUC = 0.828331
## Variable inj_body_part_cd: test_AUC = 0.796669
## Variable tot_exp_years: test_AUC = 0.637267
## Variable no_injuries: test_AUC = 0.631579
## Variable tot_exper: test_AUC = 0.631442
## Variable mine_exper: test_AUC = 0.589227
## Variable job_exper: test_AUC = 0.607662
```

From the result above, we find that some of the variables are performing very well in the Test set (Test AUC > 0.75), which are (i) nature_injury_cd (0.83), (ii) accident_type_cd (0.81) and (iii) inj_body_part_cd (0.80). As for the top performing numerical variable is no_injuries, having Test AUC = 0.63.

1.4 Multivariate Models

In this section, we will fit the data into 2 different classification models (i) Decision Tree and (ii) Logistic regression classification. Then, we will evaluate and compare the performances of the 2 fitted models to determine the best performing model.

We will first define several functions for evaluating and visualizing the performances of different classification models in later stage.

The functions below are created for models evaluation. The function will return (i) an ROC plot, (ii) double density plot, (iii) precision plots, (iv) recall plots and (v) a confusion matrix.

```
evaluate_performance <- function(predictions, true.values, do.print=TRUE,
                                 threshold.value=0.5){

  if(do.print){

    # (i) ROC plot

    predObj <- prediction(predictions, true.values)
    perf <- performance(predObj, "tpr", "fpr")
    roc.df <- data.frame('fpr'=unlist(perf@x.values),
                         'tpr'=unlist(perf@y.values),
                         'threshold'=unlist(perf@alpha.values))
```

```
    roc.p <- ggplot(roc.df, aes(x=fpr, y=tpr)) +
      geom_line(color = "coral1") +
      geom_abline(intercept = 0, slope = 1, linetype='dashed') +
      labs(title = "ROC Curve", x = "1 - Specificity (FPR)", y = "Sensitivity (TPR
)") +
      cits4009_theme + theme( legend.position = "none")

    # (ii) Double density plot

    ddp.p <- ggplot(data.frame(predictions=predictions,  true.values=true.values),
                    aes(x=predictions, fill=true.values, linetype=true.values)) +
                    geom_density(alpha=0.5)  +
                    labs(title = "Double Density Plot", y = "Density", x = "Predic
tion",
                         fill = outcome, linetype= outcome) +
                    cits4009_theme +
                    theme(legend.position = c(.95, .95),
                          legend.justification = c("right", "top"),
                          legend.box.just = "right",
                          legend.margin = margin(6, 6, 6, 6))

    grid.arrange(roc.p, ddp.p, ncol = 2)

    # (iii) Precision & (iv) recall plot

    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]]
    prec_rec.Frame <- data.frame(threshold=prec.x,
                           precision=precision,
                           recall=recall)
    pnull <-mean(true.values==pos)

    prec.p <- ggplot(prec_rec.Frame, aes(x=threshold)) +
                      geom_line(aes(y=precision), colour="blue4") +
                      coord_cartesian(xlim = c(0,1), ylim=c(0,1) )  +
                      labs(title = "Precision Curve", x = "Threshold", y = "Precis
ion") +
                      cits4009_theme +
                      theme(aspect.ratio = 1, legend.position = "none")

    rec.p <- ggplot(prec_rec.Frame, aes(x=threshold)) +
                    geom_line(aes(y=recall), colour="darkcyan") +
                    coord_cartesian(xlim = c(0,1) ) +
                    labs(title = "Recall Curve", x = "Threshold", y = "Recall") +
                    cits4009_theme +
                    theme(aspect.ratio = 1, legend.position = "none")
```

```
    grid.arrange(prec.p, rec.p, ncol = 2)


    # (v) Confusion matrix table

    cat('Confusion Matrix')
    ctab.test <- table(pred=predictions > threshold.value, true.values=true.values
)
    print(kable(ctab.test))
  }


  eval.auc <- calcAUC(predictions, true.values)
}

# Creating a function for summaries of both AUC computation and the plots above
evaluate_model <- function(model.type, model.name, predictions.train,
                                  true.values.train, predictions.cal, true.values
.cal){
  train.auc <- evaluate_performance(predictions.train, true.values.train, do.print
= TRUE)
  cal.auc <- evaluate_performance(predictions.cal, true.values.cal, do.print = FAL
SE)
  result <- data.frame(model.type = model.type,
                       model.name = model.name,
                       train.auc = train.auc,
                       cal.auc = cal.auc)
  rownames(result) <- model.name

  result
}
```

The functions below are created for computing (i) accuracy, (ii) precision, (iii) recall and (iv) F1 score, and put them into a data frame for easy reference.

```
# Creating a function for computing the 4 performance measures
performanceMeasures <- function(pred, truth, name = "model", threshold = 0.5) {
  ctable <- table(truth = truth, pred = pred >= threshold)
  accuracy <- sum(diag(ctable)) / sum(ctable)
  precision <- ctable[2, 2] / sum(ctable[, 2])
  recall <- ctable[2, 2] / sum(ctable[2, ])
  f1 <- 2 * precision * recall / (precision + recall)

  data.frame(model = name,
             precision = precision,
             recall = recall,
             f1 = f1,
             accuracy = accuracy)}

# Creating a function for making a performance measures summary table
```

```
predictvalue <- function(model, data, dt = FALSE) {
  if(dt == TRUE) {
    pred <- predict(model, newdata=data)[,pos]
    return(pred)
  }
  pred <- predict(model, newdata=data)
  return(pred)
}

perfM_table <- function(model,training,calibration,test, threshold = 0.5, dt = FAL
SE) {

  # Setting up Pander Options for better format of summary table
  panderOptions("plain.ascii", TRUE)
  panderOptions("keep.trailing.zeros", TRUE)
  panderOptions("table.style", "simple")
  perf_justify <- "lrrrr"

  # Comparing performance measures on train-calibration-test sets and put inside s
ummary table
  pred_train <- predictvalue(model, training, dt = dt)
  truth_train <- training[, outcome]
  pred_cal <- predictvalue(model, calibration, dt = dt)
  truth_cal <- calibration[, outcome]
  pred_test <- predictvalue(model, test, dt = dt)
  truth_test <- test[, outcome]

  trainperf_tree <- performanceMeasures(pred_train, truth_train, "training", thres
hold = threshold)
  calperf_tree <- performanceMeasures(pred_cal, truth_cal, "calibration", threshol
d = threshold)
  testperf_tree <- performanceMeasures(pred_test, truth_test, "test", threshold =
threshold)
  perftable <- rbind(trainperf_tree, calperf_tree,testperf_tree)
  pandoc.table(perftable, justify = perf_justify)
}
```

Formula for fitting in the models with all the selected variables and outcome:

```
formula <- paste(outcome,' ~ ',  paste(selVars, collapse=' + '), sep='')
formula
```

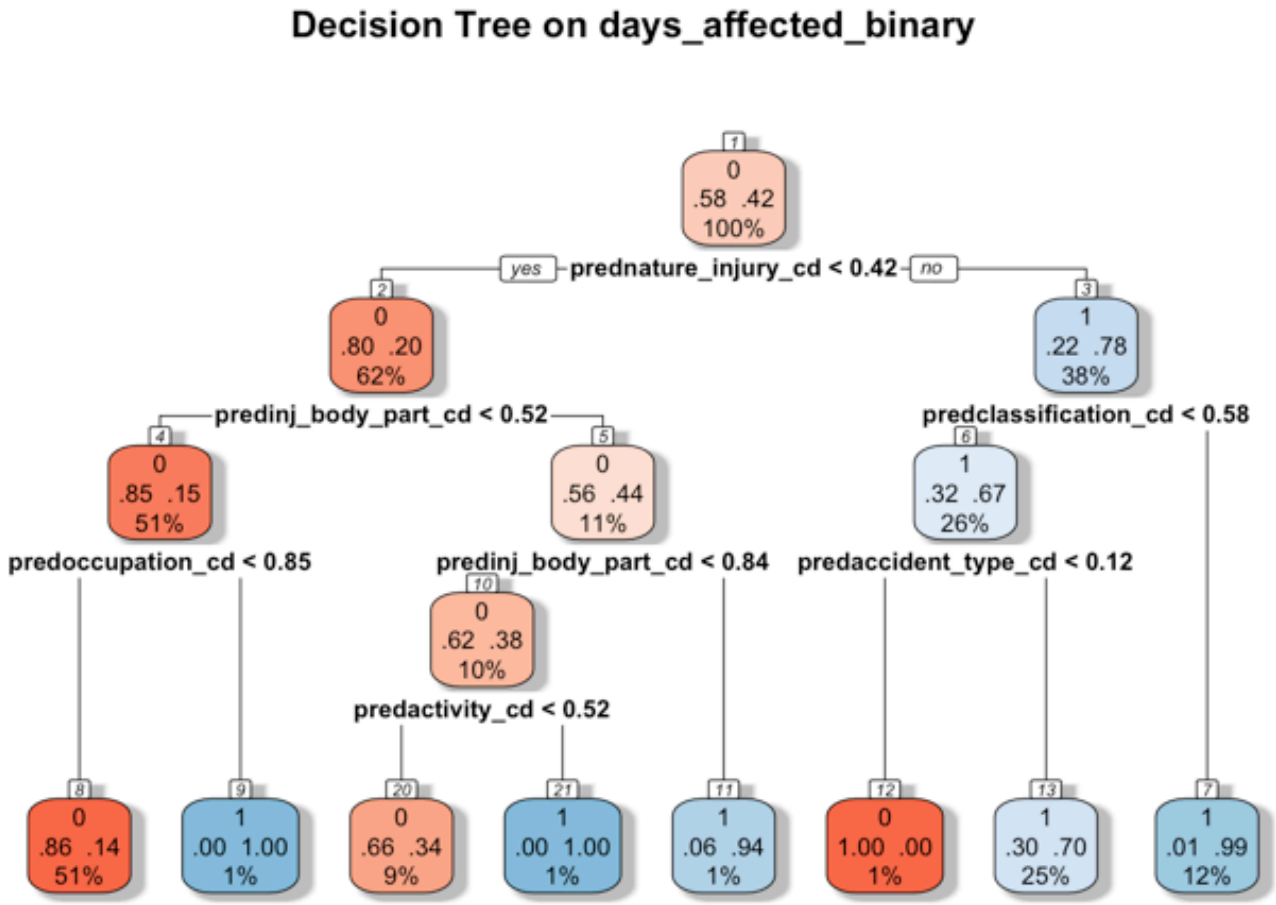
```
## [1] "days_affected_binary ~ predclassification_cd + predaccident_type_cd + pred
occupation_cd + predactivity_cd + predinjury_source_cd + prednature_injury_cd + pr
edinj_body_part_cd + predtot_exp_years + predno_injuries + predtot_exper + predmin
e_exper + predjob_exper"
```

1.4.1 Multivariate Model: Decision Tree

Decision tree is plotted below using the rpart.plot R package. The decision tree is a model with binary response. Each node shows the predicted class (0,1), the predicted probability of work affected or not, and the percentage of observations in the node.

```
# Decision Tree Model
dt.model <- rpart(formula, data=dTrain)

# Visualize the decision tree with rpart.plot
rpart.plot(dt.model, box.palette="RdBu", type = 2, extra = 104, shadow.col = "gray
", nn=TRUE, main = "Decision Tree on days_affected_binary")
```



From the above decision tree, we can see the first split at the root is based on nature_injury_cd, which may indicate it is the most importance feature among the selected variables. Moving to the leaf nodes of the decision tree, splits are mainly done by variables of inj_body_part, accident_type, classification and activity_cd. This aligns with the results from the single variable models above, where these variables have high AUC values. We will further confirm their importance using LIME analysis in later stage.

Performance evaluation is then performed for the above decision tree:

```
epsilon = 1e-6
dt.trainpred <- predict(dt.model, newdata=dTrain)[,pos]
dt.model.ll <- sum(ifelse(dTrain[,outcome]==pos, log(dt.trainpred+epsilon), log(1-
dt.trainpred-epsilon)), na.rm=T)
print(paste('Decision Tree Model Log Likelihood:', dt.model.ll))
```

```
## [1] "Decision Tree Model Log Likelihood: -668.258425755726"
```

```
print(paste('Null Model Log Likelihood:', logNull))
```

```
## [1] "Null Model Log Likelihood: -1059.66773923368"
```

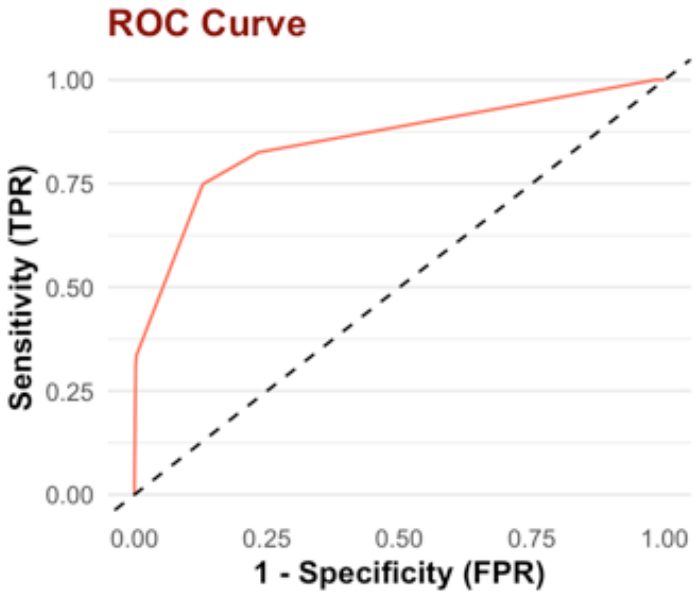
Decision Tree Log likelihood compared to Null model: The decision tree with the selected variables model has a log likelihood value of -668.26 which is larger than that of the Null model's -1059.67. This indicates decision tree model performs better than null model.

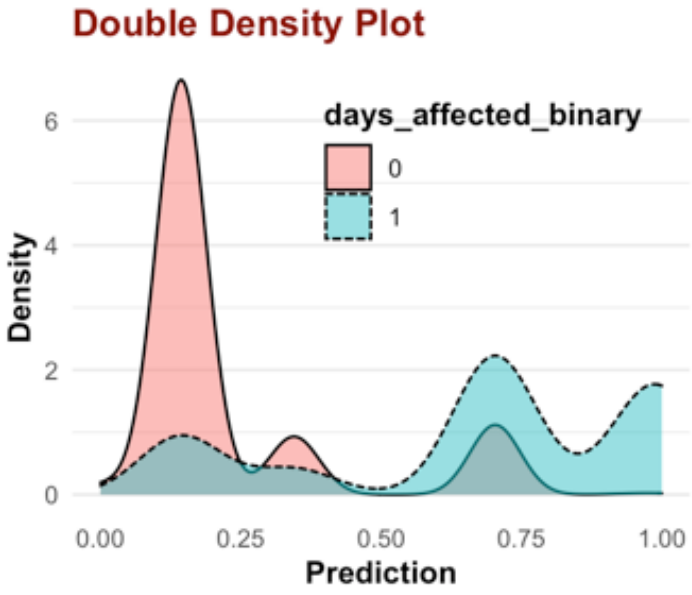
```
perfM_table(dt.model, dTrain, dCal, dTest, dt = TRUE)
```

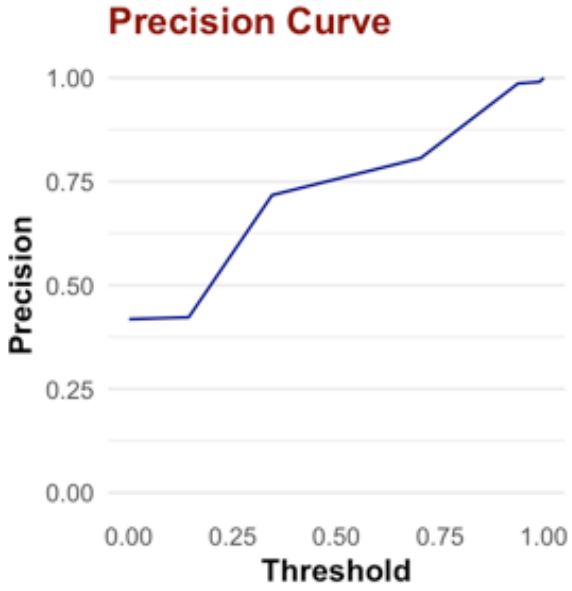
```
##
##
## model           precision   recall       f1   accuracy
## ------------- ----------- -------- -------- ----------
## training           0.8066   0.7485   0.7765     0.8198
## calibration        0.7595   0.7692   0.7643     0.7824
## test               0.8235   0.7368   0.7778     0.8140
```

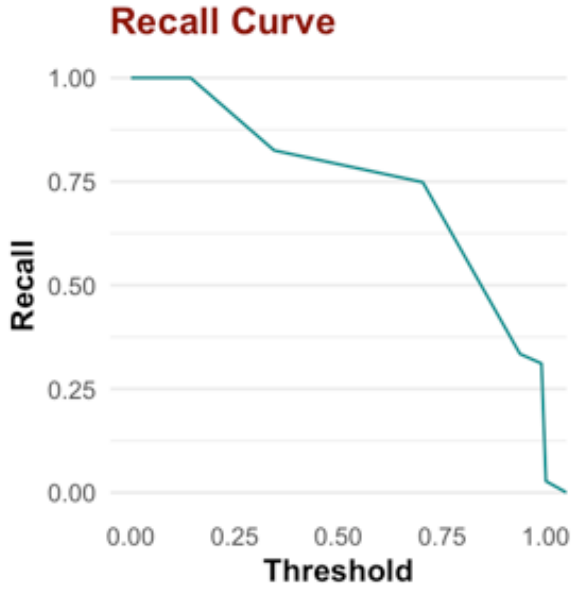
From the above table, decision tree model performs quite stable among training, calibration and test set, there is no significant drop noted from any performance measurement in any data set. The performance of the model seems good for having scores > 0.73 in all 4 measures.

```
evaluate_model(
  model.type = 'multivariate',
  model.name = 'dt.model',
  predictions.train = predict(dt.model, newdata=dTrain)[,pos],
  true.values.train = dTrain[,outcome],
  predictions.cal = predict(dt.model, newdata=dCal)[,pos],
  true.values.cal = dCal[,outcome])
```










```
## Confusion Matrix
##
## |      |   0|   1|
## |:-----|---:|---:|
## |FALSE | 790| 164|
## |TRUE  | 117| 488|
```

	model.type <chr>	model.name <chr>	train.auc <dbl>	cal.auc <dbl>
dt.model	multivariate	dt.model	0.8520057	0.8012124

1 row

From the above model evaluation, we can see:

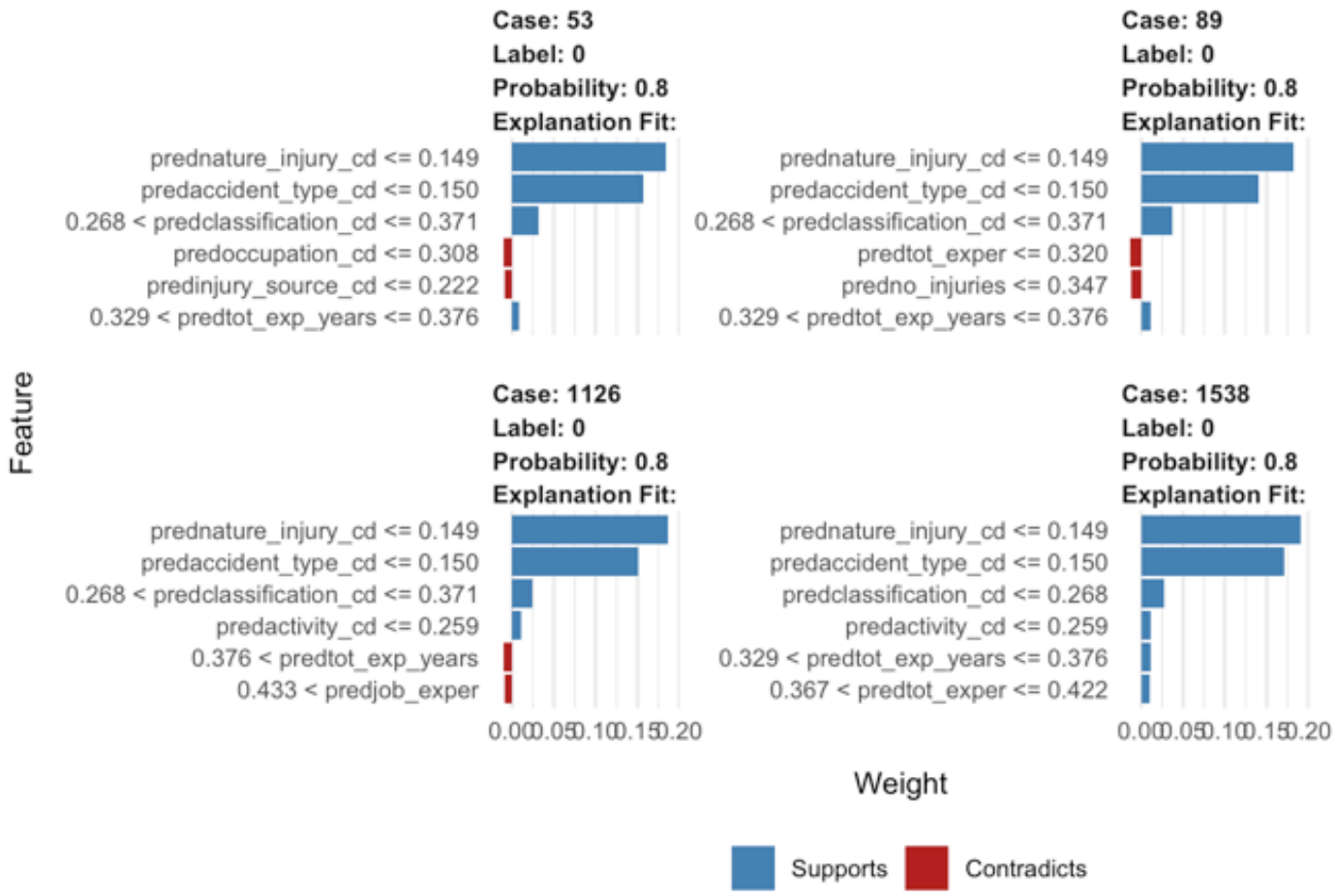
- AUC values: Decision tree model has training AUC of 0.85 and a calibration AUC of 0.80 (consistent with ROC curve), which are relatively high as compared to those single variable models.

- Double density plot: The two distributions between 0 (No work affected) and 1 (work affected) seems quite well separated, with 0 (no work affected) dominating the low probability region and 1 (work affected) dominating the high probability region. Moreover, density of 1 (work affected) bypassed 0 (no work affected) near the probability of 0.42, which is consistent to our split in days_affect_binary. However, there is a small peak on the distribution of 0 (no work affected) around the region of probability = 0.7, as well as for the density of distribution of 1 (work affected), we noted a drop near the region of probability = 0.85. We may need further investigation on the quality of prediction by decision tree model.
- Precision and recall plots: Consistent to the double density plot result, good results are also noted from precision and recall plots as well.

LIME is used for inspecting our answers and find out which feature plays a key role in predicting whether work is affected or not. Output of LIME is a list of explanations, reflecting the contribution of each feature to the prediction of a data sample. This provides local interpretability, and it also allows to determine which feature changes will have most impact on the prediction.

We further evaluate our decision tree model using LIME as follows:

```
# building model
dt.model_lime <- caret::train(x=dTrain[selVars], y= dTrain[,outcome], method='rpar
t')
# making prediction
dt_pred <- predict(dt.model_lime, dTest[selVars], type="prob")
# using LIME
dt_explainer <- lime(dTrain[selVars], model=as_classifier(dt.model_lime),
                bin_continuous=TRUE)
cases <- c(5,9,100,133)
dt_example <- dTest[cases,selVars]
dt_explanation <- lime::explain(dt_example, dt_explainer, n_labels = 1, n_features
= 6)
decision_tree_lime <- plot_features(dt_explanation)
decision_tree_lime
```



From the above LIME analysis, we can observe that “nature_injury” and “accident_type” have the most significant impact in the prediction. This result aligns with our single variable model above where “nature_injury” and “accident_type” are in the top 3 highest AUC values. Moreover, this result is consistent with the finding above that “nature_injury” is one of the most importance feature among the selected variables to perform decision tree classification, as it was the “root” of the decision tree.

Though the performance of decision tree model seems promising, we will further compare the decision tree model’s performance to another model. We will fit logistic regression classification in the next stage to determine which model is more appropriate for the dataset.

1.4.2 Multivariate Model: Logistic Regression Classification

```
# Logistic Regression Classification Model
logreg.model <- glm(formula, data=dTrain, family = binomial(link = "logit"))
summary(logreg.model)
```

```
##
## Call:
## glm(formula = formula, family = binomial(link = "logit"), data = dTrain)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5589  -0.6385  -0.3749   0.6964   2.3981
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)            -5.6606     1.3618  -4.157 3.23e-05 ***
## predclassification_cd   0.5290     0.5311   0.996 0.319215
## predaccident_type_cd    2.3090     0.4761   4.850 1.24e-06 ***
## predoccupation_cd       3.3742     0.4841   6.970 3.16e-12 ***
## predactivity_cd         0.2392     0.5155   0.464 0.642592
## predinjury_source_cd    1.3701     0.4620   2.966 0.003018 **
## prednature_injury_cd    2.1524     0.3406   6.319 2.63e-10 ***
## predinj_body_part_cd    1.3376     0.3892   3.437 0.000589 ***
## predtot_exp_years       0.0933     1.7809   0.052 0.958217
## predno_injuries         3.5025     3.8193   0.917 0.359118
## predtot_exper           0.6411     1.7799   0.360 0.718713
## predmine_exper         -0.3745     0.8288  -0.452 0.651388
## predjob_exper          -1.4797     0.8608  -1.719 0.085621 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2119.3  on 1558  degrees of freedom
## Residual deviance: 1320.4  on 1546  degrees of freedom
## AIC: 1346.4
##
## Number of Fisher Scoring iterations: 9
```

From the above summary of logistic regression model, we can observe there are several significant variables having p-value < 0.05, which are (i) accident_type, (ii) occupation, (iii) injury_source, (iv) nature_injury and (v) inj_body_part. This aligns with the results from the single variable models above, where these variables have high AUC values. We will further compare and validate this finding with the LIME analysis in the later stage.

Moreover, as compared to the Null deviance, the residual deviance of the logistic regression model is lower, which means the logistic regression model performs better than the null model.

Performance evaluation is then performed for the above logistic regression model fitted as follows:

```
epsilon = 1e-6
logreg.trainpred <- predict(logreg.model, newdata=dTrain, type = 'response')
logreg.model.ll <- sum(ifelse(dTrain[,outcome]==pos, log(logreg.trainpred+epsilon)
,
                              log(1-logreg.trainpred-epsilon)), na.rm=T)
print(paste('Logistic Regression Model Log Likelihood:', logreg.model.ll))
```

```
## [1] "Logistic Regression Model Log Likelihood: -660.191023260467"
```

```
print(paste('Null Model Log Likelihood:', logNull))
```

```
## [1] "Null Model Log Likelihood: -1059.66773923368"
```

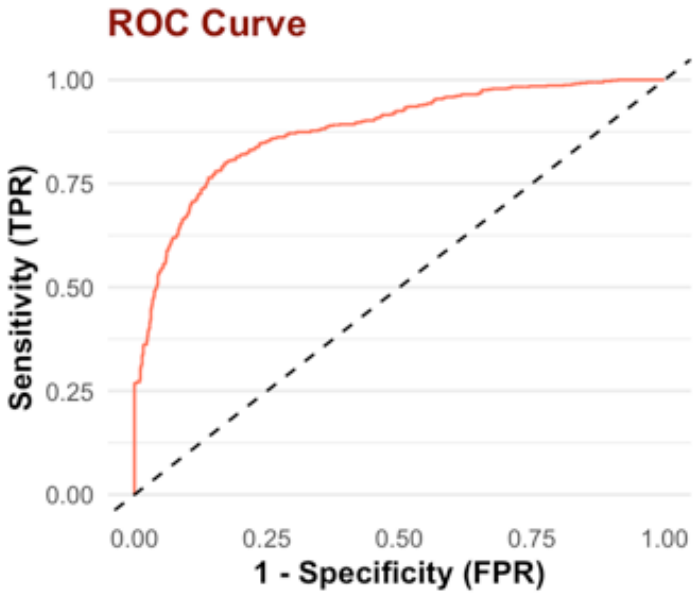
Logistic Regression Classification Log likelihood as compared to Null model: The Logistic Regression Classification model has a log likelihood value of -660.19 which is larger than that of the Null model's -1059.67. This indicates logistic regression model performs better than null model.

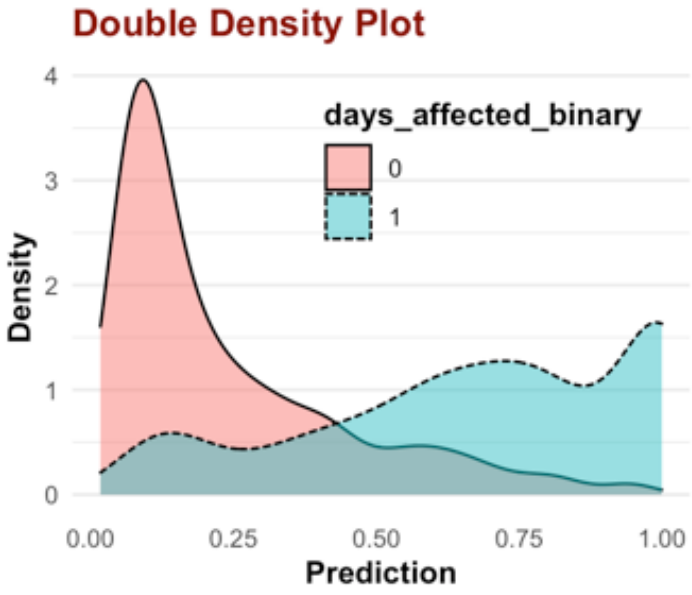
```
perfM_table(logreg.model, dTrain, dCal ,dTest)
```

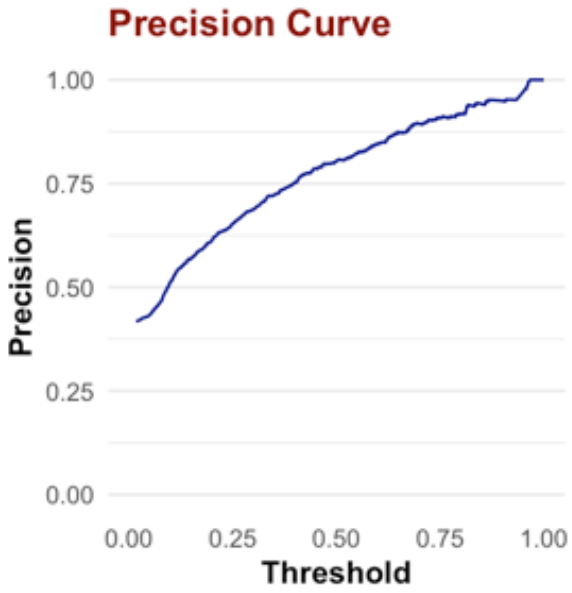
```
##
##
## model           precision   recall       f1   accuracy
## ------------- ----------- -------- -------- ----------
## training           0.8577   0.6196   0.7195     0.7979
## calibration        0.8545   0.6026   0.7068     0.7706
## test               0.8750   0.5526   0.6774     0.7674
```

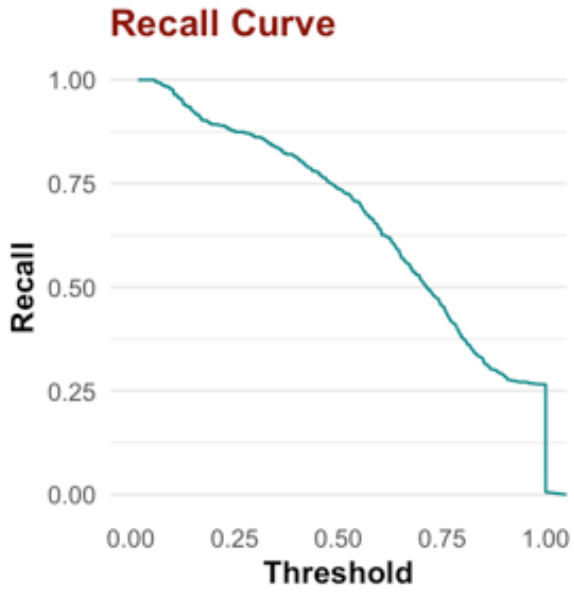
From the above table, logistic regression model performs quite stable among training, calibration and test set, there is no significant drop noted from most of performance measurement in any data set. The precision of logistic regression model are very good, with values in all dataset > 0.85. However, the recall, f1 and accuracy score are lower as compared to decision tree and a drop is noted between test set and train-calibration sets. We will further discuss on this issue in later stage.

```
# Model Evaluation
evaluate_model(
  model.type = 'multivariate',
  model.name = 'logreg.model',
  predictions.train = predict(logreg.model, newdata=dTrain, type = 'response'),
  true.values.train = dTrain[,outcome],
  predictions.cal = predict(logreg.model, newdata=dCal, type = 'response'),
  true.values.cal = dCal[,outcome])
```










```
## Confusion Matrix
##
## |      |   0|   1|
## |:-----|---:|---:|
## |FALSE | 791| 171|
## |TRUE  | 116| 481|
```

	model.type <chr>	model.name <chr>	train.auc <dbl>	cal.auc <dbl>
logreg.model	multivariate	logreg.model	0.8793281	0.847408

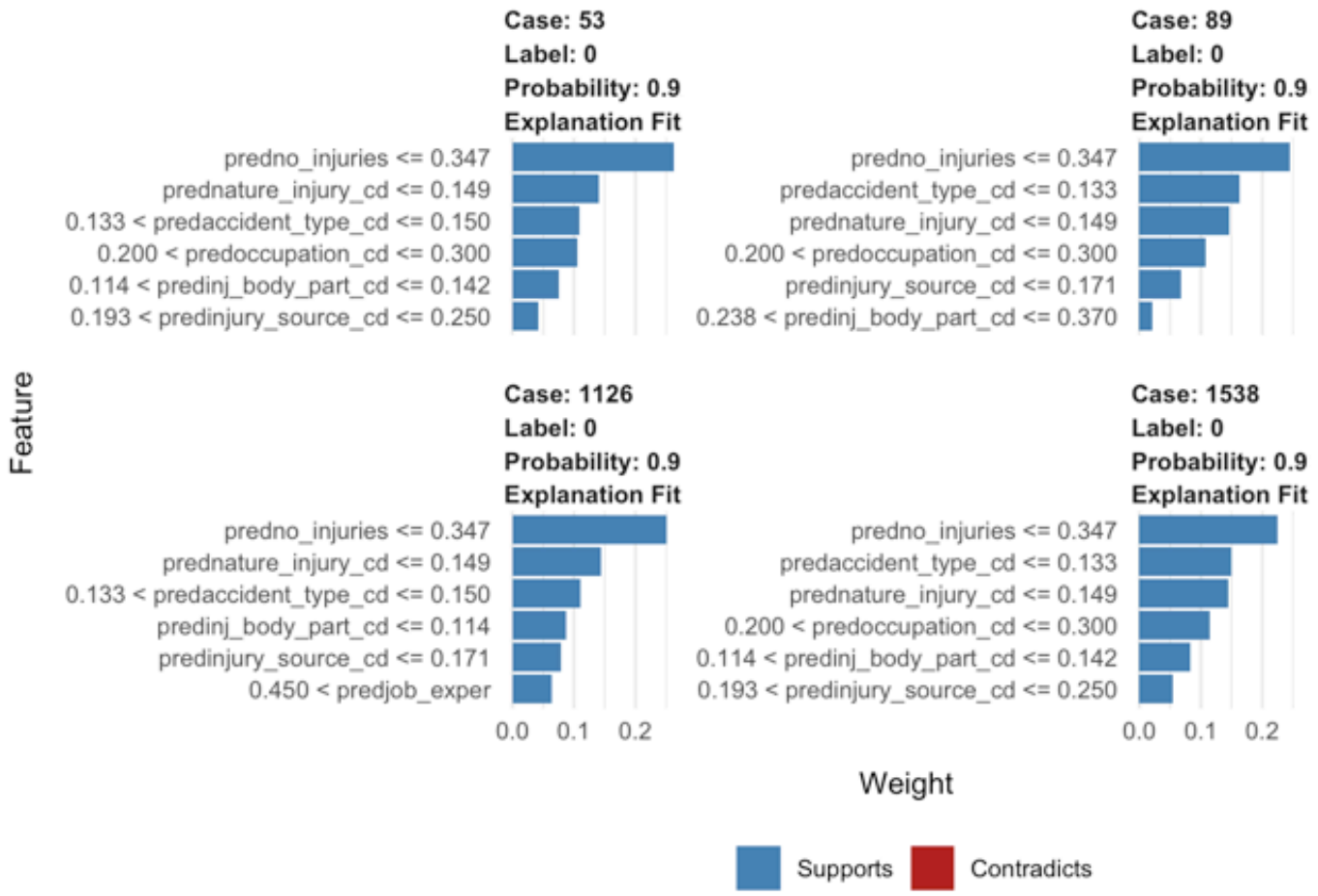
1 row

From the above model evaluation, we can see:

- AUC values: The Logistic Regression Classification model has a training AUC of 0.88 and a calibration AUC of 0.85 (consistent with ROC curve), which are relatively high as compared to those single variable models.
- Double density plot: The two distributions between 0 (No work affected) and 1 (work affected) seems well separated, with 0 (no work affected) dominating the low probability region and 1 (work affected) dominating the high probability region. Moreover, density of 1 (work affected) bypassed 0 (no work affected) near the probability of 0.42, which is consistent to our split in days_affect_binary. Compared decision tree model, the quality of prediction seems more stable with no extra peaks for 0 (no work affected) distribution nor significant drop for 1 (work affected) distributions in high probability region.
- Precision and recall plots: Consistent to double density plot result, promising results are also noted from precision and recall plots as well.

We further evaluate our logistic regression model using LIME as follows:

```
# Logistic Regression Classification
logreg.model_lime <- caret::train(x=dTrain[selVars], y= dTrain[,outcome],
                         method='glm', family=binomial(link="logit"))
# making prediction
log_pred <- predict(logreg.model_lime, dTest[selVars], type="prob")
# using LIME
log_explainer <- lime(dTrain[selVars], model=as_classifier(logreg.model_lime),
                  bin_continuous=TRUE, n_bins=10)
cases <- c(5,9,100,133)
log_example <- dTest[cases,selVars]
log_explanation <- lime::explain(log_example, log_explainer, n_labels = 1, n_featu
res = 6)
logistic_regression_lime <- plot_features(log_explanation)
logistic_regression_lime
```



From the above LIME analysis, we can observe that “no_injury”, “nature_injury” and “accident_type” have the most significant impact in the prediction. This result aligns with our single variable model above where “nature_injury” and “accident_type” have the top 3 highest AUC values. They are also consistent to the model summary above where “accident_type” and “nature_injury” are the significant variables in the logistic regression model. As for no_injury, it has a relatively low AUC (0.51) as compared to all other variables and it is not shown as significant variable in the summary of logistic regression model. However, it is the top performer among all the numerical variables. This may be because it is imbalanced with most no. of injuries as 1 with several outliers. Transformation on this feature may be considered.

1.6 Comparing all models

After fitting 2 multivariate models by 2 common classification techniques, (i) Decision Tree and (ii) Logistics Regression classification, we will compare the performance for these 2 models, top 12 single variable models and null model to determine the more appropriate model.

1.6.1 Multivariate Models vs Null Model

```
print(paste('Null Model Log Likelihood:', logNull))
```

```
## [1] "Null Model Log Likelihood: -1059.66773923368"
```

```
print(paste('Decision Tree Model Log Likelihood:', dt.model.ll))
```

```
## [1] "Decision Tree Model Log Likelihood: -668.258425755726"
```

```
print(paste('Logistic Regression Model Log Likelihood:', logreg.model.ll))
```

```
## [1] "Logistic Regression Model Log Likelihood: -660.191023260467"
```

From the above, both decision tree model and logistic regression model have larger log likelihood than null model, with logistic regression classification model having the largest log likelihood. This indicates both multivariate models perform better than null model, and logistic regression model performs the best among three models in term of log likelihood comparison.

1.6.2 Multivariate Models vs Single Variable Models

```
# Mean 100-fold AUC values for top 10 performing single variable models:
top_single.auc <- head(df %>% arrange(-mean_auc), 10)
colnames(top_single.auc) <- c("model.type", "model.name", "AUC")

# Training AUC for both multivariate models:
dt.train.auc <- evaluate_performance(predict(dt.model, dTrain)[,pos], dTrain[,outc
ome], do.print = FALSE)
logreg.train.auc <- evaluate_performance(predict(logreg.model, dTrain, type = 'res
ponse'), dTrain[,outcome], do.print = FALSE)
multi_model.train.auc <- data.frame(model.type = "multivariate",
                                    model.name = c("dt.model", "logreg.model"),
                                    AUC = c(dt.train.auc, logreg.train.auc))

# Combining both top 10 single variable models and mulitvariate models:
auc.compare.df <- rbind(top_single.auc, multi_model.train.auc)
auc.compare.df <- auc.compare.df %>% arrange(-AUC)
auc.compare.df
```

model.type <chr>	model.name <chr>	AUC <dbl>
multivariate	logreg.model	0.8793281
multivariate	dt.model	0.8520057
univariate	nature_injury_cd	0.8137945
univariate	inj_body_part_cd	0.7996297
univariate	accident_type_cd	0.7921924
univariate	injury_source_cd	0.7161354
univariate	classification_cd	0.7043203

univariate	activity_cd	0.7015865
univariate	occupation_cd	0.6655347
univariate	tot_exp_years	0.6410629

1-10 of 12 rows Previous **1** 2 Next

From the above table, we can see both multivariate models perform better than all the single variable models by higher AUC observed. While logistic regression model has the largest AUC indicates it performs the best among all models in terms of AUC comparison.

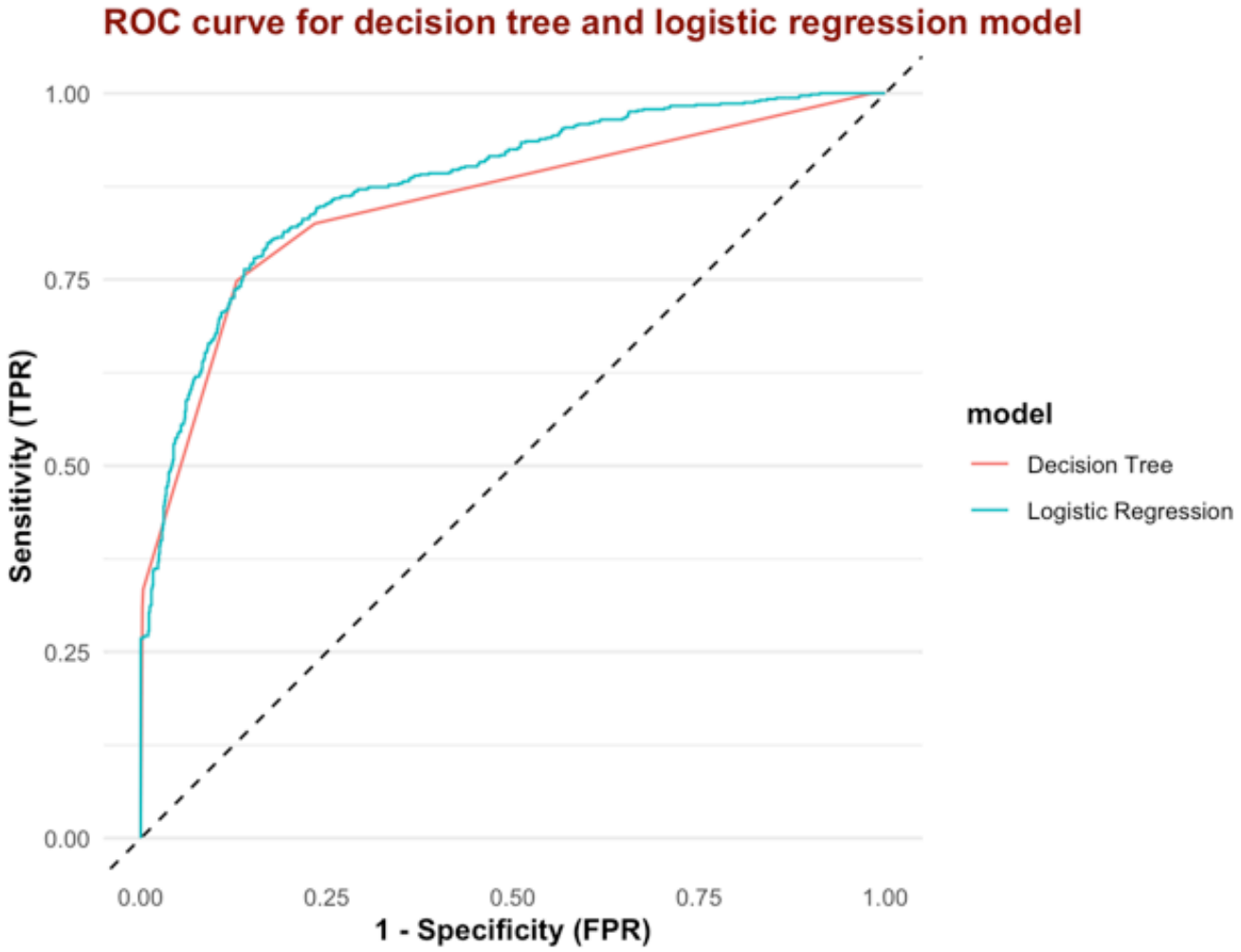
1.6.3 Among Multivariate Models

Based on the above result, logistic regression model outperforms decision tree in terms of log likelihood and AUC.

ROC curves comparison between these 2 models is performed as follows:

```
preds <- list(prediction(logreg.trainpred, dTrain[,outcome]), prediction(dt.trainp
red, dTrain[,outcome]))
model.names <- c('Logistic Regression', 'Decision Tree')
roc.df <- data.frame()
for(i in 1:length(model.names)){
  pred <- preds[[i]]
  model.name <- model.names[i]
  perf <- performance(pred, "tpr", "fpr" )
  roc.df <- rbind(roc.df, data.frame(
              "fpr"=unlist(perf@x.values),
              "tpr"=unlist(perf@y.values),
              "threshold"=unlist(perf@alpha.values),
              "model"=model.name))
}

ggplot(roc.df, aes(x=fpr, y=tpr, group=model, color=model))+
  geom_line() +
  geom_abline(intercept = 0, slope = 1, linetype='dashed') +
  labs(title = "ROC curve for decision tree and logistic regression model", x = "1
- Specificity (FPR)", y = "Sensitivity (TPR)") + cits4009_theme +
  theme(legend.position = "right", aspect.ratio = 1)
```



From the ROC plots above, we can observe that logistics regression classification model has a better performance. This further supports the higher AUC values noted in logistic regression model.

Performance measurement for both models:

```
perfM_table(dt.model, dTrain, dCal, dTest, dt = TRUE)
```

```
##
##
## model           precision   recall       f1   accuracy
## ------------- ----------- -------- -------- ----------
## training           0.8066   0.7485   0.7765     0.8198
## calibration        0.7595   0.7692   0.7643     0.7824
## test               0.8235   0.7368   0.7778     0.8140
```

```
perfM_table(logreg.model, dTrain, dCal ,dTest)
```

```
##
##
## model           precision   recall       f1   accuracy
## ------------- ----------- -------- -------- ----------
## training           0.8577   0.6196   0.7195     0.7979
## calibration        0.8545   0.6026   0.7068     0.7706
## test               0.8750   0.5526   0.6774     0.7674
```

In terms of precision, logistic regression model seems perform better than decision tree model. However, in terms of recall, f1 score and accuracy, decision tree model seems perform better than logistic regression model. In order to decide which model is more appropriate, we have to consider the business needs for the mining company. In this case, mining companies may prefer most of the accidents marked as work affected are actually work affected, so that they will not waste resources in handling false alarm. Thus, we treat precision as the most important factor among 4 performance measures and logistic regression model still consider to be appropriate.

To conclude, in terms of log likelihood, AUC, ROC plots and double density plots, logistic regression model outperforms decision tree model, single variable models and null model. Therefore, we consider logistic regression model as the most appropriate model and better fit the business needs of mining companies.

1.7 Final Performance Evaluation of Selected Model on Test Set

Based on previous analysis, knowing that logistic regression model is the most appropriate model in our report, we then evaluate its performance on the test set.

```
logNull.test <- logLikelihood(dTest[,outcome], sum(dTest[,outcome]==pos)/nrow(dTes
t)
)
epsilon = 1e-6
logreg.testpred <- predict(logreg.model, newdata=dTest, type = 'response')
logreg.model.ll.test <- sum(ifelse(dTest[,outcome]==pos, log(logreg.testpred+epsil
on),
                                log(1-logreg.testpred-epsilon)), na.rm=T)
print(paste('Null Model Log Likelihood:', logNull.test))
```

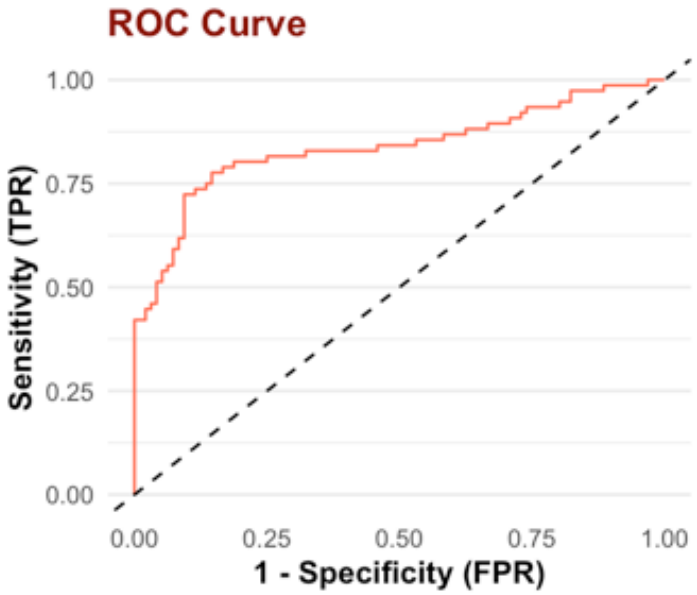
```
## [1] "Null Model Log Likelihood: -118.055889769589"
```

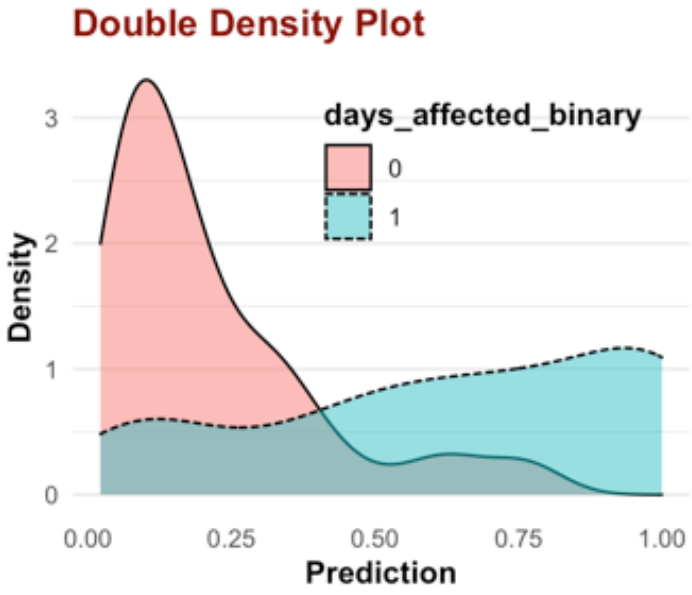
```
print(paste('Logistic Regression Model Log Likelihood:', logreg.model.ll.test))
```

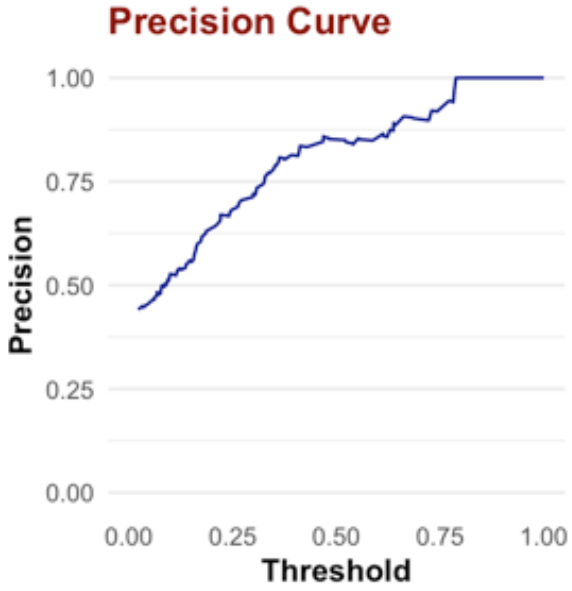
```
## [1] "Logistic Regression Model Log Likelihood: -81.8458138552206"
```

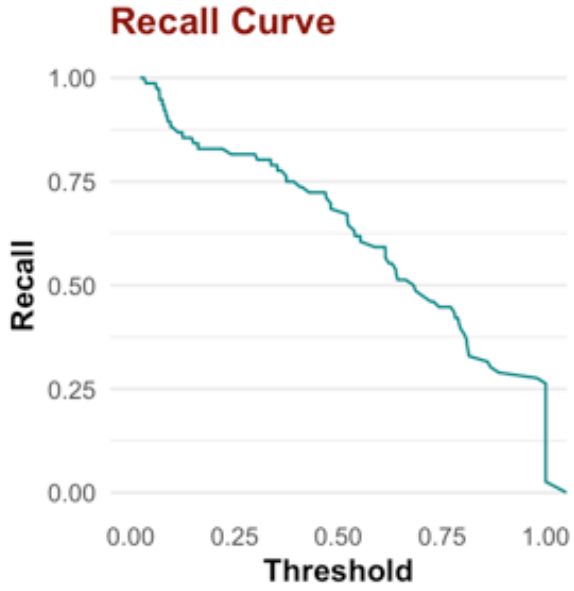
On test set, the log likelihood for logistic regression model also outperforms the null model.

```
evaluate_performance(predict(logreg.model, newdata=dTest, type = 'response'),
                     dTest[,outcome], do.print=TRUE, threshold.value=0.5)
```










```
## Confusion Matrix
##
## |      |  0|  1|
## |:-----|--:|--:|
## |FALSE | 87| 25|
## |TRUE  |  9| 51|
```

Logistic regression classification model evaluations on the test set show promising results that align with no significant difference noted from the training and calibration set above.

1.8 Implications and Further Analysis

In summary, we have run two classification models: Decision Tree and Logistic Regression Classification on the US accident dataset with selected variables from the single variable models to predict whether work is affected or not. After performing model evaluations using various techniques like log likelihood, AUC, ROC, double density plot, precision-recall plots, as well LIME, we have selected the Logistic Regression Classification model as our best model.

Although Logistic Regression performs the best in most performance measures among all models compared in this report, we did observe that the recall, accuracy and f1 score of Logistic Regression could be improved. In such case, we might need to have further discussions with the mining companies to obtain more relevant data and confirm their business needs.

Moreover, do note that there are many more machine learning models out there like random forest, SVM, KNN, Naive Bayes, XGBoost etc. that may perform better than our selected model. We are testing only two models in this project and selected the better one among the two. To further improve our modelling, we could run all the models available, compare their performances and choose the best performing model.

Part 2 - Clustering

In the second part of this project, we will group the US accident dataset into different clusters in terms of the seriousness of work affected (degree_injury) using K-Means Clustering by different numerical features, such as days_restrict, days_lost, worktime_before_accident, tot_exper, job_exper, mine_exper and days_to_resume_work. Our aim of performing clustering is to discover patterns among subsets of the US accident dataset.

2.1 Loading and cleaning the relevant Data

Loading the clustering data with cluster response (degree_injury) and relevant numerical features (days_restrict, days_lost, worktime_before_accident, tot_exper, job_exper, mine_exper and days_to_resume_work, accident_type).

```
cluster_target <- "degree_injury"
accident_clus <- select(accident_clean_v2, cluster_target, "days_restrict", "days_
lost", "worktime_before_accident", "tot_exper", "mine_exper", "job_exper", "days_t
o_resume_work", "accident_type_cd")
```

Checking on the number of NAs noted in the clustering data.

```
apply(is.na(accident_clus), 2, sum)
```

```
##            degree_injury            days_restrict                days_lost
##                       11                        0                        0
## worktime_before_accident               tot_exper              mine_exper
##                      183                      357                      330
##                job_exper      days_to_resume_work         accident_type_cd
##                      325                      317                       20
```

From the NA summary above, we observed that tot_exper, mine_exper, job_exper and days_to_resume_work each having over 300 NAs and they were widely spread in different observations (~ 20% of total observations). After looking into the dataset, we found out that these NAs are mainly due to missing values or empty entry were key-ed in to represent zero.

Since NAs contributed 20% of the total observation, we are unable to omit all of the NA rows directly. To ensure sufficient observations are included in the clustering analysis, we handled the NAs in different ways as follows:

i. tot_exper
   Most of the NAs are due to missing value in the dataset. Tot_exper of injured workers shall be the maximum of their mine_exper and job_exper. Thus, for observations with valid mine_exper and job_exper, we have replaced the missing tot_exper by the maximum of mine_exper and job_exper; otherwise, median of tot_exper was chosen to replace NAs as distribution of tot_exper is right skewed (from Project 1 EDA result).

ii. mine_exper and job_exper
   Most of the NAs are due to missing value in the dataset. Both distribution on mine_exper and job_exper are right-skewed (from Project 1 EDA result), we have then replaced these missing values with median of mine_exper and job_exper respectively as a better estimate.

iii. days_to_resume_work
   Most of the NAs are due to missing return_to_work_dt in the dataset. After looking further into details of these observations, there were no injuries in these accidents. Thus, days_to_resume_work are replaced by 0.

```
tot_exper_notna <- accident_clus$tot_exper[!is.na(accident_clus$tot_exper)]
job_exper_notna <- accident_clus$job_exper[!is.na(accident_clus$job_exper)]
mine_exper_notna <- accident_clus$mine_exper[!is.na(accident_clus$mine_exper)]

accident_clus2 <- within(accident_clus, {
  tot_exper <- ifelse(is.na(tot_exper),
                      ifelse(is.na(ifelse(mine_exper>job_exper, mine_exper, job_ex
per)),
                             median(tot_exper_notna),
                             ifelse(mine_exper>job_exper, mine_exper, job_exper)),
                      tot_exper)
  job_exper <- ifelse(is.na(job_exper), median(job_exper_notna), job_exper)
  mine_exper <- ifelse(is.na(mine_exper), median(mine_exper_notna), mine_exper)
  days_to_resume_work <- as.numeric(days_to_resume_work)
  days_to_resume_work <- ifelse(is.na(days_to_resume_work),
                                ifelse(accident_type_cd == 44, 0, NA), days_to_res
ume_work)
})
```

```
apply(is.na(accident_clus2), 2, sum)
```

```
##            degree_injury            days_restrict                days_lost
##                       11                        0                        0
## worktime_before_accident                tot_exper               mine_exper
##                      183                        0                        0
##                job_exper      days_to_resume_work         accident_type_cd
##                        0                      100                       20
```

Removing the irrelevant variables and NAs from the clustering data.

```
accident_clus3 <- select(accident_clus2,-c("accident_type_cd"))
accident_clus3 <- na.omit(accident_clus3)
nrow(accident_clus3)
```

```
## [1] 1722
```

2.2 Normalisation of numerical features

We noticed that there are different units of measure in the features, such as tot_exper is measured in number of years, while days_restrict, days_lost and days_to_resume_work are measured in number of days. Disparity in units may affect the clustering result because different units cause different distance and potentially different clustering. As such, we transformed all the numerical features to have a mean value of 0 and a standard deviation of 1 using the scale( ) function.

```
vars.to.use <- colnames(accident_clus3)[-1]     # removing clustering response "de
gree_injury"
scaled_accident_clus3 <- scale(accident_clus3[,vars.to.use])  # normalising numeri
cal features
```

Means on every features before normalisation

```
scaled_accident_clus3.center <- attr(scaled_accident_clus3, "scaled:center")
scaled_accident_clus3.center
```

```
##            days_restrict                days_lost worktime_before_accident
##                 6.932636                17.207898                 4.874293
##                tot_exper              mine_exper                job_exper
##                10.387218                 6.297334                 6.388571
##      days_to_resume_work
##                41.259001
```

Variances on every features before normalisation

```
scaled_accident_clus3.scale <- attr(scaled_accident_clus3, "scaled:scale")
scaled_accident_clus3.scale
```

```
##            days_restrict                days_lost worktime_before_accident
##                22.734582                47.084897                 3.175713
##                tot_exper              mine_exper                job_exper
##                10.029846                 8.198578                 7.765276
##      days_to_resume_work
##                82.260907
```

From the tables above, we cab observe that days_restrict, days_lost and days_to_resume_work are having much greater variance than mean. This may indicate the distributions of these features are over-dispersed.

2.3 Clustering with estimated k = 3 under K-Means Clustering algorithm

Since we are using K-Means approach to perform the clustering, we estimate the number of clustering (k) and apply it to the kmeans( ) function. In this stage, we estimate our best k to be 3. Further analysis will be performed and discussed to validate the appropriateness of best k = 3 in later stage of this project.

```
kbest.p <- 3
# run kmeans with 3 clusters, 100 random starts and 100 maximum iterations per run
.
set.seed(2455)
kmClusters <- kmeans(scaled_accident_clus3, kbest.p, nstart=100, iter.max=100)
```

Centres of the 3 clusters

```
kmClusters$centers
```

```
##   days_restrict  days_lost worktime_before_accident  tot_exper  mine_exper
## 1    1.23072689  2.5133265              -0.13114824  0.1133209  0.02013111
## 2   -0.11475216 -0.2351185               0.03985836 -0.4602973 -0.39133483
## 3   -0.03948979 -0.0780944              -0.08638630  1.4709523  1.27631628
##     job_exper days_to_resume_work
## 1 -0.03132586          2.81801450
## 2 -0.40546492         -0.27764757
## 3  1.34002856         -0.04157332
```

Size of the 3 clusters

```
kmClusters$size
```

```
## [1]  126 1223  373
```

Most of the accidents are grouped in Cluster 2.

```
cat("Total of cluster sizes =", sum(kmClusters$size))
```

```
## Total of cluster sizes = 1722
```

```
cat("Total number of observations =", nrow(accident_clus3))
```

```
## Total number of observations = 1722
```

2.4 Finding the best k by CH index and ASW, and validate our estimated k

Finding the best k by using Calinski-Harabasz index (CH index) and average silhouette width (ASW) in kmeansrun( ) over the range of k = 1 to 10

```
kmClustering.ch <- kmeansruns(scaled_accident_clus3, krange=1:10, criterion="ch")
kmClustering.ch$bestk
```

```
## [1] 3
```

```
kmClustering.asw <- kmeansruns(scaled_accident_clus3, krange=1:10, criterion="asw"
)
kmClustering.asw$bestk
```

```
## [1] 4
```

CH index values over the range of k = 1 to 10

```
cat("CH index from kmeans for k=1 to 10:","\n", kmClustering.ch$crit)
```

```
## CH index from kmeans for k=1 to 10:
##  0 559.5368 591.1989 575.8709 566.1517 539.6865 507.6526 482.0619 459.4811 441.
4753
```

From here, we can observe that k=3 has the highest value.

ASW values over the range of k = 1 to 10

```
cat("ASW from kmeans for k=1 to 10:","\n", kmClustering.asw$crit)
```

```
## ASW from kmeans for k=1 to 10:
##  0 0.4082218 0.417868 0.4284939 0.2638949 0.2670843 0.2722832 0.2811742 0.27769
63 0.2856377
```

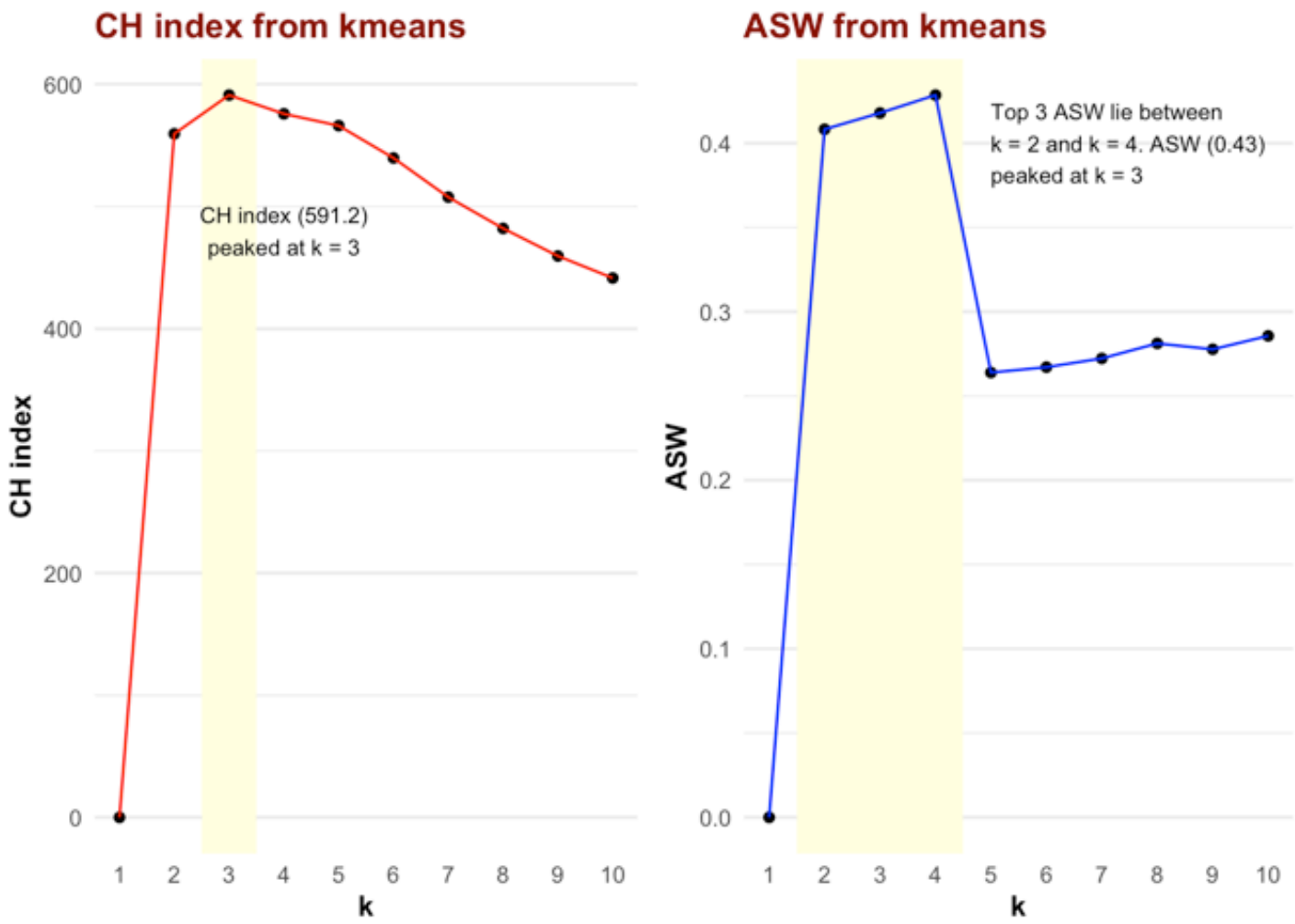
From here, we can observe that k=4 has the highest value.

Visualising CH index and ASW from kmeans for k = 1 to 10

```
kmCritframe <- data.frame(k=1:10, ch=kmClustering.ch$crit,
                          asw=kmClustering.asw$crit)
fig1 <- ggplot(kmCritframe, aes(x=k, y=ch)) +
  geom_rect(data=NULL, aes(xmin=2.5, xmax=3.5, ymin=-Inf, ymax=Inf), fill="lightye
llow") +
  geom_point() + geom_line(colour="red") +
  scale_x_continuous(breaks=1:10, labels=1:10) +
  labs(title = "CH index from kmeans" , y="CH index") + theme(text=element_text(si
ze=20)) + cits4009_theme +
  annotate("text", x = 4, y =480, label = paste("CH index (591.2)", "peaked at k =
3", sep = "\n"), size = 3, hjust = 0.5)

fig2 <- ggplot(kmCritframe, aes(x=k, y=asw)) +
  geom_rect(data=NULL, aes(xmin=1.5, xmax=4.5, ymin=-Inf, ymax=Inf), fill="lightye
llow") +
  geom_point() + geom_line(colour="blue") +
  scale_x_continuous(breaks=1:10, labels=1:10) +
  labs(title = "ASW from kmeans", y="ASW") + theme(text=element_text(size=20)) + c
its4009_theme +
  annotate("text", x = 5, y =0.4, label = paste("Top 3 ASW lie between", "k = 2 an
d k = 4. ASW (0.43)", "peaked at k = 3" , sep = "\n"), size = 3, hjust = 0)

grid.arrange(fig1, fig2, nrow=1)
```



From the analysis above, the CH index gives the best k-value as 3 and ASW gives the best k-value as 4. Thus, 3 or 4 is our optimal number of k for this data set. The ASW at k = 3 is reasonably good, being the top 3 highest ASW among the k range from 1 to 10. Therefore, our choice of 3 clusters is reasonable.

2.5 Visualising clusters in 2 dimensions based on principal component analysis

Computing the first two principal components by cluster and degree_injury

```
groups <- kmClusters$cluster
princ <- prcomp(scaled_accident_clus3) # Calculate the principal components
nComp <- 2 # focus on the first two principal components

# project scaled_accident_clus3 onto the first two principal components
project2D <- as.data.frame(predict(princ, newdata=scaled_accident_clus3)[,1:nComp]
)

# combine with `groups` and accident_clus3$degree_injury to form a 4-column data f
rame
kmean.project2D <- cbind(project2D, cluster=as.factor(groups), degree_injury=accid
ent_clus3$degree_injury)
head(kmean.project2D)
```

PC1	PC2	cluster	degree_injury

	<dbl>	<dbl>	<fct>	<fct>
1	0.09071620	0.041027703	2	DAYS RESTRICTED ACTIVITY ONLY
2	0.15778741	0.062500525	2	NO DYS AWY FRM WRK,NO RSTR ACT
3	0.06704044	0.026142936	2	DAYS AWAY FROM WORK ONLY
4	0.15138569	0.063646700	2	DAYS RESTRICTED ACTIVITY ONLY
5	0.16697106	0.051646565	2	DAYS RESTRICTED ACTIVITY ONLY
6	0.03935272	0.007763027	2	DAYS RESTRICTED ACTIVITY ONLY

6 rows

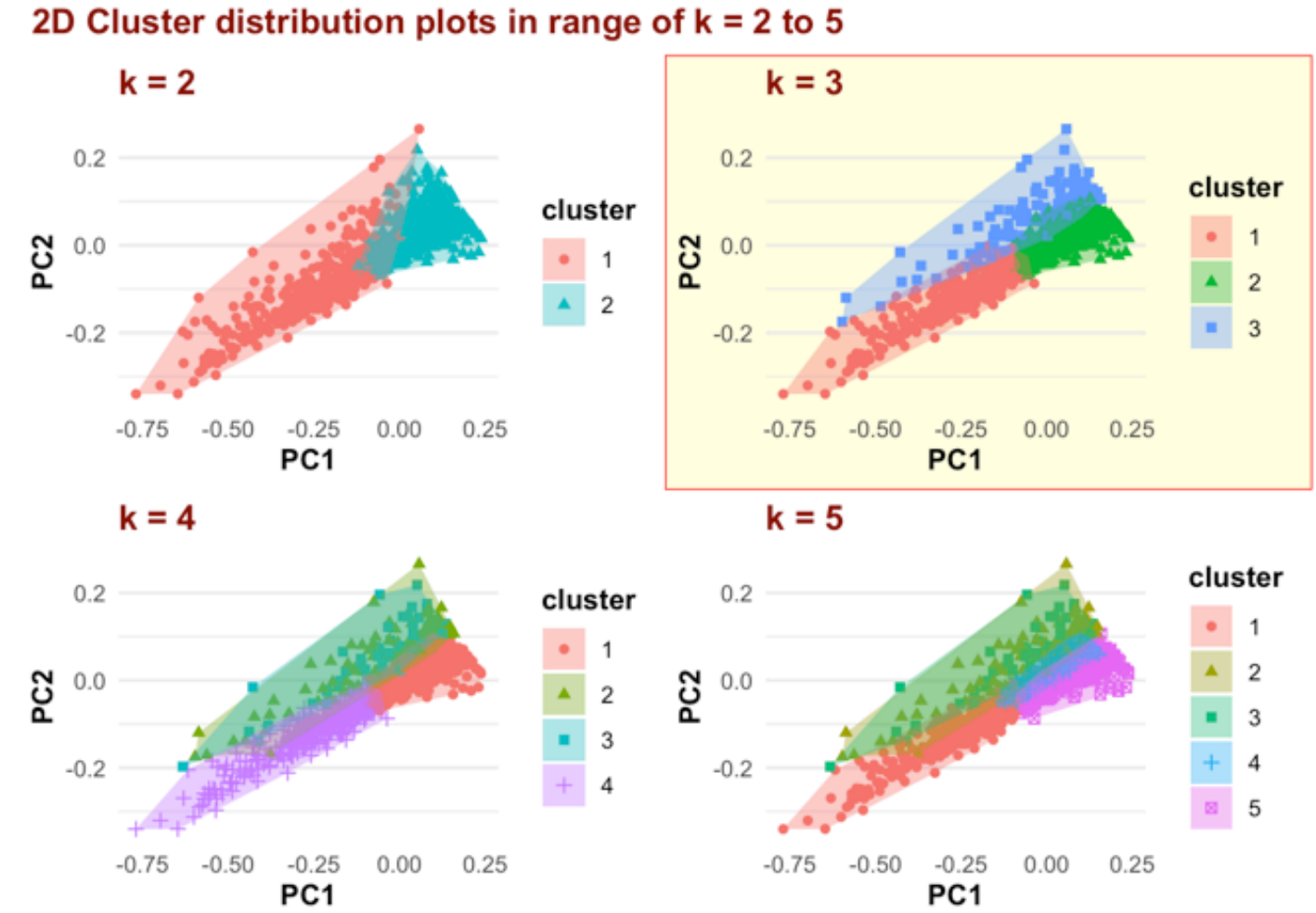
Finding the convex hull

```
find_convex_hull <- function(proj2Ddf, groups) {
  do.call(rbind,
          lapply(unique(groups),
                 FUN = function(c) {
                    f <- subset(proj2Ddf, cluster==c);
                    f[chull(f),]
                  }
                )
          )
}
kmean.hull <- find_convex_hull(kmean.project2D, groups)
```

Visualising the clusters using ggplot

```
fig <- c()
kvalues <- seq(2,5)
for (k in kvalues) {
  groups <- kmeans(scaled_accident_clus3, k, nstart=100, iter.max=100)$cluster
  kmclust.project2D <- cbind(project2D, cluster=as.factor(groups),
                             degree_injury = accident_clus3$degree_injury)
  kmclust.hull <- find_convex_hull(kmclust.project2D, groups)
  assign(paste0("fig", k),
         ggplot(kmclust.project2D, aes(x=PC1, y=PC2)) +
           geom_point(aes(shape=cluster, color=cluster)) +
           geom_polygon(data=kmclust.hull, aes(group=cluster, fill=cluster), alpha
=0.4, linetype=0) +
           labs(title = sprintf("k = %d", k)) +
           theme(legend.position="none", text=element_text(size=20)) + cits4009_th
eme
         )
}
fig3 <- fig3 + theme(panel.background = element_rect(fill = 'lightyellow', color =
"lightyellow"), plot.background = element_rect(fill="lightyellow", color = "red"))

all_plots <- (fig2 + fig3) / (fig4 + fig5) + plot_annotation(title = "2D Cluster d
istribution plots in range of k = 2 to 5") &
  theme(plot.title = element_text(color = "darkred", face="bold"))
all_plots
```



Our choice of 3 clusters is highlighted in the plot above. It shows a somewhat clear separation with these three clusters being the same even as k increases.

2.6 Assessing clusters with bootstrapping

```
cboot.kmean <- clusterboot(scaled_accident_clus3, clustermethod=kmeansCBI, k=kbest.p, seed = 2455)
```

```
summary(cboot.kmean$result)
```

```
##                Length Class  Mode
## result           11   kmeans list
## nc                1   -none- numeric
## clusterlist       3   -none- list
## partition      1722   -none- numeric
## clustermethod     1   -none- character
## nccl              1   -none- numeric
```

Calculating the stability of clusters

```
Stability <- 1 - cboot.kmean$bootbrd/100
Clusters <- c(1:kbest.p)
Eval <- cbind(Clusters, Stability)
Eval
```

```
##      Clusters Stability
## [1,]        1      0.99
## [2,]        2      0.97
## [3,]        3      0.84
```

From the mean values of Jaccard coefficient over 100 bootstrap iterations, all 3 clusters are above 0.80 and considered to be highly stable.

2.7 Summary of clusters

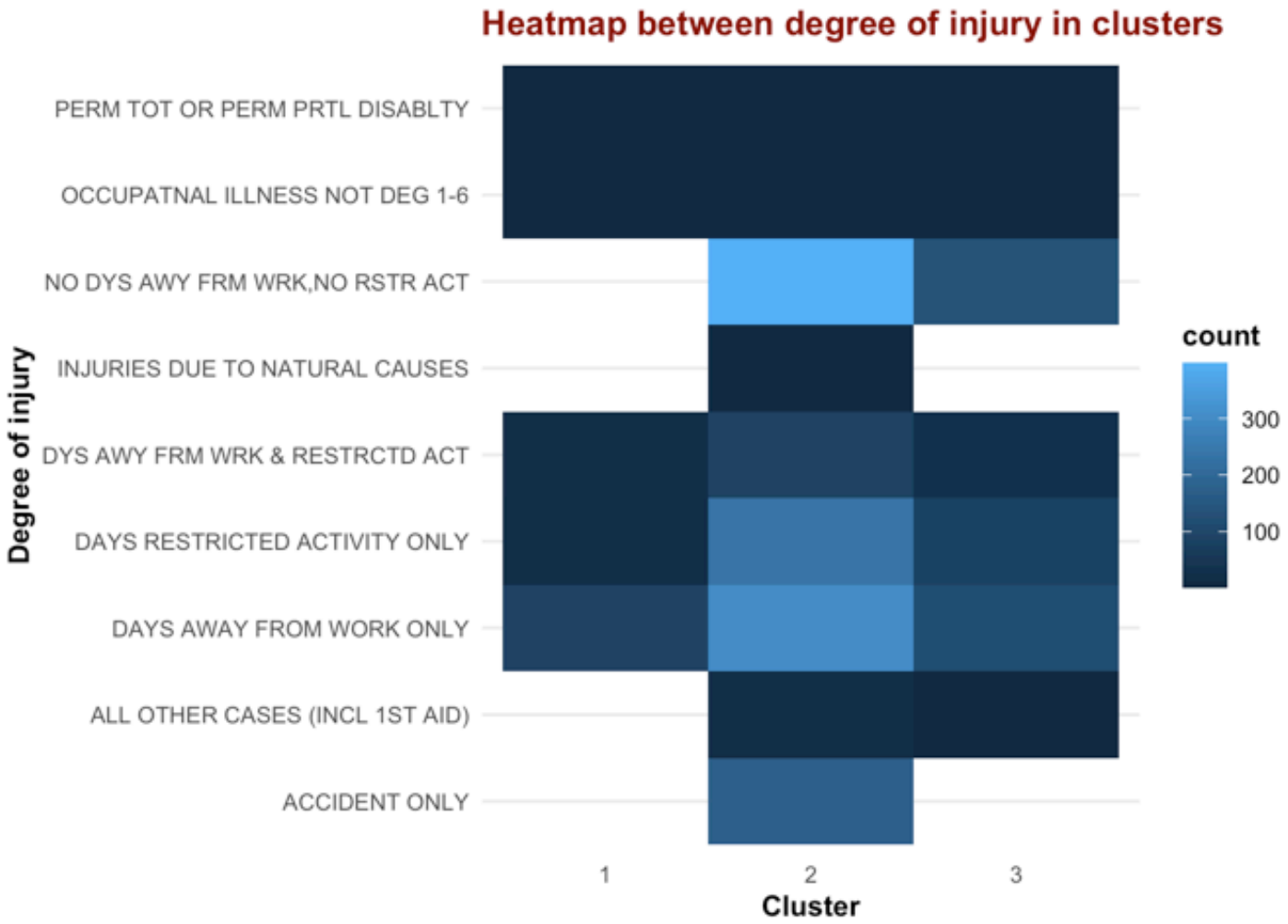
Print out the cluster for investigation

```
groups.cboot <- cboot.kmean$result$partition

print_clusters <- function(df, groups, cols_to_print){
  Ngroups <- max(groups)
  for (i in 1:Ngroups){
    print(paste("cluster", i))
    print(cbind(Cluster = paste("Cluster",i), df[groups == i, cols_to_print]))
  }
}
cols_to_print <- colnames(accident_clus3)
print_clusters(accident_clus3, groups.cboot, cols_to_print)
```

Summarising degree_injury by clusters

```
cluster_df <- kmean.project2D[,3:4]
cluster_df_gp <- cluster_df %>% count(cluster, degree_injury, sort = TRUE)
cluster_df_gp <- cluster_df_gp %>% arrange(cluster, -n)
colnames(cluster_df_gp) <- c("cluster", "degree_injury", "count")
ggplot(data = cluster_df_gp, aes(x = degree_injury, y = cluster, fill = count)) +
  geom_tile() + labs(title = "Heatmap between degree of injury in clusters", x = "
Degree of injury", y = "Cluster") + coord_flip() + cits4009_theme
```

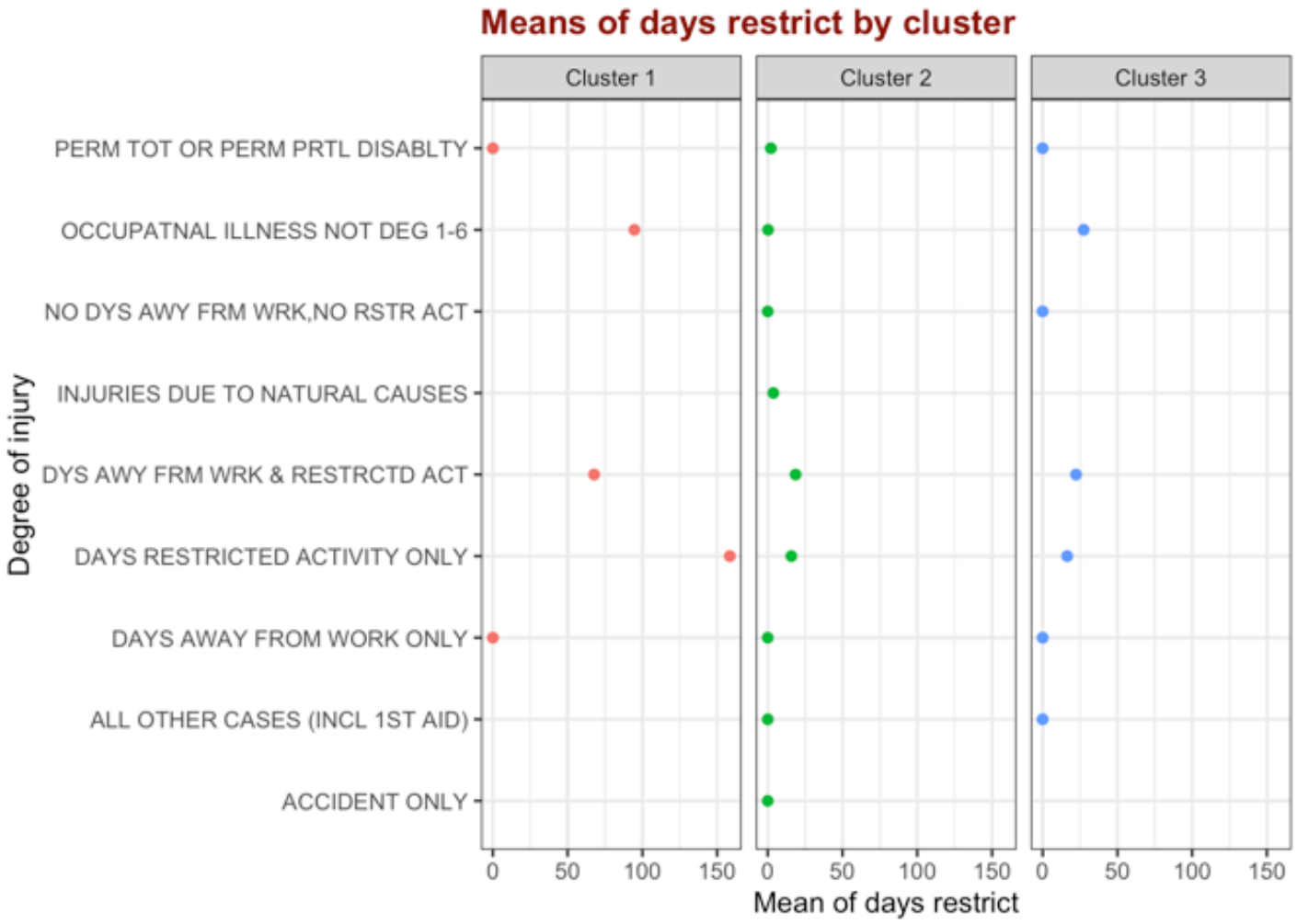


From the chart above, we can see most of the degree_injury level are widely spread to every cluster. Cluster 2 contains the largest number of accidents, with majority of it being accident with no work affected (i.e. “ACCIDENT ONLY”, “NO DYS AWY FRM WRK, NO RSTR ACT”). As for Cluster 1, majority of the accidents are causing some work being affected (i.e. “DAY AWY FRM WRK & RESTRCTD ACT”, “DAYS RESTRICTED ACTIVITY ONLY” and “DAYS AWAY FROM WORK ONLY”).

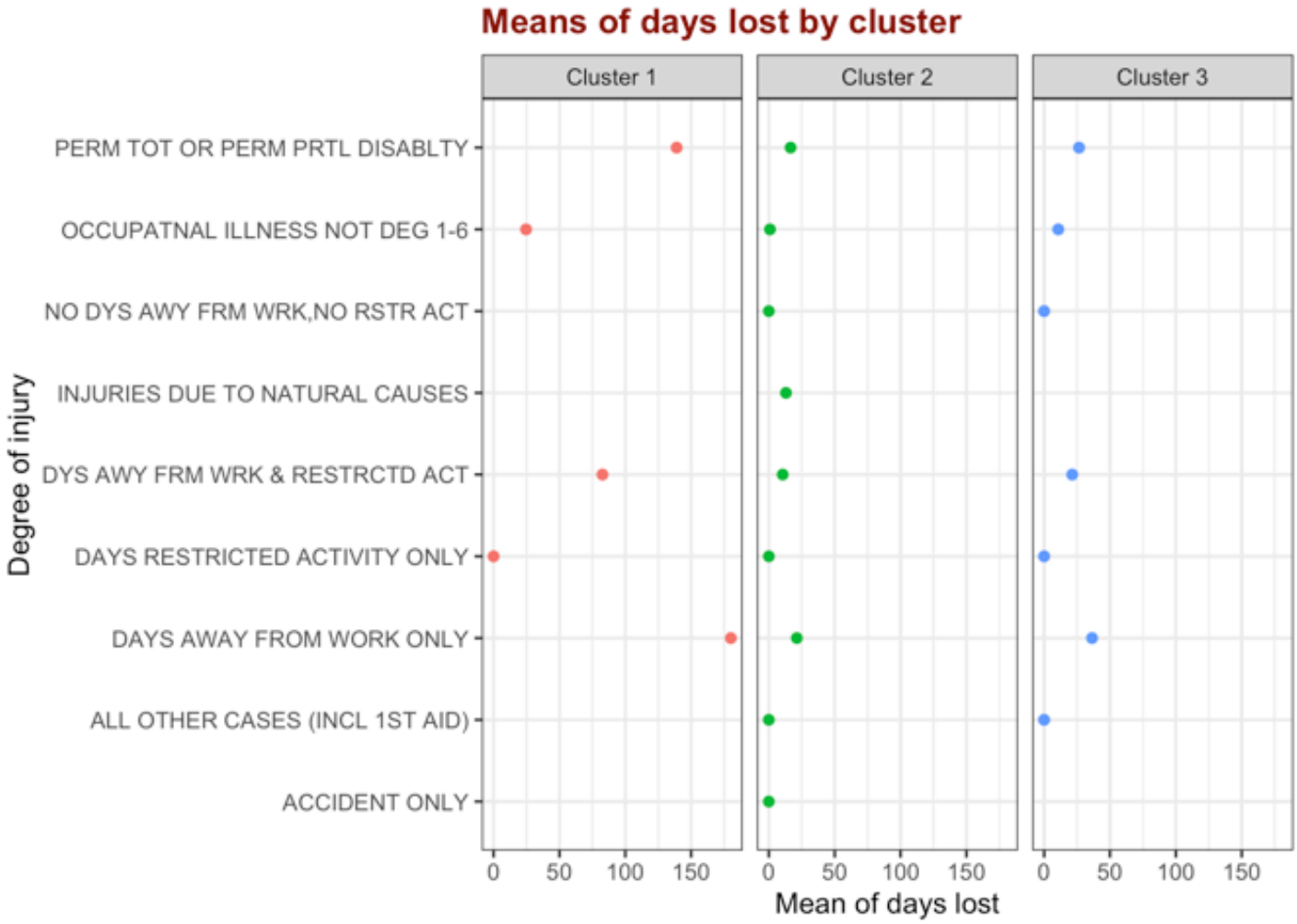
Summarising numerical features by degree of injury by clusters

```
df_clus1 <- cbind(Cluster = "Cluster 1", accident_clus3[groups.cboot == 3, cols_to
_print])
df_clus2 <- cbind(Cluster = "Cluster 2", accident_clus3[groups.cboot == 2, cols_to
_print])
df_clus3 <- cbind(Cluster = "Cluster 3", accident_clus3[groups.cboot == 1, cols_to
_print])
df_clusgroup <- rbind(df_clus1, df_clus2, df_clus3)
# Summarising the every level of degree_injury by the mean of all numerical featur
es in each cluster
df_clusgroup <- df_clusgroup %>%
  group_by(Cluster, degree_injury) %>%
  summarise(across(everything(), list(mean)))
```

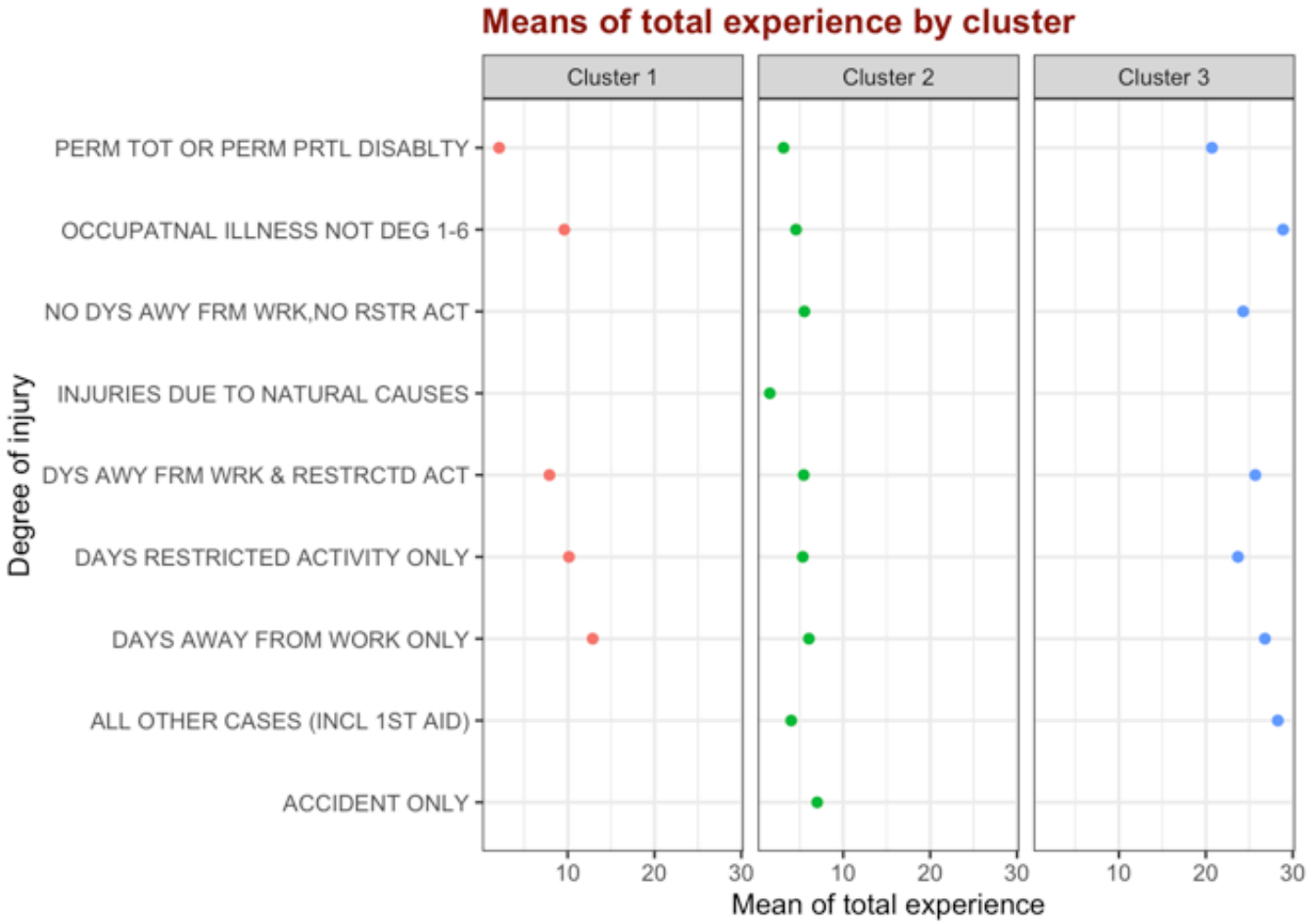
```
ggplot(df_clusgroup) + geom_point(aes(x = degree_injury, y = days_restrict_1, colo
r = Cluster)) + facet_wrap(~ Cluster, nrow = 1) + coord_flip() + labs(x = "Degree
of injury", y = "Mean of days restrict", title = "Means of days restrict by cluste
r") + theme_bw() + theme(plot.title = element_text(color = "darkred", face="bold")
, legend.position = "none")
```



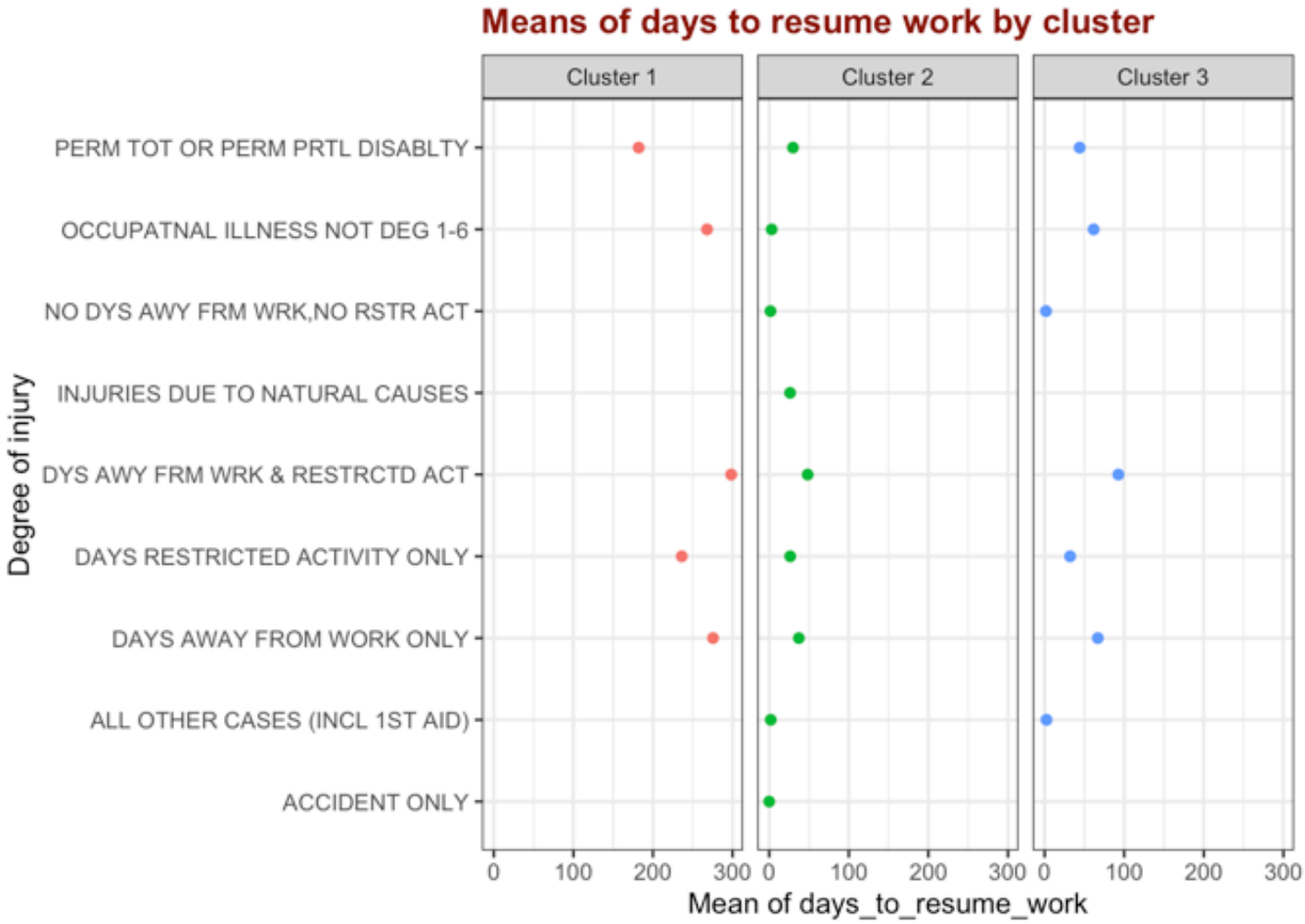

```
ggplot(df_clusgroup) + geom_point(aes(x = degree_injury, y = days_lost_1, color =
Cluster)) + facet_wrap(~ Cluster, nrow = 1) + coord_flip() + labs(x = "Degree of i
njury", y = "Mean of days lost", title = "Means of days lost by cluster") + theme_
bw() + theme(plot.title = element_text(color = "darkred", face="bold"), legend.pos
ition = "none")
```




```
ggplot(df_clusgroup) + geom_point(aes(x = degree_injury, y = tot_exper_1, color =
Cluster)) + facet_wrap(~ Cluster, nrow = 1) + coord_flip() + labs(x = "Degree of i
njury", y = "Mean of total experience", title = "Means of total experience by clus
ter") + theme_bw() + theme(plot.title = element_text(color = "darkred", face="bold
"), legend.position = "none")
```




```
ggplot(df_clusgroup) + geom_point(aes(x = degree_injury, y = days_to_resume_work_1
, color = Cluster)) + facet_wrap(~ Cluster, nrow = 1) + coord_flip() + labs(x = "D
egree of injury", y = "Mean of days_to_resume_work", title = "Means of days to res
ume work by cluster") + theme_bw() + theme(plot.title = element_text(color = "dark
red", face="bold"), legend.position = "none")
```



From the plots above, we can observe some patterns on the days affected and work experience for different cluster group. The main observations are as follow:

- Cluster 1 contains accidents with the most number of days restrict, days lost and days to resume work, with injured workers having moderate working experience.
- Cluster 2 contains accidents with the least number of days restrict, days lost and days to resume work, with injured workers having least experience.
- Cluster 3 contains accidents with injured workers having higher working experience.

2.8 Implications and Further Analysis

Under the project 2 scope, we are only required to perform one clustering algorithm. However, there are many other clustering algorithms that handle all kinds of unique data and may even outperform K-Means clustering algorithm. Hence, a possible next step would be to include other clustering algorithms such as hierarchical clustering algorithm, centroid-based clustering algorithmn, DBSCAN clustering algorithm, etc. and compare their performances against K-Means Clustering algorithm.

Moreover, we have also considered to cluster our dataset by categorical variables. However, since the categorical variables in the US accident dataset contain too many unique levels, it is very challenging to discover similarities among subsets of the dataset and obtain stable clusters. Therefore, we choose to focus our clustering analysis with numerical features using K-Means algorithm in this project.

Conclusions

In conclusion, we have performed both supervised and unsupervised machine learning techniques in this project. In the first part of this project, we used two classification models: (i) Decision Tree and (ii) Logistic Regression classification to predict whether a mining accident will result in work being affected or not. In the second part of this project, we implemented K-Means Clustering algorithm to group accidents into clusters. We were able to segregate accidents with similar traits and assign them into cluster.

We have successfully achieved our aim of using past data to help mining companies to predict whether accidents will result in work being affected or not, and cluster them into different groups that may need separate strategy to tackle the cause of accidents. This will eventually help mining companies to make better decisions and act fast on every mining accidents that may happen in the future. In the long run, this will definitely help mining companies to achieve higher work efficiency, smoother work progress and reduce expenditure.