CKME\_136 Predictive Analysis on US Traffic Fatalities Project in R

Ebun Odeniyi

rm(list = ls())

# Nesessary packages:  
library(plyr)  
library(ggplot2)  
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:plyr':  
##   
## arrange, count, desc, failwith, id, mutate, rename, summarise,  
## summarize

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(randomForest)

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':  
##   
## combine

## The following object is masked from 'package:ggplot2':  
##   
## margin

library(caret)

## Loading required package: lattice

library(ROSE)

## Loaded ROSE 0.0-3

library(e1071)  
library(nnet)

# Importing the data  
faux <- read.csv("C:/Users/YENN/Desktop/UST/FARS2016N/accident2016.csv", header = T, stringsAsFactors = F)

# Labelling missing values (9, 99, 999, 9999, 99999) as NAs

faux$COUNTY[faux$COUNTY == 999] <- NA  
faux$CITY[faux$CITY == 9999] <- NA  
faux$HOUR[faux$HOUR == 99] <- NA  
faux$MINUTE[faux$MINUTE == 999] <- NA  
faux$NHS[faux$NHS == 9] <- NA  
faux$RUR\_URB[faux$RUR\_URB == 9] <- NA  
faux$FUNC\_SYS[faux$FUNC\_SYS == 99] <- NA  
faux$RD\_OWNER[faux$RD\_OWNER == 99] <- NA  
faux$MILEPT[faux$MILEPT == 99999] <- NA  
faux$SP\_JUR[faux$SP\_JUR == 9] <- NA  
faux$MAN\_COLL[faux$MANINT == 99] <- NA  
faux$REL\_ROAD[faux$REL\_ROAD == 99] <- NA  
faux$WEATHER1[faux$WEATHER1 == 99] <- NA  
faux$WEATHER2[faux$WEATHER2 == 99] <- NA  
faux$WEATHER[faux$WEATHER == 99] <- NA  
faux$CF1[faux$CF1 == 99] <- NA  
faux$CF2[faux$CF2\_COLL == 99] <- NA  
faux$RELJCT1[faux$RELJCT1 == 99] <- NA  
faux$RELJCT2[faux$RELJCT2 == 99] <- NA  
faux$TYP\_INT[faux$TYP\_ == 99] <- NA  
faux$CF3[faux$CF3 == 99] <- NA  
faux$FATALS[faux$FATALS == 9] <- NA

# 

faux <- na.omit(faux)

# Remove TWAY\_ID2 attribute, the only variable with missing values: <sum(is.na (accs$TWAY\_ID2))> and TWAY\_ID, not appropiate for the research project

# Remove YEAR, MONTH, DAY, HOUR, MINUTE attributes - it’s been merged into Timestamps 12:14

# Remove WEATHER1, WEATHER2 attributes, are duplicate of the original WEATHER

# Remove RAIL attribute, no relevant to the research

## accs2016 <- accs[,-c(1:2,10:11,12:14,16:17,23:24,37:38,41)] ## fraud16[,-c(1:2,10:14,16:17,23:24,37:38,41,53)]

datafaux <- faux[,-c(1:2,10:14,16:17,23:24,37:38,41)]

table(datafaux$FATALS)

##   
## 1 2 3 4 5 6   
## 30566 1945 303 78 17 7

# Factorising "Class" and replacing 1 and 0 to "Yes", "No" respectively and making the response variable as factor.   
  
datafaux$FATALS[datafaux$FATALS==1]<-"No"  
datafaux$FATALS[datafaux$FATALS==2]<-"Yes"  
datafaux$FATALS[datafaux$FATALS==3]<-"Yes"  
datafaux$FATALS[datafaux$FATALS==4]<-"Yes"  
datafaux$FATALS[datafaux$FATALS==5]<-"Yes"  
datafaux$FATALS[datafaux$FATALS==6]<-"Yes"  
  
datafaux$FATALS<-as.factor(datafaux$FATALS)

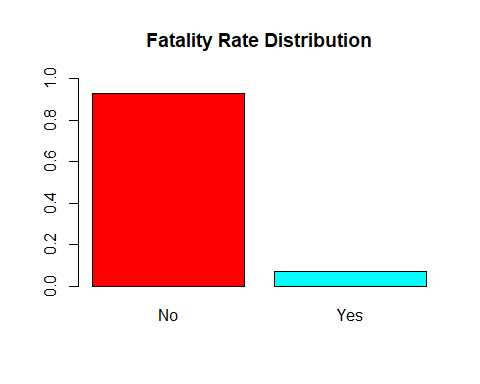
table(datafaux$FATALS)

##   
## No Yes   
## 30566 2350

prop.table(table(datafaux$FATALS))

##   
## No Yes   
## 0.92860615 0.07139385

barplot(prop.table(table(datafaux$FATALS)),  
 col = rainbow(2),  
 ylim = c(0,1),  
 main = "Fatality Rate Distribution")



### Data partition

set.seed(123)  
index <- sample(2, nrow(datafaux), replace = TRUE, prob = c(0.7, 0.3))  
  
train <- datafaux[index==1,]  
test <- datafaux[index==2,]

table(train$FATALS)

##   
## No Yes   
## 21452 1693

prop.table(table(datafaux$FATALS))

##   
## No Yes   
## 0.92860615 0.07139385

summary(train)

## VE\_TOTAL VE\_FORMS PVH\_INVL PEDS   
## Min. : 1.000 Min. : 1.000 Min. : 0.00000 Min. :0.0000   
## 1st Qu.: 1.000 1st Qu.: 1.000 1st Qu.: 0.00000 1st Qu.:0.0000   
## Median : 1.000 Median : 1.000 Median : 0.00000 Median :0.0000   
## Mean : 1.564 Mean : 1.523 Mean : 0.04161 Mean :0.2175   
## 3rd Qu.: 2.000 3rd Qu.: 2.000 3rd Qu.: 0.00000 3rd Qu.:0.0000   
## Max. :64.000 Max. :64.000 Max. :11.00000 Max. :9.0000   
## PERNOTMVIT PERMVIT PERSONS DAY\_WEEK   
## Min. :0.0000 Min. : 0.00 Min. : 0.000 Min. :1.000   
## 1st Qu.:0.0000 1st Qu.: 1.00 1st Qu.: 1.000 1st Qu.:2.000   
## Median :0.0000 Median : 2.00 Median : 2.000 Median :4.000   
## Mean :0.2289 Mean : 2.27 Mean : 2.281 Mean :4.126   
## 3rd Qu.:0.0000 3rd Qu.: 3.00 3rd Qu.: 3.000 3rd Qu.:6.000   
## Max. :9.0000 Max. :120.00 Max. :120.000 Max. :7.000   
## NHS RUR\_URB FUNC\_SYS RD\_OWNER   
## Min. :0.0000 Min. :1.000 Min. : 1.00 Min. : 1.00   
## 1st Qu.:0.0000 1st Qu.:1.000 1st Qu.: 3.00 1st Qu.: 1.00   
## Median :0.0000 Median :2.000 Median : 4.00 Median : 1.00   
## Mean :0.3845 Mean :1.694 Mean : 6.74 Mean :16.16   
## 3rd Qu.:1.0000 3rd Qu.:2.000 3rd Qu.: 5.00 3rd Qu.: 4.00   
## Max. :1.0000 Max. :8.000 Max. :98.00 Max. :98.00   
## ROUTE MILEPT LATITUDE LONGITUD   
## Min. :1.000 Min. : 0 Min. : 19.38 Min. :-174.20   
## 1st Qu.:2.000 1st Qu.: 0 1st Qu.: 32.86 1st Qu.: -98.13   
## Median :3.000 Median : 54 Median : 36.20 Median : -88.04   
## Mean :3.579 Mean :14013 Mean : 36.68 Mean : -88.14   
## 3rd Qu.:5.000 3rd Qu.: 360 3rd Qu.: 40.45 3rd Qu.: -81.62   
## Max. :9.000 Max. :99998 Max. :100.00 Max. :1000.00   
## SP\_JUR HARM\_EV MAN\_COLL RELJCT1   
## Min. :0.00000 Min. : 1.00 Min. : 0.000 Min. :0.00000   
## 1st Qu.:0.00000 1st Qu.: 8.00 1st Qu.: 0.000 1st Qu.:0.00000   
## Median :0.00000 Median :12.00 Median : 0.000 Median :0.00000   
## Mean :0.02312 Mean :17.82 Mean : 1.812 Mean :0.04446   
## 3rd Qu.:0.00000 3rd Qu.:30.00 3rd Qu.: 2.000 3rd Qu.:0.00000   
## Max. :8.00000 Max. :99.00 Max. :99.000 Max. :9.00000   
## RELJCT2 TYP\_INT WRK\_ZONE REL\_ROAD   
## Min. : 1.000 Min. : 1.0 Min. :0.00000 Min. : 1.000   
## 1st Qu.: 1.000 1st Qu.: 1.0 1st Qu.:0.00000 1st Qu.: 1.000   
## Median : 1.000 Median : 1.0 Median :0.00000 Median : 1.000   
## Mean : 2.036 Mean : 1.4 Mean :0.03789 Mean : 2.197   
## 3rd Qu.: 2.000 3rd Qu.: 1.0 3rd Qu.:0.00000 3rd Qu.: 4.000   
## Max. :98.000 Max. :98.0 Max. :4.00000 Max. :98.000   
## LGT\_COND WEATHER SCH\_BUS NOT\_HOUR   
## Min. :1.000 Min. : 1.00 Min. :0.000000 Min. : 0.00   
## 1st Qu.:1.000 1st Qu.: 1.00 1st Qu.:0.000000 1st Qu.:14.00   
## Median :2.000 Median : 1.00 Median :0.000000 Median :88.00   
## Mean :1.864 Mean : 7.12 Mean :0.002938 Mean :57.11   
## 3rd Qu.:2.000 3rd Qu.: 2.00 3rd Qu.:0.000000 3rd Qu.:99.00   
## Max. :9.000 Max. :98.00 Max. :1.000000 Max. :99.00   
## NOT\_MIN ARR\_HOUR ARR\_MIN HOSP\_HR   
## Min. : 0.00 Min. : 0.00 Min. : 0.00 Min. : 0.00   
## 1st Qu.:31.00 1st Qu.:15.00 1st Qu.:33.00 1st Qu.:22.00   
## Median :88.00 Median :99.00 Median :98.00 Median :88.00   
## Mean :65.09 Mean :59.31 Mean :66.93 Mean :72.09   
## 3rd Qu.:99.00 3rd Qu.:99.00 3rd Qu.:99.00 3rd Qu.:99.00   
## Max. :99.00 Max. :99.00 Max. :99.00 Max. :99.00   
## HOSP\_MN CF1 CF2 CF3   
## Min. : 0.00 Min. : 0.00 Min. : 0.0000 Min. : 0.00000   
## 1st Qu.:55.00 1st Qu.: 0.00 1st Qu.: 0.0000 1st Qu.: 0.00000   
## Median :88.00 Median : 0.00 Median : 0.0000 Median : 0.00000   
## Mean :76.41 Mean : 1.12 Mean : 0.1139 Mean : 0.01607   
## 3rd Qu.:99.00 3rd Qu.: 0.00 3rd Qu.: 0.0000 3rd Qu.: 0.00000   
## Max. :99.00 Max. :28.00 Max. :28.0000 Max. :27.00000   
## FATALS DRUNK\_DR   
## No :21452 Min. :0.0000   
## Yes: 1693 1st Qu.:0.0000   
## Median :0.0000   
## Mean :0.2604   
## 3rd Qu.:1.0000   
## Max. :3.0000

# ’ #Predictive Model (Random Forest)

#library(randomForest)  
rftrain <- randomForest(FATALS~., data = train)

# Predictive Model Evaluation with test data

#library(caret)  
#library(e1071)

confusionMatrix(predict(rftrain, test), test$FATALS, positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 9097 630  
## Yes 17 27  
##   
## Accuracy : 0.9338   
## 95% CI : (0.9287, 0.9386)  
## No Information Rate : 0.9328   
## P-Value [Acc > NIR] : 0.3524   
##   
## Kappa : 0.0692   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.041096   
## Specificity : 0.998135   
## Pos Pred Value : 0.613636   
## Neg Pred Value : 0.935232   
## Prevalence : 0.067240   
## Detection Rate : 0.002763   
## Detection Prevalence : 0.004503   
## Balanced Accuracy : 0.519615   
##   
## 'Positive' Class : Yes   
##

confusionMatrix(predict(rftrain, test), test$FATALS, positive = "No")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 9097 630  
## Yes 17 27  
##   
## Accuracy : 0.9338   
## 95% CI : (0.9287, 0.9386)  
## No Information Rate : 0.9328   
## P-Value [Acc > NIR] : 0.3524   
##   
## Kappa : 0.0692   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.9981   
## Specificity : 0.0411   
## Pos Pred Value : 0.9352   
## Neg Pred Value : 0.6136   
## Prevalence : 0.9328   
## Detection Rate : 0.9310   
## Detection Prevalence : 0.9955   
## Balanced Accuracy : 0.5196   
##   
## 'Positive' Class : No   
##

table (train$FATALS)

##   
## No Yes   
## 21452 1693

prop.table(table(train$FATALS))

##   
## No Yes   
## 0.92685245 0.07314755

# undersampling for better specificity: reduce observations from the majority class then you use undersampling

# Underampling  
#library(ROSE)  
datafaux\_under <- ovun.sample(FATALS ~ ., data = train, method = "under", N = 3386, seed = 1)$data  
table(datafaux\_under$FATALS)

##   
## No Yes   
## 1693 1693

# Oversampling for better sensitivity: increase observations from the minority class then you use oversampling

# Oversampling  
datafaux\_over <- ovun.sample(FATALS ~ ., data = train, method = "over",N = 42904, seed = 1)$data  
table(datafaux\_over$FATALS)

##   
## No Yes   
## 21452 21452

# Both  
datafaux\_both <- ovun.sample(FATALS ~ ., data = train, method = "both", p=0.5, N=23145, seed = 1)$data  
table(datafaux\_both$FATALS)

##   
## No Yes   
## 11651 11494

# renaming datafile

datafaux7<- datafaux\_both

# Just checking out this:

#write.table(datafaux7, file = "C:/Users/YENN/Desktop/Dataset/accid\_cleaned5.csv")

##### Using FSelector

# Using FSelector

All variables

library(FSelector)  
weights\_1 <- information.gain(FATALS~., data = datafaux7)  
row.names(weights\_1)[order(weights\_1, decreasing = TRUE)]

## [1] "PERSONS" "PERMVIT" "PEDS" "PERNOTMVIT" "HARM\_EV"   
## [6] "MAN\_COLL" "VE\_FORMS" "VE\_TOTAL" "RUR\_URB" "MILEPT"   
## [11] "ROUTE" "DRUNK\_DR" "RD\_OWNER" "HOSP\_MN" "HOSP\_HR"   
## [16] "LATITUDE" "FUNC\_SYS" "NHS" "REL\_ROAD" "LONGITUD"   
## [21] "DAY\_WEEK" "RELJCT2" "CF1" "WEATHER" "LGT\_COND"   
## [26] "SP\_JUR" "TYP\_INT" "NOT\_HOUR" "PVH\_INVL" "RELJCT1"   
## [31] "WRK\_ZONE" "SCH\_BUS" "NOT\_MIN" "ARR\_HOUR" "ARR\_MIN"   
## [36] "CF2" "CF3"

print(weights\_1)

## attr\_importance  
## VE\_TOTAL 0.0354177842  
## VE\_FORMS 0.0355089545  
## PVH\_INVL 0.0000000000  
## PEDS 0.0692734934  
## PERNOTMVIT 0.0641819055  
## PERMVIT 0.1690838171  
## PERSONS 0.1701277499  
## DAY\_WEEK 0.0023006364  
## NHS 0.0036682668  
## RUR\_URB 0.0084732400  
## FUNC\_SYS 0.0044509817  
## RD\_OWNER 0.0066593839  
## ROUTE 0.0081081489  
## MILEPT 0.0084704192  
## LATITUDE 0.0047067325  
## LONGITUD 0.0029418799  
## SP\_JUR 0.0010514712  
## HARM\_EV 0.0600327024  
## MAN\_COLL 0.0415616871  
## RELJCT1 0.0000000000  
## RELJCT2 0.0018388541  
## TYP\_INT 0.0007000392  
## WRK\_ZONE 0.0000000000  
## REL\_ROAD 0.0033251836  
## LGT\_COND 0.0013708657  
## WEATHER 0.0015375287  
## SCH\_BUS 0.0000000000  
## NOT\_HOUR 0.0006174904  
## NOT\_MIN 0.0000000000  
## ARR\_HOUR 0.0000000000  
## ARR\_MIN 0.0000000000  
## HOSP\_HR 0.0049912556  
## HOSP\_MN 0.0054408985  
## CF1 0.0017303402  
## CF2 0.0000000000  
## CF3 0.0000000000  
## DRUNK\_DR 0.0069507694

## Select top 15 variables  
subset\_15 <- cutoff.k(weights\_1, 15)  
subset\_15

## [1] "PERSONS" "PERMVIT" "PEDS" "PERNOTMVIT" "HARM\_EV"   
## [6] "MAN\_COLL" "VE\_FORMS" "VE\_TOTAL" "RUR\_URB" "MILEPT"   
## [11] "ROUTE" "DRUNK\_DR" "RD\_OWNER" "HOSP\_MN" "HOSP\_HR"

f15 <- as.simple.formula(subset\_15, "FATALS")  
#row.names(f)[order(f, decreasing = TRUE)]  
print(f15)

## FATALS ~ PERSONS + PERMVIT + PEDS + PERNOTMVIT + HARM\_EV + MAN\_COLL +   
## VE\_FORMS + VE\_TOTAL + RUR\_URB + MILEPT + ROUTE + DRUNK\_DR +   
## RD\_OWNER + HOSP\_MN + HOSP\_HR  
## <environment: 0x0000000022f2a218>

## Select top 10 variables  
subset\_10 <- cutoff.k(weights\_1, 10)  
subset\_10

## [1] "PERSONS" "PERMVIT" "PEDS" "PERNOTMVIT" "HARM\_EV"   
## [6] "MAN\_COLL" "VE\_FORMS" "VE\_TOTAL" "RUR\_URB" "MILEPT"

f10 <- as.simple.formula(subset\_10, "FATALS")  
#row.names(f)[order(f, decreasing = TRUE)]  
print(f10)

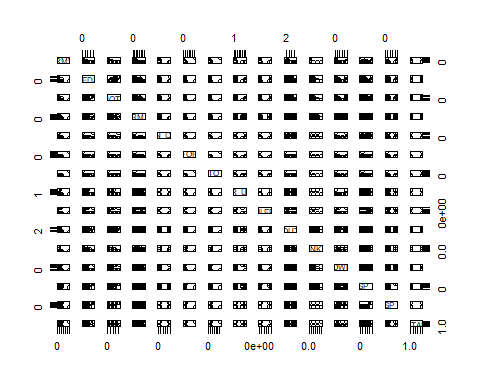
## FATALS ~ PERSONS + PERMVIT + PEDS + PERNOTMVIT + HARM\_EV + MAN\_COLL +   
## VE\_FORMS + VE\_TOTAL + RUR\_URB + MILEPT  
## <environment: 0x0000000022098e60>

# USing the selected features (PERSONS+PERMVIT+PEDS+PERNOTMVIT+HARM\_EV+MAN\_COLL+LATITUDE+VE\_TOTAL+VE\_FORMS+MILEPT+ROUTE+RUR\_URB+RD\_OWNER+DRUNK\_DR)select(datafaux7, PERSONS, PERMVIT, PEDS, PERNOTMVIT, HARM\_EV, MAN\_COLL, LATITUDE, VE\_TOTAL, VE\_FORMS, MILEPT, ROUTE, RUR\_URB, RD\_OWNER, DRUNK\_DR, FATALS)

data77 <- select(datafaux7, PERMVIT, PEDS, PERNOTMVIT, HARM\_EV, MAN\_COLL, VE\_FORMS, VE\_TOTAL, RUR\_URB, MILEPT, ROUTE,DRUNK\_DR, RD\_OWNER, HOSP\_MN, HOSP\_HR, FATALS)  
#data77

#cor(data77)

plot(data77)



library(corrplot)

## corrplot 0.84 loaded

#corrplot(cor(data77))  
#corrplot(cor(data77), method = c("number"))

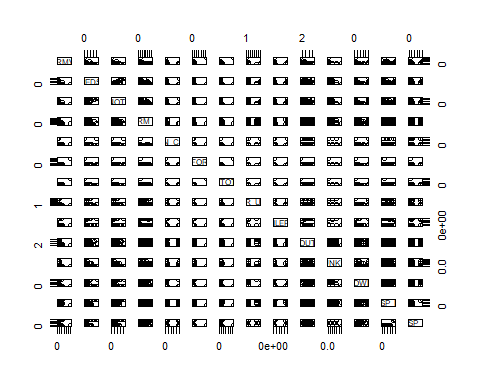
What is the correlation between the attributes other than FATALS variable?

# Remove FATALS

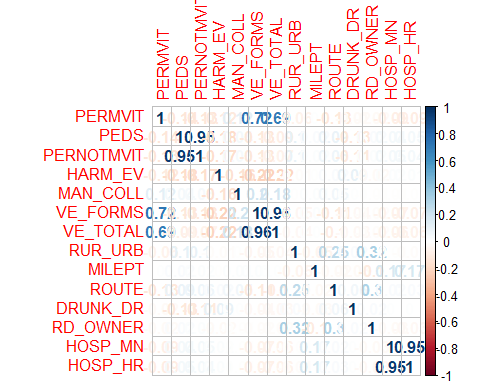
data77\_f <- data77[,-c(15)]  
  
cor(data77\_f)

## PERMVIT PEDS PERNOTMVIT HARM\_EV MAN\_COLL  
## PERMVIT 1.000000000 -0.13541071 -0.13264056 -0.116092864 0.1172856774  
## PEDS -0.135410713 1.00000000 0.95397364 -0.180823134 0.0442254463  
## PERNOTMVIT -0.132640555 0.95397364 1.00000000 -0.171604015 0.0336270523  
## HARM\_EV -0.116092864 -0.18082313 -0.17160401 1.000000000 -0.1586476154  
## MAN\_COLL 0.117285677 0.04422545 0.03362705 -0.158647615 1.0000000000  
## VE\_FORMS 0.721381119 -0.12679427 -0.12939183 -0.221719048 0.1967947687  
## VE\_TOTAL 0.692968352 -0.08985568 -0.06826025 -0.218652967 0.1840270314  
## RUR\_URB -0.064409726 0.10459510 0.09857490 -0.012285134 -0.0004325961  
## MILEPT 0.002735694 0.02235084 0.01661589 -0.003984619 0.0149894299  
## ROUTE -0.125456871 0.08921637 0.06144892 0.024303621 0.0532494042  
## DRUNK\_DR -0.020336049 -0.12604031 -0.11139136 0.091473741 0.0055046920  
## RD\_OWNER -0.022047797 0.02187235 0.01820972 -0.023768045 -0.0055717880  
## HOSP\_MN -0.088597440 0.04468254 0.04528083 0.017019503 -0.0079167331  
## HOSP\_HR -0.094121769 0.04781526 0.04447671 0.013599218 -0.0078850610  
## VE\_FORMS VE\_TOTAL RUR\_URB MILEPT ROUTE  
## PERMVIT 0.72138112 0.692968352 -0.0644097258 0.002735694 -0.12545687  
## PEDS -0.12679427 -0.089855676 0.1045951032 0.022350842 0.08921637  
## PERNOTMVIT -0.12939183 -0.068260247 0.0985748968 0.016615894 0.06144892  
## HARM\_EV -0.22171905 -0.218652967 -0.0122851338 -0.003984619 0.02430362  
## MAN\_COLL 0.19679477 0.184027031 -0.0004325961 0.014989430 0.05324940  
## VE\_FORMS 1.00000000 0.956178553 -0.0466720412 0.012375034 -0.11030154  
## VE\_TOTAL 0.95617855 1.000000000 -0.0369683049 0.006089381 -0.09654698  
## RUR\_URB -0.04667204 -0.036968305 1.0000000000 -0.051074513 0.25175967  
## MILEPT 0.01237503 0.006089381 -0.0510745134 1.000000000 -0.02496567  
## ROUTE -0.11030154 -0.096546976 0.2517596654 -0.024965671 1.00000000  
## DRUNK\_DR -0.03966249 -0.030516314 0.0241663794 -0.027572221 0.05758370  
## RD\_OWNER -0.01600469 -0.019869570 0.3204079021 -0.103075556 0.30404968  
## HOSP\_MN -0.06914487 -0.060333783 -0.0084340416 0.167369769 0.02403869  
## HOSP\_HR -0.07251903 -0.064889216 -0.0132200116 0.170562482 0.02884273  
## DRUNK\_DR RD\_OWNER HOSP\_MN HOSP\_HR  
## PERMVIT -0.020336049 -0.022047797 -0.088597440 -0.094121769  
## PEDS -0.126040305 0.021872346 0.044682539 0.047815260  
## PERNOTMVIT -0.111391355 0.018209724 0.045280834 0.044476710  
## HARM\_EV 0.091473741 -0.023768045 0.017019503 0.013599218  
## MAN\_COLL 0.005504692 -0.005571788 -0.007916733 -0.007885061  
## VE\_FORMS -0.039662495 -0.016004687 -0.069144865 -0.072519031  
## VE\_TOTAL -0.030516314 -0.019869570 -0.060333783 -0.064889216  
## RUR\_URB 0.024166379 0.320407902 -0.008434042 -0.013220012  
## MILEPT -0.027572221 -0.103075556 0.167369769 0.170562482  
## ROUTE 0.057583704 0.304049676 0.024038688 0.028842731  
## DRUNK\_DR 1.000000000 0.022318072 0.004973338 0.004744726  
## RD\_OWNER 0.022318072 1.000000000 -0.026065494 -0.031999232  
## HOSP\_MN 0.004973338 -0.026065494 1.000000000 0.951792868  
## HOSP\_HR 0.004744726 -0.031999232 0.951792868 1.000000000

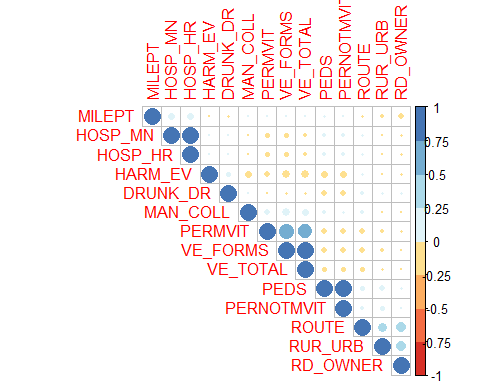
plot(data77\_f)



#corrplot(cor(data77\_f))  
corrplot(cor(data77\_f), method = c("number"))



library(RColorBrewer)  
  
M <-cor(data77\_f)  
corrplot(M, type="upper", order="hclust",  
 col=brewer.pal(n=8, name="RdYlBu"))



mcor<-round(cor(data77\_f),2)  
mcor

## PERMVIT PEDS PERNOTMVIT HARM\_EV MAN\_COLL VE\_FORMS VE\_TOTAL  
## PERMVIT 1.00 -0.14 -0.13 -0.12 0.12 0.72 0.69  
## PEDS -0.14 1.00 0.95 -0.18 0.04 -0.13 -0.09  
## PERNOTMVIT -0.13 0.95 1.00 -0.17 0.03 -0.13 -0.07  
## HARM\_EV -0.12 -0.18 -0.17 1.00 -0.16 -0.22 -0.22  
## MAN\_COLL 0.12 0.04 0.03 -0.16 1.00 0.20 0.18  
## VE\_FORMS 0.72 -0.13 -0.13 -0.22 0.20 1.00 0.96  
## VE\_TOTAL 0.69 -0.09 -0.07 -0.22 0.18 0.96 1.00  
## RUR\_URB -0.06 0.10 0.10 -0.01 0.00 -0.05 -0.04  
## MILEPT 0.00 0.02 0.02 0.00 0.01 0.01 0.01  
## ROUTE -0.13 0.09 0.06 0.02 0.05 -0.11 -0.10  
## DRUNK\_DR -0.02 -0.13 -0.11 0.09 0.01 -0.04 -0.03  
## RD\_OWNER -0.02 0.02 0.02 -0.02 -0.01 -0.02 -0.02  
## HOSP\_MN -0.09 0.04 0.05 0.02 -0.01 -0.07 -0.06  
## HOSP\_HR -0.09 0.05 0.04 0.01 -0.01 -0.07 -0.06  
## RUR\_URB MILEPT ROUTE DRUNK\_DR RD\_OWNER HOSP\_MN HOSP\_HR  
## PERMVIT -0.06 0.00 -0.13 -0.02 -0.02 -0.09 -0.09  
## PEDS 0.10 0.02 0.09 -0.13 0.02 0.04 0.05  
## PERNOTMVIT 0.10 0.02 0.06 -0.11 0.02 0.05 0.04  
## HARM\_EV -0.01 0.00 0.02 0.09 -0.02 0.02 0.01  
## MAN\_COLL 0.00 0.01 0.05 0.01 -0.01 -0.01 -0.01  
## VE\_FORMS -0.05 0.01 -0.11 -0.04 -0.02 -0.07 -0.07  
## VE\_TOTAL -0.04 0.01 -0.10 -0.03 -0.02 -0.06 -0.06  
## RUR\_URB 1.00 -0.05 0.25 0.02 0.32 -0.01 -0.01  
## MILEPT -0.05 1.00 -0.02 -0.03 -0.10 0.17 0.17  
## ROUTE 0.25 -0.02 1.00 0.06 0.30 0.02 0.03  
## DRUNK\_DR 0.02 -0.03 0.06 1.00 0.02 0.00 0.00  
## RD\_OWNER 0.32 -0.10 0.30 0.02 1.00 -0.03 -0.03  
## HOSP\_MN -0.01 0.17 0.02 0.00 -0.03 1.00 0.95  
## HOSP\_HR -0.01 0.17 0.03 0.00 -0.03 0.95 1.00

sapply(data77, class)

## PERMVIT PEDS PERNOTMVIT HARM\_EV MAN\_COLL VE\_FORMS   
## "integer" "integer" "integer" "integer" "integer" "integer"   
## VE\_TOTAL RUR\_URB MILEPT ROUTE DRUNK\_DR RD\_OWNER   
## "integer" "integer" "integer" "integer" "integer" "integer"   
## HOSP\_MN HOSP\_HR FATALS   
## "integer" "integer" "factor"

#data77\_f <- as.numeric(data77\_f)

# Graph the frequency distribution of FATALS variable

#hist(data77\_f$FATALS, freq = T)  
  
#accs\_LM <- lm(formula = FATALS ~ ., data = data77\_f)  
  
#summary(data77\_f)

#library(ElemStatLearn)  
#library(FSelector)  
  
att.scores <- random.forest.importance(FATALS ~ ., datafaux7)  
att.scores

## attr\_importance  
## VE\_TOTAL 34.80644  
## VE\_FORMS 34.10325  
## PVH\_INVL 42.32724  
## PEDS 43.96890  
## PERNOTMVIT 37.39027  
## PERMVIT 81.06192  
## PERSONS 80.45789  
## DAY\_WEEK 100.92342  
## NHS 31.91833  
## RUR\_URB 60.85369  
## FUNC\_SYS 98.10216  
## RD\_OWNER 51.81555  
## ROUTE 58.83038  
## MILEPT 101.06138  
## LATITUDE 190.92752  
## LONGITUD 132.60855  
## SP\_JUR 36.34133  
## HARM\_EV 52.00917  
## MAN\_COLL 57.17715  
## RELJCT1 45.09546  
## RELJCT2 48.68733  
## TYP\_INT 40.77759  
## WRK\_ZONE 64.01197  
## REL\_ROAD 45.36407  
## LGT\_COND 66.16032  
## WEATHER 116.81026  
## SCH\_BUS 23.11150  
## NOT\_HOUR 105.45842  
## NOT\_MIN 129.18312  
## ARR\_HOUR 102.47720  
## ARR\_MIN 112.53709  
## HOSP\_HR 98.94500  
## HOSP\_MN 99.76108  
## CF1 72.50802  
## CF2 32.54906  
## CF3 11.15591  
## DRUNK\_DR 54.84573

# The FSelector package offers several functions to choose the best features using the importance values returned by random.forest.importance.

# The cutoff.biggest.diff function automatically identifies the features which have a significantly higher importance value than other features.

# cutoff.k provides the k features with the highest importance values.

# Similarly, cutoff.k.percent returns k percent of the features with the highest importance values.

f1 <- cutoff.biggest.diff(att.scores)  
print(f1)

## [1] "LATITUDE"

f2 <- cutoff.k(att.scores, k = 15)  
f2

## [1] "LATITUDE" "LONGITUD" "NOT\_MIN" "WEATHER" "ARR\_MIN" "NOT\_HOUR"  
## [7] "ARR\_HOUR" "MILEPT" "DAY\_WEEK" "HOSP\_MN" "HOSP\_HR" "FUNC\_SYS"  
## [13] "PERMVIT" "PERSONS" "CF1"

f3 <- cutoff.k.percent(att.scores, 0.4)  
f3

## [1] "LATITUDE" "LONGITUD" "NOT\_MIN" "WEATHER" "ARR\_MIN" "NOT\_HOUR"  
## [7] "ARR\_HOUR" "MILEPT" "DAY\_WEEK" "HOSP\_MN" "HOSP\_HR" "FUNC\_SYS"  
## [13] "PERMVIT" "PERSONS" "CF1"

# Using all the variables before pre-process using FSelectorRcpp

library(FSelectorRcpp)  
x <- datafaux7  
y <- datafaux7$FATALS  
information\_gain(x=x,y=y)

## attributes importance  
## 1 VE\_TOTAL 3.579099e-02  
## 2 VE\_FORMS 3.580785e-02  
## 3 PVH\_INVL 3.299183e-04  
## 4 PEDS 6.970593e-02  
## 5 PERNOTMVIT 6.490028e-02  
## 6 PERMVIT 1.707645e-01  
## 7 PERSONS 1.717013e-01  
## 8 DAY\_WEEK 2.893260e-03  
## 9 NHS 3.668267e-03  
## 10 RUR\_URB 8.749882e-03  
## 11 FUNC\_SYS 5.051047e-03  
## 12 RD\_OWNER 9.300128e-03  
## 13 ROUTE 8.652305e-03  
## 14 MILEPT 1.182502e-01  
## 15 LATITUDE 4.706732e-03  
## 16 LONGITUD 2.941880e-03  
## 17 SP\_JUR 1.301518e-03  
## 18 HARM\_EV 6.493087e-02  
## 19 MAN\_COLL 4.237324e-02  
## 20 RELJCT1 3.705950e-07  
## 21 RELJCT2 3.337741e-03  
## 22 TYP\_INT 9.089536e-04  
## 23 WRK\_ZONE 2.166899e-04  
## 24 REL\_ROAD 4.399477e-03  
## 25 LGT\_COND 2.255728e-03  
## 26 WEATHER 2.411879e-03  
## 27 SCH\_BUS 1.930379e-06  
## 28 NOT\_HOUR 4.101196e-03  
## 29 NOT\_MIN 7.931440e-03  
## 30 ARR\_HOUR 3.565188e-03  
## 31 ARR\_MIN 7.086643e-03  
## 32 HOSP\_HR 7.893646e-03  
## 33 HOSP\_MN 1.099802e-02  
## 34 CF1 3.416340e-03  
## 35 CF2 1.169767e-03  
## 36 CF3 6.675124e-04  
## 37 FATALS 6.931242e-01  
## 38 DRUNK\_DR 7.065406e-03

FSelectorRcpp

x <- information\_gain(FATALS ~ ., datafaux7)  
cut\_attrs(attrs = x)

## [1] "PERSONS" "PERMVIT" "MILEPT" "PEDS" "HARM\_EV"   
## [6] "PERNOTMVIT" "MAN\_COLL" "VE\_FORMS" "VE\_TOTAL" "HOSP\_MN"   
## [11] "RD\_OWNER" "RUR\_URB" "ROUTE" "NOT\_MIN" "HOSP\_HR"   
## [16] "ARR\_MIN" "DRUNK\_DR" "FUNC\_SYS"

to\_formula(cut\_attrs(attrs = x), "FATALS")

## FATALS ~ PERSONS + PERMVIT + MILEPT + PEDS + HARM\_EV + PERNOTMVIT +   
## MAN\_COLL + VE\_FORMS + VE\_TOTAL + HOSP\_MN + RD\_OWNER + RUR\_URB +   
## ROUTE + NOT\_MIN + HOSP\_HR + ARR\_MIN + DRUNK\_DR + FUNC\_SYS  
## <environment: 0x00000000242e9120>

cut\_attrs(attrs = x, k = 1)

## [1] "PERSONS"

# From FSelectorRcpp, variable selected

# (LATITUDE, NOT\_MIN, LONGITUD, WEATHER, ARR\_MIN, NOT\_HOUR, MILEPT, DAY\_WEEK, ARR\_HOUR, HOSP\_MN, FUNC\_SYS, HOSP\_HR, PERMVIT, PERSONS, LGT\_COND)

# From FSelector, 15 variable selected:

# select(datafaux7, PERMVIT, PEDS, PERNOTMVIT, HARM\_EV, MAN\_COLL, VE\_FORMS, VE\_TOTAL, RUR\_URB, MILEPT, ROUTE,DRUNK\_DR, RD\_OWNER, HOSP\_MN, HOSP\_HR, FATALS)

# Create a Validation Dataset

# By spliting the loaded dataset into two, 80% of which we will used to train our models and 20% that we will hold back as a validation dataset.

# create a list of 80% of the rows in the original dataset we can use for training

# select 20% of the data for validation  
set.seed(7)  
validation\_index <- createDataPartition(data77$FATALS, p=0.80, list=FALSE)

# use the remaining 80% of data to training and testing the models  
set.seed(7)  
validation <- data77[-validation\_index,]  
dataset <- data77[validation\_index,]

table(validation$FATALS)

##   
## No Yes   
## 2330 2298

prop.table(table(validation$FATALS))\*100

##   
## No Yes   
## 50.34572 49.65428

table(data77$FATALS)

##   
## No Yes   
## 11651 11494

prop.table(table(dataset$FATALS))\*100

##   
## No Yes   
## 50.33753 49.66247

# ‘You now have’ The above is the training data in the dataset variable and a validation set that will be used later in the validation variable.

# Note that I’ve replaced the dataset variable with the 80% sample of the dataset. This was an attempt to keep the rest of the code simpler and readable.

# dimensions of dataset  
#dim(dataset)

# 3’ Types of Attributes

# It is a good idea to get an idea of the types of the attributes. They could be doubles, integers, strings, factors and other types.

# Knowing the types is important as it will give you an idea of how to better summarize the data you have and the types of transforms you might need to use to prepare the data before you model it.

# list types for each attribute  
sapply(data77, class)

## PERMVIT PEDS PERNOTMVIT HARM\_EV MAN\_COLL VE\_FORMS   
## "integer" "integer" "integer" "integer" "integer" "integer"   
## VE\_TOTAL RUR\_URB MILEPT ROUTE DRUNK\_DR RD\_OWNER   
## "integer" "integer" "integer" "integer" "integer" "integer"   
## HOSP\_MN HOSP\_HR FATALS   
## "integer" "integer" "factor"

# 3.3 Peek at the Data

It is also always a good idea to actually eyeball your data.

# take a peek at the first 5 rows of the data  
head(data77)

## PERMVIT PEDS PERNOTMVIT HARM\_EV MAN\_COLL VE\_FORMS VE\_TOTAL RUR\_URB  
## 1 4 0 0 38 0 1 1 1  
## 2 2 0 0 33 0 1 1 2  
## 3 1 0 0 42 0 1 1 1  
## 4 3 0 0 12 6 2 2 2  
## 5 2 1 1 8 0 2 2 2  
## 6 2 0 0 24 0 2 2 1  
## MILEPT ROUTE DRUNK\_DR RD\_OWNER HOSP\_MN HOSP\_HR FATALS  
## 1 61 3 1 1 99 99 No  
## 2 153 3 1 1 99 99 No  
## 3 12 3 0 1 99 99 No  
## 4 43 4 0 98 88 88 No  
## 5 0 6 0 4 88 88 No  
## 6 888 1 0 1 99 99 No

3.4 Levels of the Class

# list the levels for the class  
levels(data77$FATALS)

## [1] "No" "Yes"

# This is a binary classification.

# ’ Class Distribution

Let’s now take a look at the number of instances (rows) that belong to each class. We can view this as an absolute count and as a percentage.

# summarize the class distribution  
percentage <- prop.table(table(data77$FATALS)) \* 100  
cbind(freq=table(data77$FATALS), percentage=percentage)

## freq percentage  
## No 11651 50.33917  
## Yes 11494 49.66083

3.6 Statistical Summary

Now finally, we can take a look at a summary of each attribute.

This includes the mean, the min and max values as well as some percentiles (25th, 50th or media and 75th e.g. values at this points if we ordered all the values for an attribute).

# summarize attribute distributions  
summary(data77)

## PERMVIT PEDS PERNOTMVIT HARM\_EV   
## Min. : 0.000 Min. :0.0000 Min. :0.0000 Min. : 1.00   
## 1st Qu.: 2.000 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:12.00   
## Median : 3.000 Median :0.0000 Median :0.0000 Median :12.00   
## Mean : 3.034 Mean :0.1669 Mean :0.1816 Mean :17.58   
## 3rd Qu.: 4.000 3rd Qu.:0.0000 3rd Qu.:0.0000 3rd Qu.:24.00   
## Max. :120.000 Max. :9.0000 Max. :9.0000 Max. :73.00   
## MAN\_COLL VE\_FORMS VE\_TOTAL RUR\_URB   
## Min. : 0.000 Min. : 1.000 Min. : 1.000 Min. :1.000   
## 1st Qu.: 0.000 1st Qu.: 1.000 1st Qu.: 1.000 1st Qu.:1.000   
## Median : 0.000 Median : 2.000 Median : 2.000 Median :1.000   
## Mean : 2.171 Mean : 1.687 Mean : 1.733 Mean :1.609   
## 3rd Qu.: 2.000 3rd Qu.: 2.000 3rd Qu.: 2.000 3rd Qu.:2.000   
## Max. :99.000 Max. :64.000 Max. :64.000 Max. :8.000   
## MILEPT ROUTE DRUNK\_DR RD\_OWNER   
## Min. : 0 Min. :1.000 Min. :0.0000 Min. : 1.0   
## 1st Qu.: 0 1st Qu.:2.000 1st Qu.:0.0000 1st Qu.: 1.0   
## Median : 77 Median :3.000 Median :0.0000 Median : 1.0   
## Mean :14112 Mean :3.407 Mean :0.3072 Mean :16.1   
## 3rd Qu.: 486 3rd Qu.:4.000 3rd Qu.:1.0000 3rd Qu.: 4.0   
## Max. :99998 Max. :9.000 Max. :3.0000 Max. :98.0   
## HOSP\_MN HOSP\_HR FATALS   
## Min. : 0.00 Min. : 0.00 No :11651   
## 1st Qu.:51.00 1st Qu.:21.00 Yes:11494   
## Median :88.00 Median :88.00   
## Mean :75.61 Mean :71.19   
## 3rd Qu.:99.00 3rd Qu.:99.00   
## Max. :99.00 Max. :99.00

# Test Harness

We will 10-fold crossvalidation to estimate accuracy.

This will split the dataset into 10 parts, train in 9 and test on 1 and release for all combinations of train-test splits. This process will be repeated 3 times for each algorithm with different splits of the data into 10 groups, in an effort to get a more accurate estimate.

# Run algorithms using 10-fold cross validation  
control <- trainControl(method="cv", number=10)  
metric <- "Accuracy"

I am using the metric of “Accuracy” to evaluate models. This is a ratio of the number of correctly predicted instances in divided by the total number of instances in the dataset multiplied by 100 to give a percentage (e.g. 95% accurate). We will be using the metric variable when we run build and evaluate each model next.

# ’ Build Models

Evaluating 5 different algorithms:

Linear Discriminant Analysis (LDA)  
Classification and Regression Trees (CART).  
k-Nearest Neighbors (kNN).  
Support Vector Machines (SVM) with a linear kernel.  
Random Forest (RF)  
Generalized Linear Model (glm)  
Gradient Boosting Machine (gbm)  
LogitBoost

This is a good mixture of simple linear (LDA), nonlinear (CART, kNN) and complex nonlinear methods (SVM, RF). We reset the random number seed before reach run to ensure that the evaluation of each algorithm is performed using exactly the same data splits. It ensures the results are directly comparable.

# To build five models:  
# a) linear algorithms - Linear Discriminant Analysis (LDA)  
set.seed(7)  
fit.lda <- train(FATALS~., data=data77, method="lda", metric=metric, trControl=control)  
fit.lda

## Linear Discriminant Analysis   
##   
## 23145 samples  
## 14 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 20829, 20830, 20831, 20831, 20831, 20831, ...   
## Resampling results:  
##   
## Accuracy Kappa   
## 0.7048601 0.4091952

# b) nonlinear algorithms  
# Classification and Regression Trees (CART)  
set.seed(7)  
fit.cart <- train(FATALS~., data=data77, method="rpart", metric=metric, trControl=control)  
fit.cart

## CART   
##   
## 23145 samples  
## 14 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 20829, 20830, 20831, 20831, 20831, 20831, ...   
## Resampling results across tuning parameters:  
##   
## cp Accuracy Kappa   
## 0.02694159 0.7592132 0.5191390  
## 0.05176614 0.7333284 0.4678173  
## 0.40864799 0.6010148 0.1998124  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was cp = 0.02694159.

# k-Nearest Neighbors (kNN)  
set.seed(7)  
fit.knn <- train(FATALS~., data=data77, method="knn", metric=metric, trControl=control)  
fit.knn

## k-Nearest Neighbors   
##   
## 23145 samples  
## 14 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 20829, 20830, 20831, 20831, 20831, 20831, ...   
## Resampling results across tuning parameters:  
##   
## k Accuracy Kappa   
## 5 0.7888950 0.5787272  
## 7 0.7527757 0.5066952  
## 9 0.7301786 0.4615132  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was k = 5.

# c) advanced algorithms  
# Support Vector Machines (SVM) with a linear kernel  
set.seed(7)  
fit.svm <- train(FATALS~., data=data77, method="svmRadial", metric=metric, trControl=control)  
fit.svm

## Support Vector Machines with Radial Basis Function Kernel   
##   
## 23145 samples  
## 14 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 20829, 20830, 20831, 20831, 20831, 20831, ...   
## Resampling results across tuning parameters:  
##   
## C Accuracy Kappa   
## 0.25 0.7994802 0.5993070  
## 0.50 0.8060038 0.6123672  
## 1.00 0.8146451 0.6296758  
##   
## Tuning parameter 'sigma' was held constant at a value of 0.08257444  
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were sigma = 0.08257444 and C = 1.

# Random Forest (rf)  
set.seed(7)  
fit.rf <- train(FATALS~., data=data77, method="rf", metric=metric, trControl=control)  
fit.rf

## Random Forest   
##   
## 23145 samples  
## 14 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 20829, 20830, 20831, 20831, 20831, 20831, ...   
## Resampling results across tuning parameters:  
##   
## mtry Accuracy Kappa   
## 2 0.8682205 0.7368277  
## 8 0.9503988 0.9008514  
## 14 0.9509173 0.9018874  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was mtry = 14.

# Generalized Linear Model (glm)  
set.seed(7)  
fit.glm <- train(FATALS~., data=data77, method="glm", metric=metric, trControl=control)

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred  
  
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred  
  
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred  
  
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred  
  
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred  
  
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred  
  
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred  
  
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred  
  
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

fit.glm

## Generalized Linear Model   
##   
## 23145 samples  
## 14 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 20829, 20830, 20831, 20831, 20831, 20831, ...   
## Resampling results:  
##   
## Accuracy Kappa   
## 0.722013 0.4435306

# Gradient Boosting Machine (gbm)  
#set.seed(7)  
#fit.gbm <- train(FATALS~., data=data77, method="gbm", metric=metric, trControl=control)  
#fit.gbm

# mda  
#set.seed(7)  
#fit.mda <- train(FATALS~., data=dataset, method="mda", metric=metric, trControl=control)  
#fit.mda

# LogitBoost  
set.seed(7)  
fit.LogitBoost <- train(FATALS~., data=data77, method="LogitBoost", metric=metric, trControl=control)  
fit.LogitBoost

## Boosted Logistic Regression   
##   
## 23145 samples  
## 14 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 20829, 20830, 20831, 20831, 20831, 20831, ...   
## Resampling results across tuning parameters:  
##   
## nIter Accuracy Kappa   
## 11 0.7331596 0.4669077  
## 21 0.7646577 0.5298133  
## 31 0.7574866 0.5155549  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was nIter = 21.

# Select Best Model

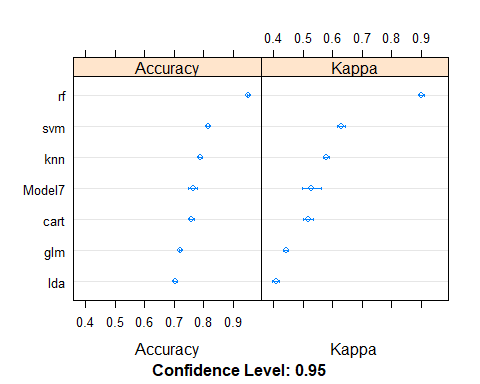
There are 5 models and accuracy estimations for each. I’m going to compare the models to each other and select the most accurate.

# summarize accuracy of models  
results <- resamples(list(lda=fit.lda, cart=fit.cart, knn=fit.knn, svm=fit.svm, rf=fit.rf, glm=fit.glm, fit.LogitBoost))  
summary(results)

##   
## Call:  
## summary.resamples(object = results)  
##   
## Models: lda, cart, knn, svm, rf, glm, Model7   
## Number of resamples: 10   
##   
## Accuracy   
## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  
## lda 0.6949006 0.6967698 0.7046889 0.7048601 0.7125422 0.7165082 0  
## cart 0.7352052 0.7511884 0.7617199 0.7592132 0.7700615 0.7713915 0  
## knn 0.7770095 0.7834918 0.7891548 0.7888950 0.7947084 0.8016422 0  
## svm 0.7994814 0.8078003 0.8169836 0.8146451 0.8206841 0.8259179 0  
## rf 0.9412273 0.9466467 0.9518150 0.9509173 0.9551836 0.9585492 0  
## glm 0.7136069 0.7165385 0.7212619 0.7220130 0.7280283 0.7299049 0  
## Model7 0.7299049 0.7449765 0.7698790 0.7646577 0.7818712 0.7961123 0  
##   
## Kappa   
## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  
## lda 0.3893195 0.3929951 0.4089008 0.4091952 0.4245819 0.4324738 0  
## cart 0.4704385 0.5031623 0.5242367 0.5191390 0.5407461 0.5437945 0  
## knn 0.5550348 0.5680243 0.5792677 0.5787272 0.5902238 0.6041526 0  
## svm 0.5993699 0.6160485 0.6343812 0.6296758 0.6416662 0.6521270 0  
## rf 0.8825332 0.8933558 0.9036803 0.9018874 0.9104071 0.9171350 0  
## glm 0.4265381 0.4326577 0.4420070 0.4435306 0.4554677 0.4593413 0  
## Model7 0.4597550 0.4899433 0.5406953 0.5298133 0.5646525 0.5929685 0

Plotting of the model evaluation results and compare the spread and the mean accuracy of each model. There is a population of accuracy measures for each algorithm because each algorithm was evaluated 10 times (10 fold cross validation).

# compare accuracy of models  
dotplot(results)



We can see that the most accurate model in this case was LDA: Comparison of Machine Learning Algorithms on Iris Dataset in R

Comparison of Machine Learning Algorithms on Iris Dataset in R

#The results for just the rf model can be summarized.  
# summarize Best Model  
print(fit.rf)

## Random Forest   
##   
## 23145 samples  
## 14 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 20829, 20830, 20831, 20831, 20831, 20831, ...   
## Resampling results across tuning parameters:  
##   
## mtry Accuracy Kappa   
## 2 0.8682205 0.7368277  
## 8 0.9503988 0.9008514  
## 14 0.9509173 0.9018874  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was mtry = 14.

#The results for just the svm model can be summarized.  
# summarize Best Model  
print(fit.svm)

## Support Vector Machines with Radial Basis Function Kernel   
##   
## 23145 samples  
## 14 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 20829, 20830, 20831, 20831, 20831, 20831, ...   
## Resampling results across tuning parameters:  
##   
## C Accuracy Kappa   
## 0.25 0.7994802 0.5993070  
## 0.50 0.8060038 0.6123672  
## 1.00 0.8146451 0.6296758  
##   
## Tuning parameter 'sigma' was held constant at a value of 0.08257444  
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were sigma = 0.08257444 and C = 1.

#The results for just the cart model can be summarized.  
# summarize Best Model  
print(fit.cart)

## CART   
##   
## 23145 samples  
## 14 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 20829, 20830, 20831, 20831, 20831, 20831, ...   
## Resampling results across tuning parameters:  
##   
## cp Accuracy Kappa   
## 0.02694159 0.7592132 0.5191390  
## 0.05176614 0.7333284 0.4678173  
## 0.40864799 0.6010148 0.1998124  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was cp = 0.02694159.

# 6. Make Predictions

The LDA was the most accurate model. Now we want to get an idea of the accuracy of the model on our validation set.

This will give us an independent final check on the accuracy of the best model. It is valuable to keep a validation set just in case you made a slip during such as overfitting to the training set or a data leak. Both will result in an overly optimistic result.

We can run the rf model directly on the validation set and summarize the results in a confusion matrix.

# estimate skill of rf on the validation dataset  
predictions <- predict(fit.rf, validation)  
confusionMatrix(predictions, validation$FATALS)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 2276 10  
## Yes 54 2288  
##   
## Accuracy : 0.9862   
## 95% CI : (0.9824, 0.9893)  
## No Information Rate : 0.5035   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.9723   
## Mcnemar's Test P-Value : 7.658e-08   
##   
## Sensitivity : 0.9768   
## Specificity : 0.9956   
## Pos Pred Value : 0.9956   
## Neg Pred Value : 0.9769   
## Prevalence : 0.5035   
## Detection Rate : 0.4918   
## Detection Prevalence : 0.4939   
## Balanced Accuracy : 0.9862   
##   
## 'Positive' Class : No   
##

`

# estimate skill of SVM on the validation dataset  
predictions <- predict(fit.svm, validation)  
confusionMatrix(predictions, validation$FATALS)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 1735 221  
## Yes 595 2077  
##   
## Accuracy : 0.8237   
## 95% CI : (0.8124, 0.8346)  
## No Information Rate : 0.5035   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.6477   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.7446   
## Specificity : 0.9038   
## Pos Pred Value : 0.8870   
## Neg Pred Value : 0.7773   
## Prevalence : 0.5035   
## Detection Rate : 0.3749   
## Detection Prevalence : 0.4226   
## Balanced Accuracy : 0.8242   
##   
## 'Positive' Class : No   
##

# estimate skill of LogitBoost on the validation dataset  
predictions <- predict(fit.LogitBoost, validation)  
confusionMatrix(predictions, validation$FATALS)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 1727 608  
## Yes 603 1690  
##   
## Accuracy : 0.7383   
## 95% CI : (0.7254, 0.7509)  
## No Information Rate : 0.5035   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.4766   
## Mcnemar's Test P-Value : 0.9085   
##   
## Sensitivity : 0.7412   
## Specificity : 0.7354   
## Pos Pred Value : 0.7396   
## Neg Pred Value : 0.7370   
## Prevalence : 0.5035   
## Detection Rate : 0.3732   
## Detection Prevalence : 0.5045   
## Balanced Accuracy : 0.7383   
##   
## 'Positive' Class : No   
##

# estimate skill of lda on the validation dataset  
predictions <- predict(fit.lda, validation)  
confusionMatrix(predictions, validation$FATALS)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 1792 838  
## Yes 538 1460  
##   
## Accuracy : 0.7027   
## 95% CI : (0.6893, 0.7158)  
## No Information Rate : 0.5035   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.4048   
## Mcnemar's Test P-Value : 7.598e-16   
##   
## Sensitivity : 0.7691   
## Specificity : 0.6353   
## Pos Pred Value : 0.6814   
## Neg Pred Value : 0.7307   
## Prevalence : 0.5035   
## Detection Rate : 0.3872   
## Detection Prevalence : 0.5683   
## Balanced Accuracy : 0.7022   
##   
## 'Positive' Class : No   
##

# estimate skill of rf on the validation dataset  
predictions <- predict(fit.rf, validation)  
confusionMatrix(predictions, validation$FATALS)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 2276 10  
## Yes 54 2288  
##   
## Accuracy : 0.9862   
## 95% CI : (0.9824, 0.9893)  
## No Information Rate : 0.5035   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.9723   
## Mcnemar's Test P-Value : 7.658e-08   
##   
## Sensitivity : 0.9768   
## Specificity : 0.9956   
## Pos Pred Value : 0.9956   
## Neg Pred Value : 0.9769   
## Prevalence : 0.5035   
## Detection Rate : 0.4918   
## Detection Prevalence : 0.4939   
## Balanced Accuracy : 0.9862   
##   
## 'Positive' Class : No   
##

# estimate skill of knn on the validation dataset  
predictions <- predict(fit.knn, validation)  
confusionMatrix(predictions, validation$FATALS)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 1635 43  
## Yes 695 2255  
##   
## Accuracy : 0.8405   
## 95% CI : (0.8297, 0.851)  
## No Information Rate : 0.5035   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.6817   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.7017   
## Specificity : 0.9813   
## Pos Pred Value : 0.9744   
## Neg Pred Value : 0.7644   
## Prevalence : 0.5035   
## Detection Rate : 0.3533   
## Detection Prevalence : 0.3626   
## Balanced Accuracy : 0.8415   
##   
## 'Positive' Class : No   
##

#sessionInfo()