Analysis of Collision Type in Seattle with Tree-based Methods and Multinomial Regression - R Notebook

Aishwarya Saibewar, Karthika Selvaraj, Xuanzhi Li

# Importing the libraries required for the computation  
library(tidyverse)

## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.1 ──

## ✔ ggplot2 3.3.6 ✔ purrr 0.3.4  
## ✔ tibble 3.1.7 ✔ dplyr 1.0.9  
## ✔ tidyr 1.2.0 ✔ stringr 1.4.0  
## ✔ readr 2.1.2 ✔ forcats 0.5.1

## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()

library(ggplot2)  
library(readr)  
library(tidyr)  
library(dplyr)  
library(boot)  
library(randomForest)

## randomForest 4.7-1

## 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(gbm)

## Loaded gbm 2.1.8

library(tree)  
library(rpart)  
library(rpart.plot)  
library(caret)

## Loading required package: lattice

##   
## Attaching package: 'lattice'

## The following object is masked from 'package:boot':  
##   
## melanoma

##   
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':  
##   
## lift

library(nnet)

# Importing the Collision data provided by Seattle Police department.Pass "stringsAsFactors" since we have factor variables  
  
df <- read.csv("/Users/kat/Documents/DATA 5322 - ML 2/Project/Collisions.csv",stringsAsFactors = TRUE)

#Remove document key variables which don't have much significance  
df1 = subset (df, select = -c(OBJECTID,COLDETKEY,INTKEY,REPORTNO,STATUS,LOCATION,EXCEPTRSNCODE,EXCEPTRSNDESC,SEVERITYDESC,SERIOUSINJURIES,JUNCTIONTYPE,SDOT\_COLCODE,SDOT\_COLDESC,ST\_COLCODE,ST\_COLDESC,PEDROWNOTGRNT,SDOTCOLNUM,SEGLANEKEY,CROSSWALKKEY,INCDTTM))

#Converted INCDATE to date  
df1$INCDATE = as.Date(df1$INCDATE)

#Convert values in SPEEDING, Y as 1 and N as 0  
df1$SPEEDING <- ifelse(df1$SPEEDING == 'Y', 1 , 0)

#Convert values in INATTENTIONIND, Y as 1 and N as 0  
df1$INATTENTIONIND <- ifelse(df1$INATTENTIONIND == 'Y', 1 , 0)

#Convert values in UNDERINFL, "Y and 1 as 1" and "N and 0 as 0"  
df1$UNDERINFL <- ifelse(df1$UNDERINFL == 'Y'|| df1$UNDERINFL == '1', 1 , 0)

## Warning in df1$UNDERINFL == "Y" || df1$UNDERINFL == "1": 'length(x) = 234522 >  
## 1' in coercion to 'logical(1)'  
  
## Warning in df1$UNDERINFL == "Y" || df1$UNDERINFL == "1": 'length(x) = 234522 >  
## 1' in coercion to 'logical(1)'

#Convert values in SEVERITYCODE "2b as 4" and "N and 0 as 0"  
df1$SEVERITYCODE = as.integer(df1$SEVERITYCODE)  
replace(df1$SEVERITYCODE, '2b','4" "2" )

#Convert the SEVERITYCODE to factor variable  
df1$SEVERITYCODE = as.factor(df1$SEVERITYCODE)

# Removing the NA if any  
df1 = na.omit(df1)

#Altering the values for COLLISIONTYPE to view clearly on decision tree with out overlapping--- Collision of vehicles at angles=1 ,Collision with Cycles=2,Head On Collision =3,Collision while turning left or right =4, Collision with Parked Car =5,Collision with Pedestrian =6,Rear end collision = 7,Sideswipe collision = 8, Unknown  
  
  
df1$COLLISIONTYPE <- ifelse(df1$COLLISIONTYPE == 'Angles', '1',  
 ifelse(df1$COLLISIONTYPE == 'Cycles', '2',  
 ifelse(df1$COLLISIONTYPE == 'Head On', '3',  
 ifelse(df1$COLLISIONTYPE == 'Left Turn', '4',  
 ifelse(df1$COLLISIONTYPE == 'Parked Car', '5',  
 ifelse(df1$COLLISIONTYPE == 'Pedestrian', '6',  
 ifelse(df1$COLLISIONTYPE == 'Rear Ended', '7',  
 ifelse(df1$COLLISIONTYPE == 'Right Turn', '4',  
 ifelse(df1$COLLISIONTYPE == 'Sideswipe', '8','Unknown')))))))))

#Altering values for LIGHTCOND-- 0 = Light, 1 = Medium, 2 = Dark  
  
df1$LIGHTCOND <- ifelse(df1$LIGHTCOND == 'Dark - No Street Lights', '2',  
 ifelse(df1$LIGHTCOND == 'Dark - Street Lights Off', '2',  
 ifelse(df1$LIGHTCOND == 'Dark - Street Lights On', '1',  
 ifelse(df1$LIGHTCOND == 'Dark - Unknown Lighting', '2',  
 ifelse(df1$LIGHTCOND == 'Dawn', '1',  
 ifelse(df1$LIGHTCOND == 'Daylight', '0',  
 ifelse(df1$LIGHTCOND == 'Dusk', '1',  
 ifelse(df1$LIGHTCOND == 'Others', 'Unknown','Unknown'))))))))

#Altering values for WEATHER-- 0 = Clear, 1 = Overcast and Cloudy, 2 = Windy, 3 = Rain and Snow  
  
df1$WEATHER <- ifelse(df1$WEATHER == 'Overcast', '1',  
 ifelse(df1$WEATHER == 'Clear', '0',  
 ifelse(df1$WEATHER == 'Other', '3',  
 ifelse(df1$WEATHER == 'Raining', '3',  
 ifelse(df1$WEATHER == 'Snowing', '3',  
 ifelse(df1$WEATHER == 'Fog/Smog/Smoke', '2',  
 ifelse(df1$WEATHER == 'Sleet/Hail/Freezing Rain Blowing Sand/Dirt', '3','Unknown')))))))

#Altering values for ROADCOND-- 0 = Dry, 1 = Mushy, 2 = Wet  
  
df1$ROADCOND <- ifelse(df1$ROADCOND == 'Wet', 1,  
 ifelse(df1$ROADCOND == 'Standing Water', 1,  
 ifelse(df1$ROADCOND == 'Snow/Slush', 2,  
 ifelse(df1$ROADCOND == 'Ice', 2,  
 ifelse(df1$ROADCOND == 'Dry', 3,  
 ifelse(df1$ROADCOND == 'Sand/Mud/Dirt', 3,  
 ifelse(df1$ROADCOND == 'Oil', 1,  
 ifelse(df1$ROADCOND == 'Other', 'Unknown','Unknown'))))))))

#Converting the datatype of below variables to factors  
df1$LIGHTCOND = as.factor(df1$LIGHTCOND)  
df1$UNDERINFL = as.factor(df1$UNDERINFL)  
df1$SPEEDING = as.factor(df1$SPEEDING)  
df1$COLLISIONTYPE = as.factor(df1$COLLISIONTYPE)  
df1$WEATHER = as.factor(df1$WEATHER)  
df1$ROADCOND = as.factor(df1$ROADCOND)  
df1$LIGHTCOND = as.factor(df1$LIGHTCOND)  
df1$INCDATE = as.character(df1$INCDATE)  
df1$SEVERITYCODE = as.factor(df1$SEVERITYCODE)  
df1$PEDCOUNT = as.factor(df1$PEDCOUNT)  
df1$VEHCOUNT = as.factor(df1$VEHCOUNT)  
df1$PEDCYLCOUNT = as.factor(df1$PEDCYLCOUNT)

#Renaming variables to meaningful names  
df1 = rename(df1, VEHICLE\_COUNT = VEHCOUNT, PEDESTRIANS\_COUNT = PEDCOUNT, BICYCLE\_COUNT= PEDCYLCOUNT,ADDRESS\_TYPE=ADDRTYPE,SEVERITY=SEVERITYCODE,LIGHT\_CONDITIONS=LIGHTCOND,COLLISION\_TYPE=COLLISIONTYPE)

#Split the data into training and test in the ratio of 60:20 respectively  
train <- sample(nrow(df1) \* 0.6)  
df\_train <- df1[train, ]  
df\_test <- df1[-train, ]

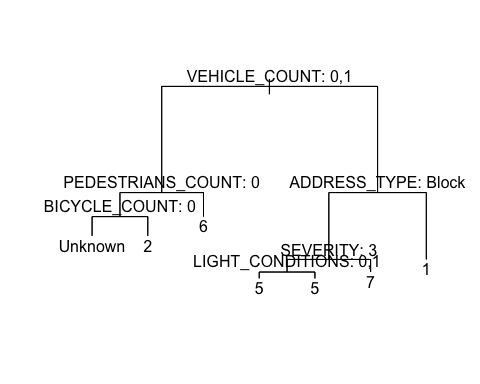
#Fit the tree model  
df\_tree<- tree(COLLISION\_TYPE~., data = df\_train)

## Warning in tree(COLLISION\_TYPE ~ ., data = df\_train): NAs introduced by coercion

#Summary of the tree model  
summary(df\_tree)

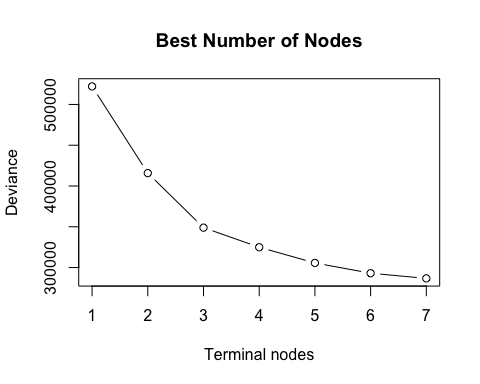
##   
## Classification tree:  
## tree(formula = COLLISION\_TYPE ~ ., data = df\_train)  
## Variables actually used in tree construction:  
## [1] "VEHICLE\_COUNT" "PEDESTRIANS\_COUNT" "BICYCLE\_COUNT"   
## [4] "ADDRESS\_TYPE" "SEVERITY" "LIGHT\_CONDITIONS"   
## Number of terminal nodes: 7   
## Residual mean deviance: 2.106 = 286600 / 136100   
## Misclassification error rate: 0.3674 = 50007 / 136099

#Plot the decision tree  
plot(df\_tree)  
text(df\_tree, pretty = 0)



# Performing the cross validation to find the best node with the least deviance.It reduces the decision tree's size and creates a subtree to balance variation and bias  
cv.df <- cv.tree(df\_tree)

plot(cv.df$size, cv.df$dev, type = "b", xlab = "Terminal nodes", ylab="Deviance")+  
 title(main="Best Number of Nodes")



## integer(0)

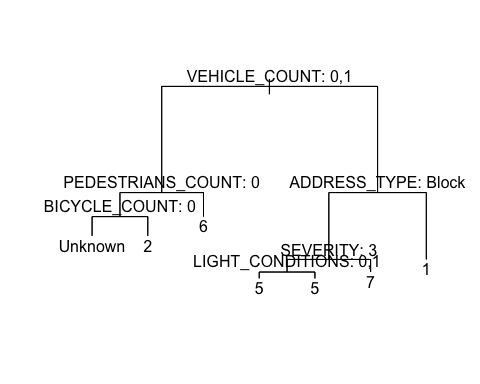
# Pruning the tree model and display the pruned tree  
prune.df <- prune.tree(df\_tree,best=7, method = c("misclass"))   
prune.df

## node), split, n, deviance, yval, (yprob)  
## \* denotes terminal node  
##   
## 1) root 136099 522100.00 5 ( 1.629e-01 2.525e-02 9.780e-03 8.024e-02 2.362e-01 3.316e-02 1.605e-01 8.389e-02 2.081e-01 )   
## 2) VEHICLE\_COUNT: 0,1 28729 44470.00 Unknown ( 6.962e-05 1.176e-01 3.481e-05 0.000e+00 1.044e-04 1.548e-01 0.000e+00 0.000e+00 7.274e-01 )   
## 4) PEDESTRIANS\_COUNT: 0 24195 19440.00 Unknown ( 8.266e-05 1.373e-01 4.133e-05 0.000e+00 4.133e-05 0.000e+00 0.000e+00 0.000e+00 8.626e-01 )   
## 8) BICYCLE\_COUNT: 0 20868 65.68 Unknown ( 4.792e-05 4.792e-05 0.000e+00 0.000e+00 4.792e-05 0.000e+00 0.000e+00 0.000e+00 9.999e-01 ) \*  
## 9) BICYCLE\_COUNT: 1,2 3327 111.40 2 ( 3.006e-04 9.979e-01 3.006e-04 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 1.503e-03 ) \*  
## 5) PEDESTRIANS\_COUNT: 1,2,3,4,5,6 4534 985.50 6 ( 0.000e+00 1.279e-02 0.000e+00 0.000e+00 4.411e-04 9.808e-01 0.000e+00 0.000e+00 5.955e-03 ) \*  
## 3) VEHICLE\_COUNT: 2,3,4,5,6,7,8,9,10,11,12,14 107370 371400.00 5 ( 2.064e-01 5.402e-04 1.239e-02 1.017e-01 2.993e-01 6.147e-04 2.035e-01 1.063e-01 6.919e-02 )   
## 6) ADDRESS\_TYPE: Block 72075 214200.00 5 ( 4.078e-02 4.717e-04 1.430e-02 2.520e-02 4.347e-01 7.492e-04 2.682e-01 1.243e-01 9.132e-02 )   
## 12) SEVERITY: 3 56839 157800.00 5 ( 3.666e-02 3.519e-05 9.272e-03 2.291e-02 5.210e-01 5.278e-05 1.884e-01 1.362e-01 8.554e-02 )   
## 24) LIGHT\_CONDITIONS: 0,1 48216 142900.00 5 ( 4.185e-02 4.148e-05 1.056e-02 2.638e-02 4.545e-01 6.222e-05 2.148e-01 1.544e-01 9.740e-02 ) \*  
## 25) LIGHT\_CONDITIONS: 2,Unknown 8623 8505.00 5 ( 7.654e-03 0.000e+00 2.087e-03 3.479e-03 8.926e-01 0.000e+00 4.047e-02 3.444e-02 1.925e-02 ) \*  
## 13) SEVERITY: 4,5,6 15236 43800.00 7 ( 5.612e-02 2.100e-03 3.308e-02 3.374e-02 1.127e-01 3.347e-03 5.659e-01 8.014e-02 1.129e-01 ) \*  
## 7) ADDRESS\_TYPE: Intersection 35295 90240.00 1 ( 5.447e-01 6.800e-04 8.471e-03 2.579e-01 2.295e-02 3.400e-04 7.134e-02 6.956e-02 2.400e-02 ) \*

names(prune.df)

## [1] "frame" "where" "terms" "call" "y" "weights"

plot(prune.df)  
text(prune.df, pretty = 0)



#Examine the model's performance on the test data.  
yhat <- predict(prune.df, newdata = df\_test, type="class")

## Warning in pred1.tree(object, tree.matrix(newdata)): NAs introduced by coercion

treemodel<- mean(yhat == df\_test$COLLISION\_TYPE)  
treemodel

## [1] 0.6516631

#BAGGING Method (m=p)

# mtry= m= number of predictors considered at each split  
#Fit the bagging model with m=p, i.e., with the full set of predictors  
npredictors = length(df\_train)  
bagging.collision <- randomForest(COLLISION\_TYPE~VEHICLE\_COUNT+PEDESTRIANS\_COUNT+BICYCLE\_COUNT+ADDRESS\_TYPE+SEVERITY+LIGHT\_CONDITIONS,   
 data=df\_train,  
 mtry=npredictors-14,  
 importance=TRUE,ntree=500)

#Examine the model's performance on the test data.  
predicted\_collision <-predict(bagging.collision, newdata = df\_test)  
bagging<-mean(predicted\_collision == df\_test$COLLISION\_TYPE)  
bagging

## [1] 0.6542972

The model predicted correctly 65.42% of the time

#Examine the error rate on the test data.  
mean(predicted\_collision != df\_test$COLLISION\_TYPE)

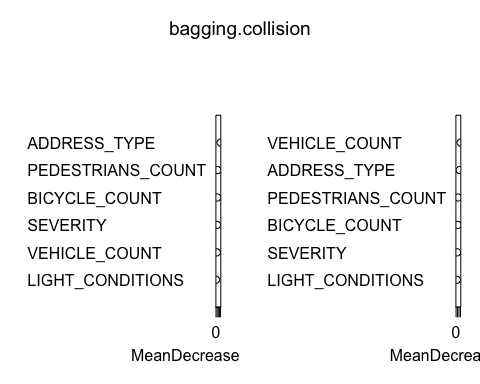
## [1] 0.3457028

The model has an Error rate of 34.5%

#Plot the importance of each predictors  
importance(bagging.collision)

## 1 2 3 4 5  
## VEHICLE\_COUNT 106.61834 53.997521 1.9939090 1.877134 1148.47846  
## PEDESTRIANS\_COUNT 30.60635 12.728272 4.8272171 -16.459928 21.02954  
## BICYCLE\_COUNT 76.14788 4476.028952 2.4631925 5.250089 61.71163  
## ADDRESS\_TYPE 2093.01718 7.220817 13.1653019 13.684305 1226.26156  
## SEVERITY 23.05846 30.331319 -48.0711941 -22.393710 508.19951  
## LIGHT\_CONDITIONS 456.58556 14.834984 0.7459782 -1.786583 184.72105  
## 6 7 8 Unknown  
## VEHICLE\_COUNT 138.057389 657.7047 8.086356 182.134103  
## PEDESTRIANS\_COUNT 3633.857188 172.7616 -8.457122 369.444775  
## BICYCLE\_COUNT 81.046744 133.8585 1.417022 305.555067  
## ADDRESS\_TYPE 7.321964 560.6540 3.507063 6.112638  
## SEVERITY 23.281820 340.6714 -22.565370 1.962019  
## LIGHT\_CONDITIONS 7.397364 142.7200 -8.796107 26.627339  
## MeanDecreaseAccuracy MeanDecreaseGini  
## VEHICLE\_COUNT 356.00837 15544.560  
## PEDESTRIANS\_COUNT 649.66220 6618.014  
## BICYCLE\_COUNT 595.92261 5889.165  
## ADDRESS\_TYPE 2025.62936 12411.156  
## SEVERITY 477.98248 4010.185  
## LIGHT\_CONDITIONS 40.20967 2482.976

varImpPlot(bagging.collision)



#Random Forest model(m= p/2)

#Fit the Random forest model with m=p/2, i.e., with half of the predictors  
npredictors = length(df\_train)  
rf.collision1 <- randomForest(COLLISION\_TYPE~VEHICLE\_COUNT+PEDESTRIANS\_COUNT+BICYCLE\_COUNT+ADDRESS\_TYPE+SEVERITY+LIGHT\_CONDITIONS,   
 data=df\_train,  
 mtry=(npredictors-14)/2,  
 importance=TRUE,ntree=500)

#Examine the model's performance on the test data.  
predicted\_collision <-predict(rf.collision1, newdata = df\_test)  
rfhalf<-mean(predicted\_collision == df\_test$COLLISION\_TYPE)  
rfhalf

## [1] 0.6542531

The model predicted correctly 65.43% of the time

#Examine the error rate on the test data.  
mean(predicted\_collision != df\_test$COLLISION\_TYPE)

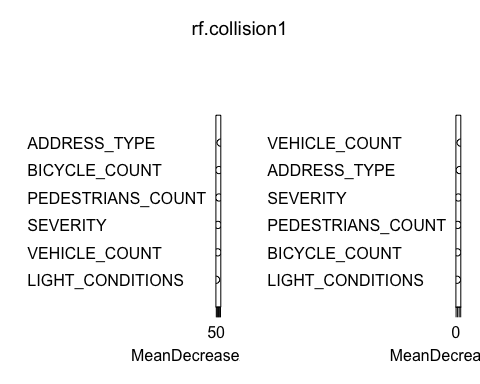
## [1] 0.3457469

The model has an Error rate of 34.56%

#Plot the importance of each predictors  
importance(rf.collision1)

## 1 2 3 4 5  
## VEHICLE\_COUNT 128.14342 27.487755 3.583232 0.2081204 183.41677  
## PEDESTRIANS\_COUNT 20.11285 10.844505 2.777621 -11.1099792 16.36994  
## BICYCLE\_COUNT 64.09046 319.773528 3.941781 3.0272308 26.95727  
## ADDRESS\_TYPE 505.69629 3.075265 11.330858 11.9505863 179.41083  
## SEVERITY 20.58951 13.174019 -19.362483 -13.5798594 210.89656  
## LIGHT\_CONDITIONS 47.36663 3.644856 -0.512663 -0.4000770 42.67314  
## 6 7 8 Unknown  
## VEHICLE\_COUNT 115.19402113 132.07792 -2.013353 69.448686  
## PEDESTRIANS\_COUNT 312.58631108 78.04451 -6.860783 59.807943  
## BICYCLE\_COUNT 84.25292950 76.12144 0.000000 63.681158  
## ADDRESS\_TYPE 5.03312511 235.96048 2.558568 2.119653  
## SEVERITY 9.66938283 111.47237 -5.757831 -1.834624  
## LIGHT\_CONDITIONS 0.01007431 22.08645 -6.042651 19.842414  
## MeanDecreaseAccuracy MeanDecreaseGini  
## VEHICLE\_COUNT 147.69223 14706.819  
## PEDESTRIANS\_COUNT 240.67524 5261.029  
## BICYCLE\_COUNT 241.08760 4268.717  
## ADDRESS\_TYPE 357.87561 11492.639  
## SEVERITY 166.11777 7343.166  
## LIGHT\_CONDITIONS 33.83426 3178.438

varImpPlot(rf.collision1)



#Random Forest model(m= sqrt(p))

#Fit the Random forest model with m=sqrt(p), i.e., with square root of predictors  
npredictors = length(df\_train)  
rf.collision2 <- randomForest(COLLISION\_TYPE~VEHICLE\_COUNT+PEDESTRIANS\_COUNT+BICYCLE\_COUNT+ADDRESS\_TYPE+SEVERITY+LIGHT\_CONDITIONS,   
 data=df\_train,  
 mtry=sqrt(npredictors-14),  
 importance=TRUE,ntree=500)

#Examine the model's performance on the test data.  
predicted\_collision <-predict(rf.collision2, newdata = df\_test)  
rfsqrt<-mean(predicted\_collision == df\_test$COLLISION\_TYPE)  
rfsqrt

## [1] 0.6535808

The model predicted correctly 65.36% of the time

#Examine the error rate on the test data.  
mean(predicted\_collision != df\_test$COLLISION\_TYPE)

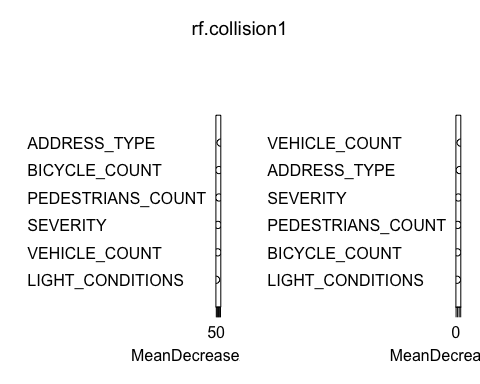
## [1] 0.3464192

The model has an Error rate of 34.63%

#Plot the importance of each predictors  
importance(rf.collision2)

## 1 2 3 4 5 6  
## VEHICLE\_COUNT 55.87557 13.057919 4.471004 1.921332 64.59262 69.197064  
## PEDESTRIANS\_COUNT 15.77901 7.382353 1.897393 -9.075702 20.69713 105.284159  
## BICYCLE\_COUNT 44.59010 80.546615 2.324701 3.074993 29.18157 60.432894  
## ADDRESS\_TYPE 131.13821 4.161340 8.627864 9.523937 85.96264 8.180127  
## SEVERITY 17.30953 12.493220 -11.072160 -7.854974 70.07429 11.688826  
## LIGHT\_CONDITIONS 28.93332 4.548970 1.126242 0.855323 10.09360 3.599495  
## 7 8 Unknown MeanDecreaseAccuracy  
## VEHICLE\_COUNT 52.62083 -1.403581 51.741952 79.06057  
## PEDESTRIANS\_COUNT 35.71605 -6.425537 40.461334 105.13001  
## BICYCLE\_COUNT 36.00242 0.000000 42.282660 95.97802  
## ADDRESS\_TYPE 58.29844 2.190480 4.833810 123.37900  
## SEVERITY 46.50244 -5.739345 6.185213 64.68045  
## LIGHT\_CONDITIONS 12.66071 -6.082562 19.378357 27.54376  
## MeanDecreaseGini  
## VEHICLE\_COUNT 13065.273  
## PEDESTRIANS\_COUNT 4506.968  
## BICYCLE\_COUNT 3632.098  
## ADDRESS\_TYPE 10768.749  
## SEVERITY 7304.188  
## LIGHT\_CONDITIONS 3355.665

varImpPlot(rf.collision1)



#Random Forest model(m= p/3)

#Fit the Random forest model with m=sqrt(p), i.e., with one third of predictors  
npredictors = length(df\_train)  
rf.collision3 <- randomForest(COLLISION\_TYPE~VEHICLE\_COUNT+PEDESTRIANS\_COUNT+BICYCLE\_COUNT+ADDRESS\_TYPE+SEVERITY+LIGHT\_CONDITIONS,   
 data=df\_train,  
 mtry=(npredictors-14)/3,  
 importance=TRUE,ntree=500)

#Examine the model's performance on the test data.  
predicted\_collision <-predict(rf.collision3, newdata = df\_test)  
rfonethird<-mean(predicted\_collision == df\_test$COLLISION\_TYPE)  
rfonethird

## [1] 0.6535257

The model predicted correctly 65.45% of the time

#Examine the error rate on the test data.  
mean(predicted\_collision != df\_test$COLLISION\_TYPE)

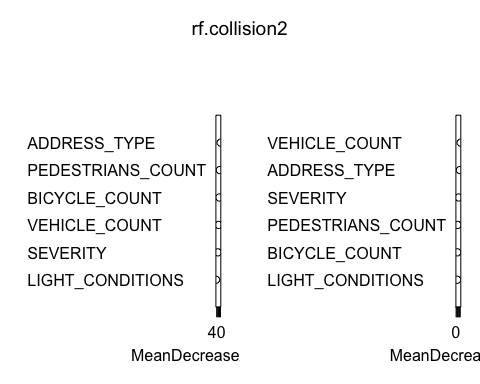
## [1] 0.3464743

The model has an Error rate of 34.54%

#Plot the importance of each predictors  
importance(rf.collision2)

## 1 2 3 4 5 6  
## VEHICLE\_COUNT 55.87557 13.057919 4.471004 1.921332 64.59262 69.197064  
## PEDESTRIANS\_COUNT 15.77901 7.382353 1.897393 -9.075702 20.69713 105.284159  
## BICYCLE\_COUNT 44.59010 80.546615 2.324701 3.074993 29.18157 60.432894  
## ADDRESS\_TYPE 131.13821 4.161340 8.627864 9.523937 85.96264 8.180127  
## SEVERITY 17.30953 12.493220 -11.072160 -7.854974 70.07429 11.688826  
## LIGHT\_CONDITIONS 28.93332 4.548970 1.126242 0.855323 10.09360 3.599495  
## 7 8 Unknown MeanDecreaseAccuracy  
## VEHICLE\_COUNT 52.62083 -1.403581 51.741952 79.06057  
## PEDESTRIANS\_COUNT 35.71605 -6.425537 40.461334 105.13001  
## BICYCLE\_COUNT 36.00242 0.000000 42.282660 95.97802  
## ADDRESS\_TYPE 58.29844 2.190480 4.833810 123.37900  
## SEVERITY 46.50244 -5.739345 6.185213 64.68045  
## LIGHT\_CONDITIONS 12.66071 -6.082562 19.378357 27.54376  
## MeanDecreaseGini  
## VEHICLE\_COUNT 13065.273  
## PEDESTRIANS\_COUNT 4506.968  
## BICYCLE\_COUNT 3632.098  
## ADDRESS\_TYPE 10768.749  
## SEVERITY 7304.188  
## LIGHT\_CONDITIONS 3355.665

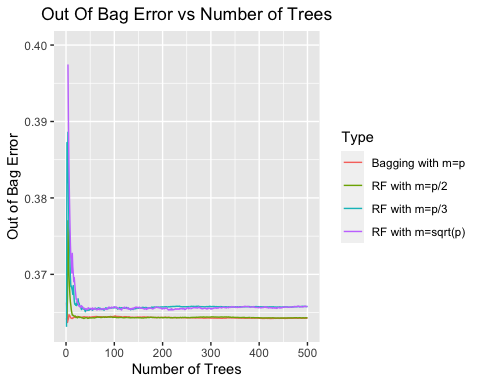
varImpPlot(rf.collision2)



# Plotting the OOB error vs number of trees:

collision.err <- data.frame(  
 Trees=1:bagging.collision$ntree,  
 Error=c(bagging.collision$err.rate[,"OOB"],rf.collision1$err.rate[,"OOB"], rf.collision2$err.rate[,"OOB"],rf.collision3$err.rate[,"OOB"]),  
 Type=rep(c("Bagging with m=p", "RF with m=p/2", "RF with m=sqrt(p)","RF with m=p/3"), each=bagging.collision$ntree)  
)

ggplot(data=collision.err, aes(x=Trees, y=Error)) + geom\_line(aes(color=Type)) + ggtitle("Out Of Bag Error vs Number of Trees") + xlim(0,500)+ylim(0.363,0.40) +ylab("Out of Bag Error")+xlab("Number of Trees")+theme(plot.title = element\_text(hjust=0.5))



Among the Bagging models, the model with the subset of p/3 has predicted well with the accuracy of 65.45%

#Boosting Method

#GBM was executed at interaction depths 1,2,3,4 and with a shrinkage parameter of 0.01 by passing the distribution as multinomial  
  
df1.boost1 = gbm(COLLISION\_TYPE~VEHICLE\_COUNT+PEDESTRIANS\_COUNT+BICYCLE\_COUNT+ADDRESS\_TYPE+SEVERITY+LIGHT\_CONDITIONS, data = df\_train, distribution = "multinomial",n.trees = 500, interaction.depth = 1, shrinkage = 0.01)

## Warning: Setting `distribution = "multinomial"` is ill-advised as it is  
## currently broken. It exists only for backwards compatibility. Use at your own  
## risk.

df1.boost2 = gbm(COLLISION\_TYPE~VEHICLE\_COUNT+PEDESTRIANS\_COUNT+BICYCLE\_COUNT+ADDRESS\_TYPE+SEVERITY+LIGHT\_CONDITIONS, data = df\_train, distribution = "multinomial",n.trees = 500, interaction.depth = 2, shrinkage = 0.01)

## Warning: Setting `distribution = "multinomial"` is ill-advised as it is  
## currently broken. It exists only for backwards compatibility. Use at your own  
## risk.

df1.boost3 = gbm(COLLISION\_TYPE~VEHICLE\_COUNT+PEDESTRIANS\_COUNT+BICYCLE\_COUNT+ADDRESS\_TYPE+SEVERITY+LIGHT\_CONDITIONS, data = df\_train, distribution = "multinomial",n.trees = 500, interaction.depth = 3, shrinkage = 0.01)

## Warning: Setting `distribution = "multinomial"` is ill-advised as it is  
## currently broken. It exists only for backwards compatibility. Use at your own  
## risk.

df1.boost4 = gbm(COLLISION\_TYPE~VEHICLE\_COUNT+PEDESTRIANS\_COUNT+BICYCLE\_COUNT+ADDRESS\_TYPE+SEVERITY+LIGHT\_CONDITIONS, data = df\_train, distribution = "multinomial",n.trees = 500, interaction.depth = 4, shrinkage = 0.01)

## Warning: Setting `distribution = "multinomial"` is ill-advised as it is  
## currently broken. It exists only for backwards compatibility. Use at your own  
## risk.

#Predicting on test data.  
df1.boost1.predict <- predict(df1.boost1, newdata = df\_test, type = "response", n.trees = 500)  
df1.boost2.predict <- predict(df1.boost2, newdata = df\_test, type = "response", n.trees = 500)  
df1.boost3.predict <- predict(df1.boost3, newdata = df\_test, type = "response", n.trees = 500)  
df1.boost4.predict <- predict(df1.boost4, newdata = df\_test, type = "response", n.trees = 500)

#Examine the boosting model1 on the test data.  
labels1 = colnames(df1.boost1.predict)[apply(df1.boost1.predict, 1, which.max)]  
table(df\_test$COLLISION\_TYPE, labels1)

## labels1  
## 1 2 5 6 7 Unknown  
## 1 11749 6 2164 27 958 0  
## 2 0 2800 0 1 0 0  
## 3 208 0 417 2 359 1  
## 4 5034 3 1201 18 555 0  
## 5 608 0 14538 34 1159 3  
## 6 0 4 0 3541 0 1  
## 7 2120 3 6452 22 4431 4  
## 8 1639 1 5101 5 919 1  
## Unknown 535 22 1484 35 464 22105

boost1<-mean(df\_test$COLLISION\_TYPE == labels1)  
boost1

## [1] 0.6520599

#Examine the boosting model2 on the test data.  
labels2 = colnames(df1.boost2.predict)[apply(df1.boost2.predict, 1, which.max)]  
table(df\_test$COLLISION\_TYPE, labels2)

## labels2  
## 1 2 3 5 6 7 Unknown  
## 1 11769 6 1 2096 6 1026 0  
## 2 0 2800 0 0 1 0 0  
## 3 210 0 2 398 0 376 1  
## 4 5047 3 1 1169 5 586 0  
## 5 608 0 4 13992 30 1706 2  
## 6 16 4 0 0 3524 1 1  
## 7 2120 3 3 5755 18 5129 4  
## 8 1641 1 1 4928 3 1091 1  
## Unknown 535 22 1 1428 34 520 22105

boost2<-mean(df\_test$COLLISION\_TYPE == labels2)  
boost2

## [1] 0.6537902

#Examine the boosting model3 on the test data.  
labels3 = colnames(df1.boost3.predict)[apply(df1.boost3.predict, 1, which.max)]  
table(df\_test$COLLISION\_TYPE, labels3)

## labels3  
## 1 2 3 5 6 7 Unknown  
## 1 11705 6 1 2160 6 1026 0  
## 2 0 2800 0 0 1 0 0  
## 3 205 0 2 403 0 376 1  
## 4 5020 3 1 1196 5 586 0  
## 5 486 0 4 14114 30 1706 2  
## 6 16 4 0 0 3524 1 1  
## 7 2080 3 3 5800 18 5126 2  
## 8 1608 1 1 4961 3 1091 1  
## Unknown 523 22 1 1441 34 519 22105

boost3<-mean(df\_test$COLLISION\_TYPE == labels3)  
boost3

## [1] 0.6533973

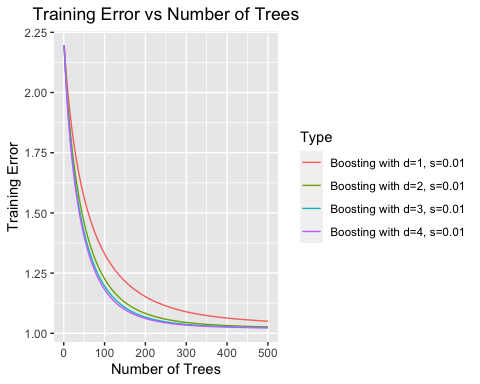
#Examine the boosting model4 on the test data.  
labels4 = colnames(df1.boost4.predict)[apply(df1.boost4.predict, 1, which.max)]  
table(df\_test$COLLISION\_TYPE, labels3)

## labels3  
## 1 2 3 5 6 7 Unknown  
## 1 11705s 6 1 2160 6 1026 0  
## 2 0 2800 0 0 1 0 0  
## 3 205 0 2 403 0 376 1  
## 4 5020 3 1 1196 5 586 0  
## 5 486 0 4 14114 30 1706 2  
## 6 16 4 0 0 3524 1 1  
## 7 2080 3 3 5800 18 5126 2  
## 8 1608 1 1 4961 3 1091 1  
## Unknown 523 22 1 1441 34 519 22105

boost4<-mean(df\_test$COLLISION\_TYPE == labels4)  
boost4

## [1] 0.6543964

# Plotting the training error vs number of trees:  
bagging\_boosting1.err <- data.frame(  
 Trees=1:500,  
 Error=c(df1.boost1$train.error, df1.boost2$train.error, df1.boost3$train.error,df1.boost4$train.error),  
 Type=rep(c( "Boosting with d=1, s=0.01", "Boosting with d=2, s=0.01", "Boosting with d=3, s=0.01","Boosting with d=4, s=0.01"), each=500)  
)  
  
ggplot(data=bagging\_boosting1.err, aes(x=Trees, y=Error)) + geom\_line(aes(color=Type)) + ggtitle("Training Error vs Number of Trees")+ylab("Training Error")+xlab("Number of Trees")+theme(plot.title = element\_text(hjust=0.5))



The best among the boosting model is the model with depth 3 and shrinkage rate 0.01 has performed well with the accuracy of 65.43%

#Logistic Regression

#Fit multinomial logistic regression on training dataset  
logistic\_model <- multinom(COLLISION\_TYPE~VEHICLE\_COUNT+PEDESTRIANS\_COUNT+BICYCLE\_COUNT+ADDRESS\_TYPE+SEVERITY+LIGHT\_CONDITIONS, data=df\_train)

## # weights: 333 (288 variable)  
## initial value 299040.067751   
## iter 10 value 173479.883656  
## iter 20 value 163801.195936  
## iter 30 value 161494.073285  
## iter 40 value 156914.218883  
## iter 50 value 151610.626880  
## iter 60 value 147863.192078  
## iter 70 value 143955.133495  
## iter 80 value 141717.921292  
## iter 90 value 140360.544496  
## iter 100 value 139834.919053  
## final value 139834.919053   
## stopped after 100 iterations

#Predict it on test data  
predicted\_collision<-predict(logistic\_model, df\_test, type="class")

#Examine the model's performance on the test data.  
table(predicted\_collision, df\_test$COLLISION\_TYPE)

##   
## predicted\_collision 1 2 3 4 5 6 7 8 Unknown  
## 1 11698 0 204 5018 485 9 2079 1607 523  
## 2 6 2800 0 3 0 4 3 0 22  
## 3 0 0 0 0 0 0 0 0 0  
## 4 0 0 0 0 0 0 0 0 0  
## 5 2225 0 417 1223 14675 3 6486 5128 1495  
## 6 13 1 1 7 26 3529 15 4 33  
## 7 961 0 360 559 1151 0 4444 925 466  
## 8 0 0 0 0 0 0 0 0 0  
## Unknown 1 0 5 1 5 1 5 2 22106

logistic <-mean(predicted\_collision == df\_test$COLLISION\_TYPE)  
logistic

## [1] 0.6530297

The logistic regression model has accuracy of 65.3%

types <- c("Decision Tree","RF-(p/3)", "Boosting (d=4,s=0.01)", "Logistic Regression")  
values <- c(65.16, 65.45, 65.43, 65.30)  
x = c(1,2,3,4)  
plot(values, type = 'o', ylim = c(64, 66), at = x,labels = types,xlab="Models",ylab="Performance of the models",main="Comparing the performance of models")

## Warning in plot.window(...): "at" is not a graphical parameter

## Warning in plot.window(...): "labels" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "at" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "labels" is not a graphical parameter

## Warning in box(...): "at" is not a graphical parameter

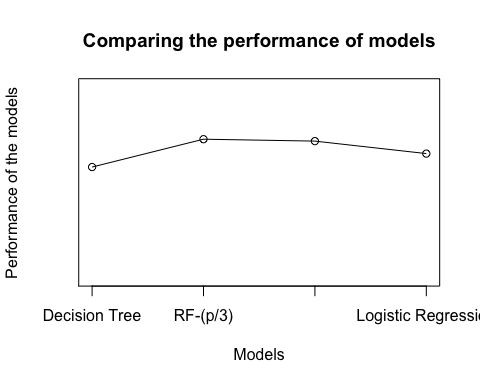
## Warning in box(...): "labels" is not a graphical parameter

## Warning in title(...): "at" is not a graphical parameter

## Warning in title(...): "labels" is not a graphical parameter

text(x=types,y=values,labels=as.character(values))

## Warning in xy.coords(x, y, recycle = TRUE, setLab = FALSE): NAs introduced by  
## coercion



On comparing, all the models has exhibited similar performance but among them random forest with the subset of p/3 has the performnace rate 65.45%