Model Comparison

Rajat Chandna

August 17, 2018

Contents


```
## 'data.frame':    500 obs. of  14 variables:
##  $ SUB_ID   : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ SITE_ID  : int  1 4 6 6 1 5 5 1 1 4 ...
##  $ PHY_ID   : int  14 284 305 309 37 299 302 36 8 282 ...
##  $ PRIORFRAC: Factor w/ 2 levels "0","1": 1 1 2 1 1 2 1 2 2 1 ...
##  $ AGE      : int  62 65 88 82 61 67 84 82 86 58 ...
##  $ WEIGHT   : num  70.3 87.1 50.8 62.1 68 68 50.8 40.8 62.6 63.5 ...
##  $ HEIGHT   : int  158 160 157 160 152 161 150 153 156 166 ...
##  $ BMI      : num  28.2 34 20.6 24.3 29.4 ...
##  $ PREMENO  : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ MOMFRAC  : Factor w/ 2 levels "0","1": 1 1 2 1 1 1 1 1 1 1 ...
##  $ ARMASSIST: Factor w/ 2 levels "0","1": 1 1 2 1 1 1 1 1 1 1 ...
##  $ SMOKE    : Factor w/ 2 levels "0","1": 1 1 1 1 1 2 1 1 1 1 ...
##  $ RATERISK : Factor w/ 3 levels "1","2","3": 2 2 1 1 2 2 1 2 2 1 ...
##  $ FRACTURE : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
```

Create Train and Validation Datasets

```
set.seed(999)
validation_index = createDataPartition(dataset$FRACTURE, p=0.70, list=FALSE)
validationData = dataset[-validation_index,c(4:14)]
trainingData = dataset[validation_index,c(4:14)]

table(dataset$FRACTURE)
```

```
##
##   0   1
## 375 125
```

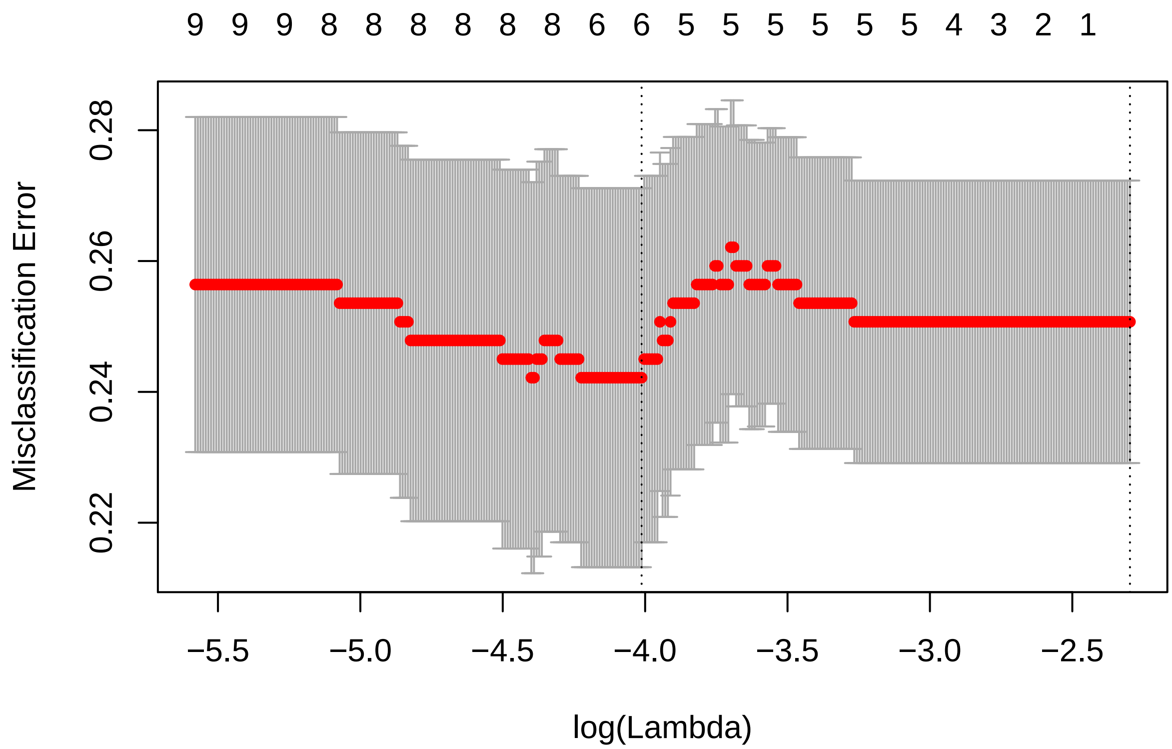
```
table(trainingData$FRACTURE)
```

```
##
##   0   1
## 263  88
```

```r
table(validationData$FRACTURE)
```

```
##
##   0   1
## 112  37
```

```r
set.seed(999)

## Formatting Test Data Set
# Recode Rate Risk Variable since its ordinal and we donot want to loose its info if it gets
# coded as nominal variable before running the Model
validationData$RATERISK <- factor(validationData$RATERISK, levels = c(1,2,3), ordered = T)

xfactors_test <- model.matrix(validationData$FRACTURE ~ validationData$PRIORFRAC + validationData$PREMEN
x_test <- as.matrix(data.frame(validationData$AGE, validationData$WEIGHT, validationData$HEIGHT, valida

## Formatting Training Data Set
trainingData$RATERISK <- factor(trainingData$RATERISK, levels = c(1,2,3), ordered = T)
xfactors_train <- model.matrix(trainingData$FRACTURE ~ trainingData$PRIORFRAC + trainingData$PREMENO + t
x_train <- as.matrix(data.frame(trainingData$AGE, trainingData$WEIGHT, trainingData$HEIGHT, trainingDat

# Doing Cross validation to find the best fitting model based upon Lasso
cvfit <- cv.glmnet(x_train, y=trainingData$FRACTURE, family = "binomial", type.measure = "class", nlamb
plot(cvfit)
```

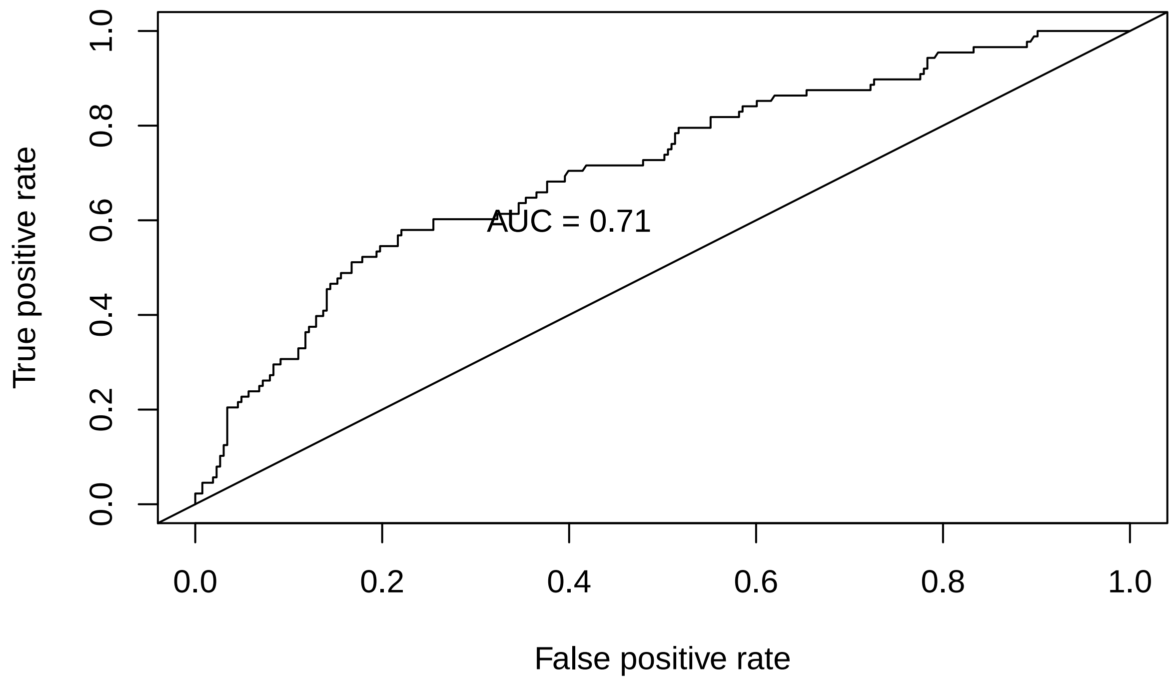



```r
# Model with Lowest Lambda is shrinking all the coefficients, hence selecting lambda based upon
# Test Set AUC and EDA Results
#cvfit$glmnet.fit
coef(cvfit, s="lambda.min")
```

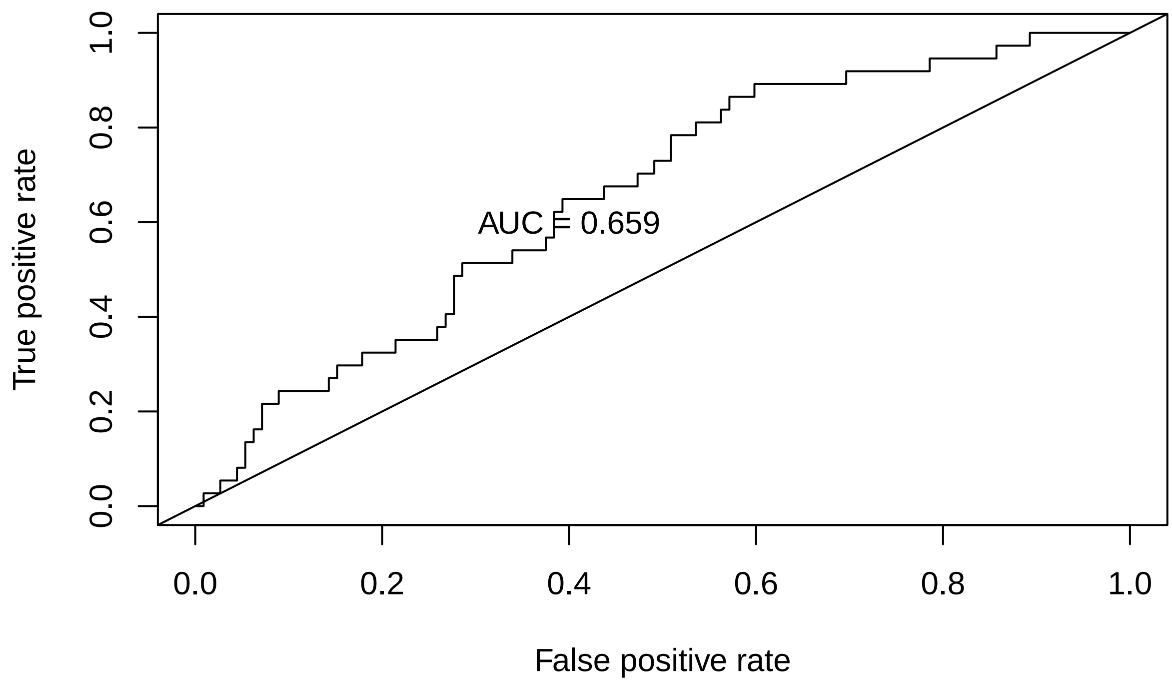
```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                                 1
## (Intercept)              1.52902669
## trainingData.AGE         0.03598544
## trainingData.WEIGHT      .
## trainingData.HEIGHT     -0.03340871
## trainingData.BMI         .
## trainingData.PRIORFRAC1  0.15180349
## trainingData.PREMENO1    .
## trainingData.MOMFRAC1    0.04035537
## trainingData.ARMASSIST1  0.52512963
## trainingData.SMOKE1      .
## trainingData.RATERISK.L  0.33991586
## trainingData.RATERISK.Q  .
```

```r
# Fitting the best model based upon selected lambda
fit <- glmnet(x_train, y=trainingData$FRACTURE, family="binomial", alpha = 1, lambda = cvfit$lambda.min

# First Predicting the responses on training data set itself
fit.pred <- predict(fit, newx = x_train, type = "response")
```

```r
#Create ROC curves for training Data Set
pred <- prediction(fit.pred[,1], trainingData$FRACTURE)
roc.perf = performance(pred, measure = "tpr", x.measure = "fpr")
auc.train <- performance(pred, measure = "auc")
auc.train <- auc.train@y.values

##Plot ROC for training Set
plot(roc.perf)
abline(a=0, b= 1) #Ref line indicating poor performance
text(x = .40, y = .6,paste("AUC = ", round(auc.train[[1]],3), sep = ""))
```




```r
#Run model from training set on validation Set
fit.pred1 <- predict(fit, newx = x_test, type = "response")

#ROC curves
pred1 <- prediction(fit.pred1[,1], validationData$FRACTURE)
roc.perf1 = performance(pred1, measure = "tpr", x.measure = "fpr")
auc.val1 <- performance(pred1, measure = "auc")
auc.val1 <- auc.val1@y.values
plot(roc.perf1)
abline(a=0, b= 1)
text(x = .40, y = .6,paste("AUC = ", round(auc.val1[[1]],3), sep = ""))
```




```
#confusion matrix
pdata <- predict(fit, newx = x_test, type = "response")
pdata_logical <- pdata[, 1] > 0.5
confusionMatrix(data = as.factor(as.numeric(pdata_logical)), reference = as.factor(as.numeric(validatio
```

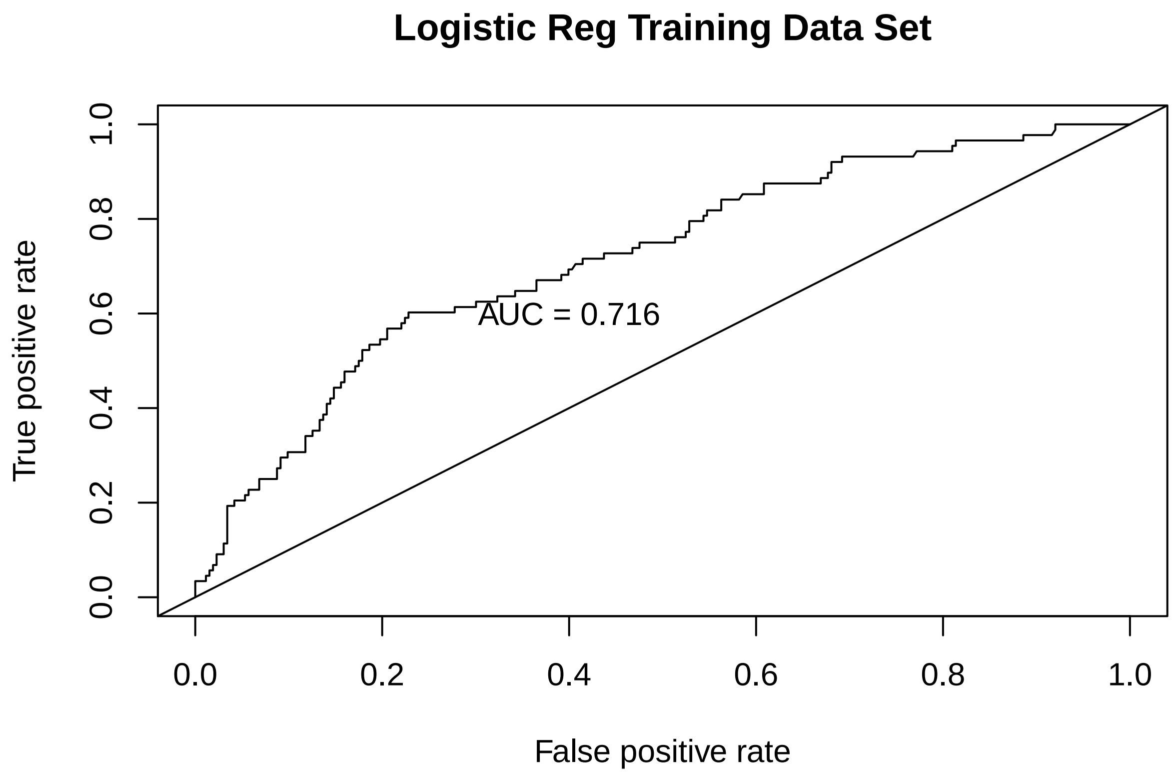
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 108  35
##          1   4   2
##
##                Accuracy : 0.7383
##                  95% CI : (0.66, 0.8068)
##     No Information Rate : 0.7517
##     P-Value [Acc > NIR] : 0.6866
##
##                   Kappa : 0.0255
##  Mcnemar's Test P-Value : 1.556e-06
##
##             Sensitivity : 0.96429
##             Specificity : 0.05405
##          Pos Pred Value : 0.75524
##          Neg Pred Value : 0.33333
##              Prevalence : 0.75168
##          Detection Rate : 0.72483
```

```
##    Detection Prevalence : 0.95973
##       Balanced Accuracy : 0.50917
##
##        'Positive' Class : 0
##
```

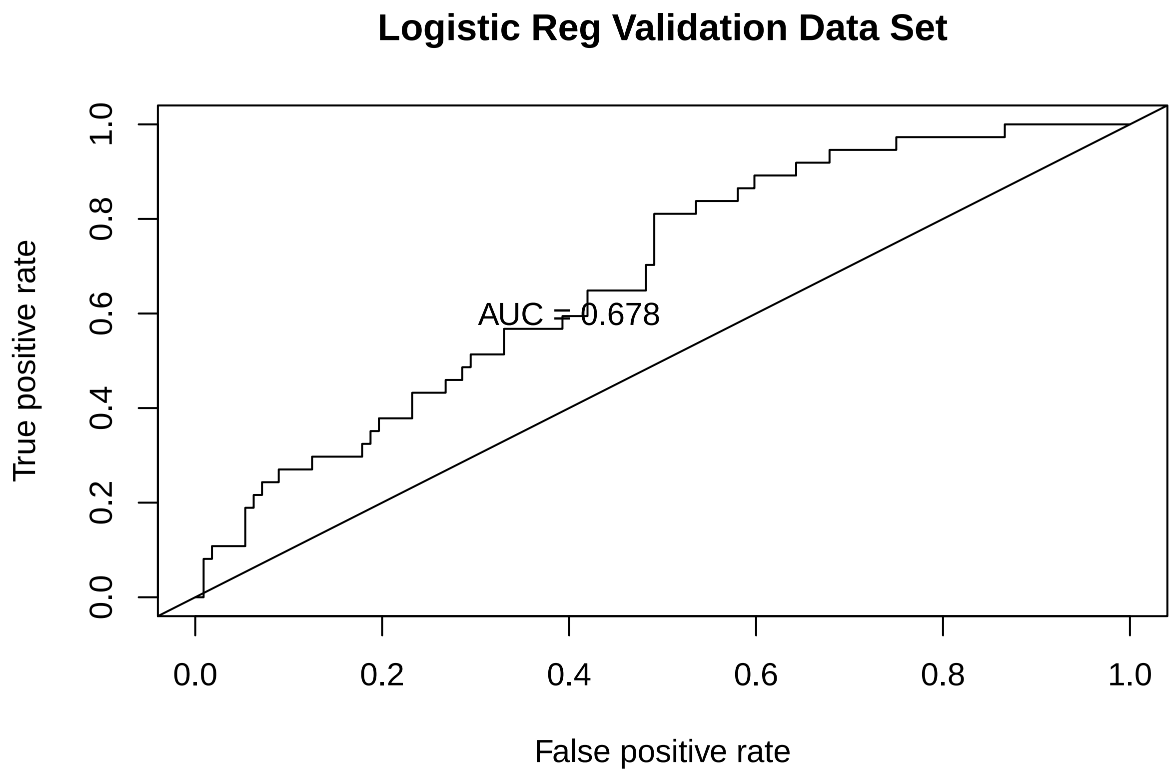
```r
#mydata <- dataset[, c(4:14)] %>% dplyr::select_if(is.numeric)
#predictors <- colnames(mydata)
#mydata <- mydata %>%
#  mutate(logit = log(probabilities/(1-probabilities))) %>%
#  gather(key = "predictors", value = "predictor.value", -logit)
```

Run Normal Logit Model with Identified Predictors

```r
set.seed(999)

logit.fit <- glm(FRACTURE ~ AGE + HEIGHT + PRIORFRAC + MOMFRAC + ARMASSIST + RATERISK + SMOKE, data = t
# First Predicting the responses on training data set itself
logistic.fit.pred.train <- predict(logit.fit, newdata=trainingData, type = "response")

#Create ROC curves for training Data Set
pred.train <- prediction(logistic.fit.pred.train, trainingData$FRACTURE)
roc.perf = performance(pred.train, measure = "tpr", x.measure = "fpr")
auc.train <- performance(pred.train, measure = "auc")
auc.train <- auc.train@y.values

##Plot ROC for training Set
plot(roc.perf, main="Logistic Reg Training Data Set")
abline(a=0, b= 1) #Ref line indicating poor performance
text(x = .40, y = .6,paste("AUC = ", round(auc.train[[1]],3), sep = ""))
```




```r
#Run model from training set on validation Set
logistic.fit.pred.test <- predict(logit.fit, newdata=validationData, type = "response")

#ROC curves
pred.test <- prediction(logistic.fit.pred.test, validationData$FRACTURE)
roc.perf1 = performance(pred.test, measure = "tpr", x.measure = "fpr")
auc.val1 <- performance(pred.test, measure = "auc")
auc.val1 <- auc.val1@y.values
plot(roc.perf1, main="Logistic Reg Validation Data Set")
abline(a=0, b= 1)
text(x = .40, y = .6,paste("AUC = ", round(auc.val1[[1]],3), sep = ""))
```




```
#confusion matrix
pdata_logical <-  logistic.fit.pred.test > 0.5
confusionMatrix(data = as.factor(as.numeric(pdata_logical)), reference = as.factor(as.numeric(validatio
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 106  32
##          1   6   5
##
##                Accuracy : 0.745
##                  95% CI : (0.6672, 0.8128)
##     No Information Rate : 0.7517
##     P-Value [Acc > NIR] : 0.6175
##
##                   Kappa : 0.1067
##  Mcnemar's Test P-Value : 5.002e-05
##
##             Sensitivity : 0.9464
##             Specificity : 0.1351
##          Pos Pred Value : 0.7681
##          Neg Pred Value : 0.4545
##              Prevalence : 0.7517
##          Detection Rate : 0.7114
##    Detection Prevalence : 0.9262
```

```
##       Balanced Accuracy : 0.5408
##
##        'Positive' Class : 0
##
```

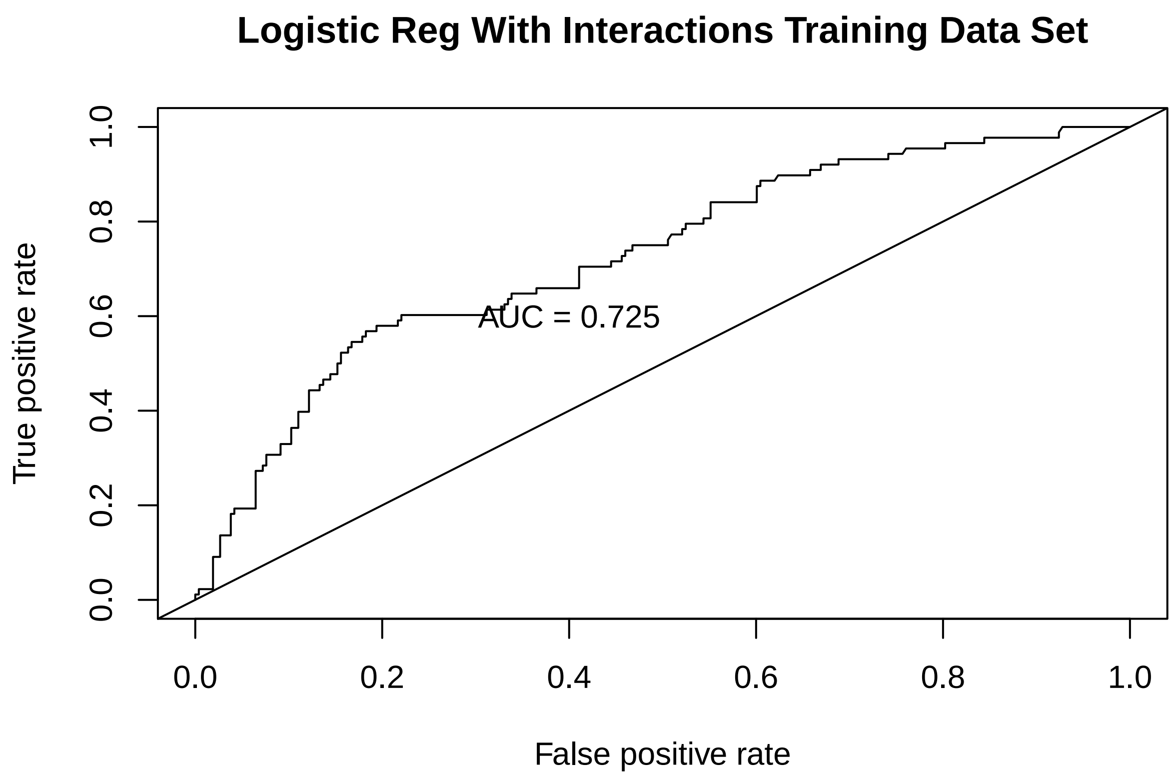
Add Interactions to Normal logit

```r
set.seed(999)
# Since top 3 predictors are Age, PriorFrac and RISK, adding model complexity
# via interactions
logit.fit.interactions <- glm(FRACTURE ~ AGE + HEIGHT + PRIORFRAC + MOMFRAC + ARMASSIST + RATERISK + SMO
summary(logit.fit.interactions)
```

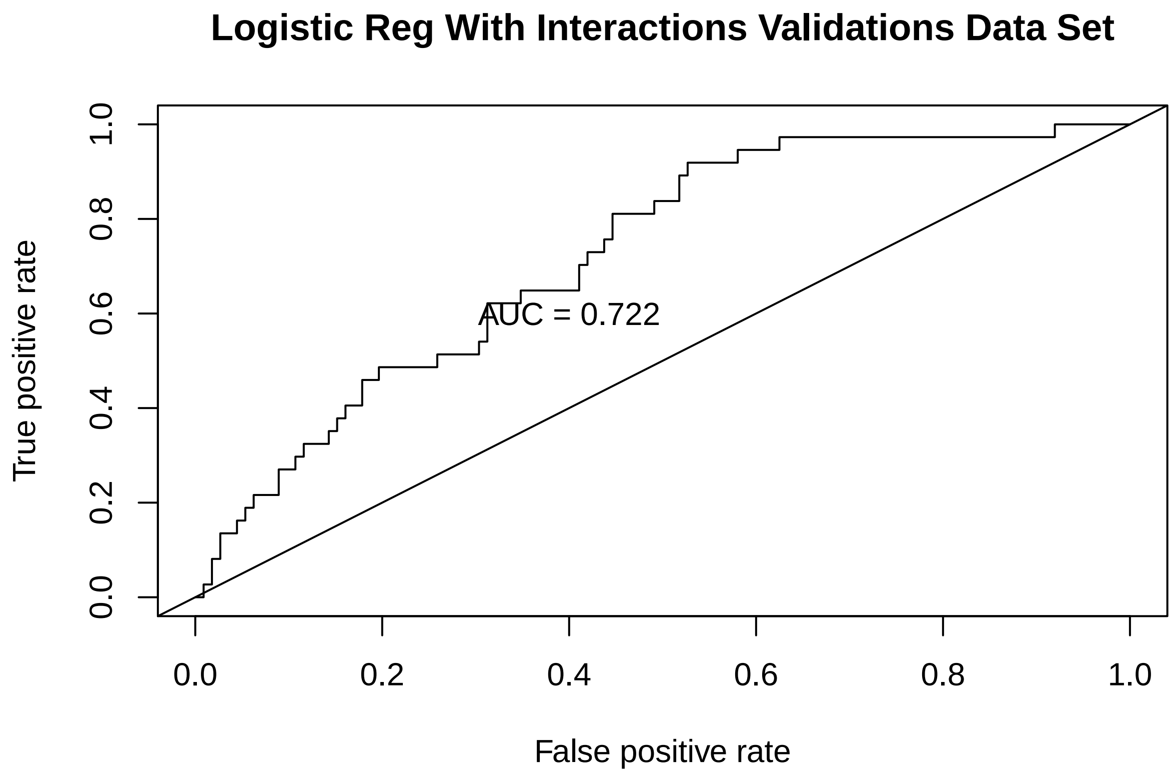
```
##
## Call:
## glm(formula = FRACTURE ~ AGE + HEIGHT + PRIORFRAC + MOMFRAC +
##     ARMASSIST + RATERISK + SMOKE + AGE:PRIORFRAC + RATERISK:AGE +
##     MOMFRAC:ARMASSIST, family = binomial(link = "logit"), data = trainingData)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5642  -0.7471  -0.5468   0.2803   2.3844
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)          1.72515    3.99154   0.432  0.66559
## AGE                  0.06691    0.02059   3.250  0.00116 **
## HEIGHT              -0.04935    0.02196  -2.247  0.02464 *
## PRIORFRAC1           3.90581    2.26574   1.724  0.08473 .
## MOMFRAC1             0.75783    0.50958   1.487  0.13697
## ARMASSIST1           0.82575    0.30351   2.721  0.00651 **
## RATERISK.L           1.72049    2.00908   0.856  0.39180
## RATERISK.Q          -1.35206    1.79608  -0.753  0.45158
## SMOKE1              -0.22408    0.49927  -0.449  0.65356
## AGE:PRIORFRAC1      -0.05123    0.03130  -1.637  0.10164
## AGE:RATERISK.L      -0.01646    0.02796  -0.589  0.55608
## AGE:RATERISK.Q       0.01768    0.02517   0.703  0.48227
## MOMFRAC1:ARMASSIST1 -0.77400    0.76657  -1.010  0.31264
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 395.31  on 350  degrees of freedom
## Residual deviance: 350.93  on 338  degrees of freedom
## AIC: 376.93
##
## Number of Fisher Scoring iterations: 5
```

```r
# First Predicting the responses on training data set itself
logistic.fit.pred.train.interaction <- predict(logit.fit.interactions, newdata=trainingData, type = "re
```

```r
#Create ROC curves for training Data Set
pred.train.interaction <- prediction(logistic.fit.pred.train.interaction, trainingData$FRACTURE)
roc.perf = performance(pred.train.interaction, measure = "tpr", x.measure = "fpr")
auc.train <- performance(pred.train.interaction, measure = "auc")
auc.train <- auc.train@y.values

##Plot ROC for training Set
plot(roc.perf, main="Logistic Reg With Interactions Training Data Set")
abline(a=0, b= 1) #Ref line indicating poor performance
text(x = .40, y = .6,paste("AUC = ", round(auc.train[[1]],3), sep = ""))
```




```r
#Run model from training set on validation Set
logistic.fit.pred.test.interaction <- predict(logit.fit.interactions, newdata=validationData, type = "re

#ROC curves
pred.test.interaction <- prediction(logistic.fit.pred.test.interaction, validationData$FRACTURE)
roc.perf1 = performance(pred.test.interaction, measure = "tpr", x.measure = "fpr")
auc.val1 <- performance(pred.test.interaction, measure = "auc")
auc.val1 <- auc.val1@y.values
plot(roc.perf1, main="Logistic Reg With Interactions Validations Data Set")
abline(a=0, b= 1)
text(x = .40, y = .6,paste("AUC = ", round(auc.val1[[1]],3), sep = ""))
```




```r
#confusion matrix
pdata_logical <-  logistic.fit.pred.test.interaction > 0.5
confusionMatrix(data = as.factor(as.numeric(pdata_logical)), reference = as.factor(as.numeric(validatio
```

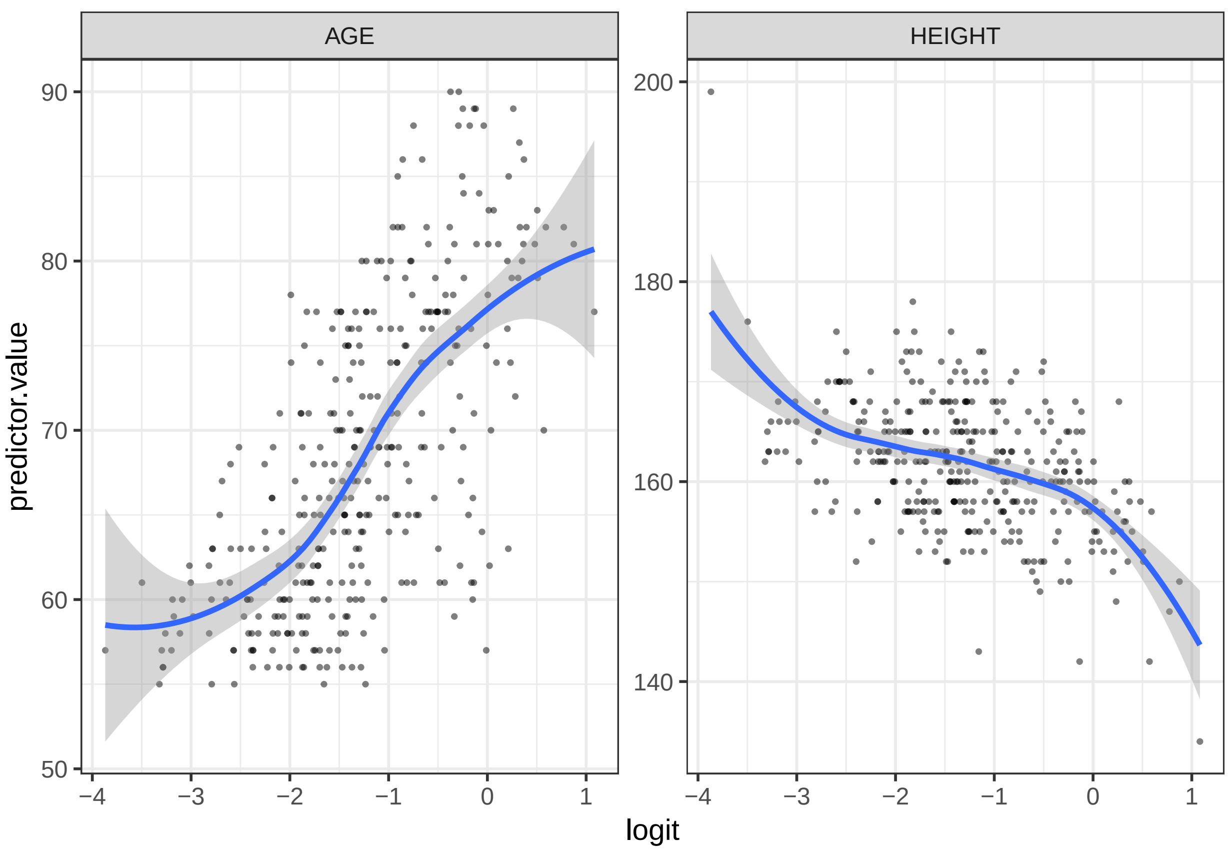
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 104  29
##          1   8   8
##
##                Accuracy : 0.7517
##                  95% CI : (0.6743, 0.8187)
##     No Information Rate : 0.7517
##     P-Value [Acc > NIR] : 0.544018
##
##                   Kappa : 0.1788
##  Mcnemar's Test P-Value : 0.001009
##
##             Sensitivity : 0.9286
##             Specificity : 0.2162
##          Pos Pred Value : 0.7820
##          Neg Pred Value : 0.5000
##              Prevalence : 0.7517
##          Detection Rate : 0.6980
##    Detection Prevalence : 0.8926
```

```
##        Balanced Accuracy : 0.5724
##
##         'Positive' Class : 0
##
```

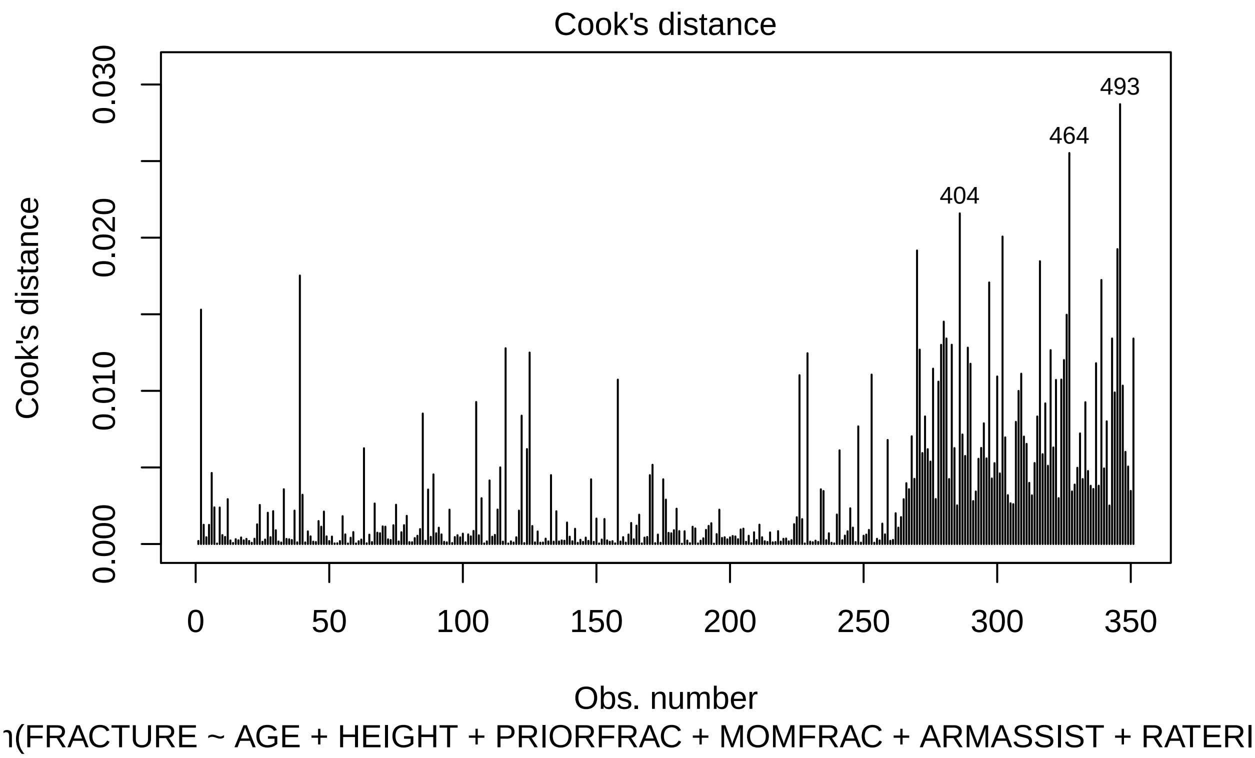
```r
# Checking the assumptions
probabilities <- predict(logit.fit.interactions, type = "response")
predicted.classes <- ifelse(probabilities > 0.5, "pos", "neg")
head(predicted.classes)
```

```
##     1     3     4     5     6     7
## "neg" "neg" "neg" "neg" "neg" "neg"
```

```r
# Linearity assumption
subNumericPred <- trainingData %>% dplyr::select(AGE, HEIGHT)
predictors <- colnames(subNumericPred)
subNumericPred <- subNumericPred %>%
                  mutate(logit = log(probabilities/(1-probabilities))) %>%
                  gather(key = "predictors", value = "predictor.value", -logit)

ggplot(subNumericPred, aes(logit, predictor.value)) +
                geom_point(size = 0.5, alpha = 0.5) +
                geom_smooth(method = "loess") +
                theme_bw() +
                facet_wrap(~predictors, scales = "free_y")
```



```r
plot(logit.fit.interactions, which = 4, id.n =3)
```

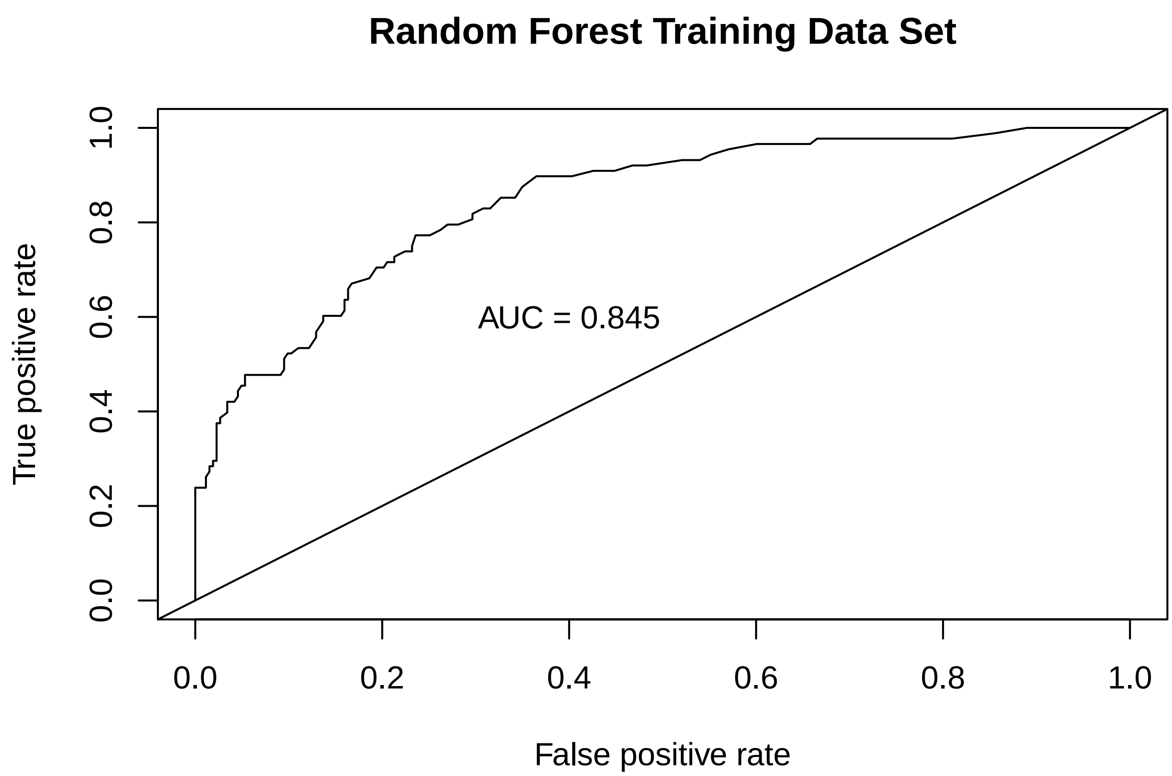


Running Random Forest Fit

```r
set.seed(999)

str(trainingData)
```

```
## 'data.frame':    351 obs. of  11 variables:
##  $ PRIORFRAC: Factor w/ 2 levels "0","1": 1 2 1 1 2 1 2 1 1 1 ...
##  $ AGE      : int  62 88 82 61 67 84 86 58 67 56 ...
##  $ WEIGHT   : num  70.3 50.8 62.1 68 68 ...
##  $ HEIGHT   : int  158 157 160 152 161 150 156 166 153 167 ...
##  $ BMI      : num  28.2 20.6 24.3 29.4 26.2 ...
##  $ PREMENO  : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ MOMFRAC  : Factor w/ 2 levels "0","1": 1 2 1 1 1 1 1 1 2 1 ...
##  $ ARMASSIST: Factor w/ 2 levels "0","1": 1 2 1 1 1 1 1 1 1 2 ...
##  $ SMOKE    : Factor w/ 2 levels "0","1": 1 1 1 1 2 1 1 1 2 2 ...
##  $ RATERISK : Ord.factor w/ 3 levels "1"<"2"<"3": 2 1 1 2 2 1 2 1 1 2 ...
##  $ FRACTURE : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
```

```r
rf.fit <- randomForest(FRACTURE ~ ., data=trainingData, mtry=4, ntree=500, maxnodes = 12, importance=T)

rf.fit.pred.train <- predict(rf.fit, newdata=trainingData, type="prob")
pred.train.rf <- prediction(rf.fit.pred.train[,2], trainingData$FRACTURE)
roc.perf = performance(pred.train.rf, measure = "tpr", x.measure = "fpr")
auc.train <- performance(pred.train.rf, measure = "auc")
auc.train <- auc.train@y.values
plot(roc.perf, main="Random Forest Training Data Set")
abline(a=0, b= 1)
text(x = .40, y = .6,paste("AUC = ", round(auc.train[[1]],3), sep = ""))
```

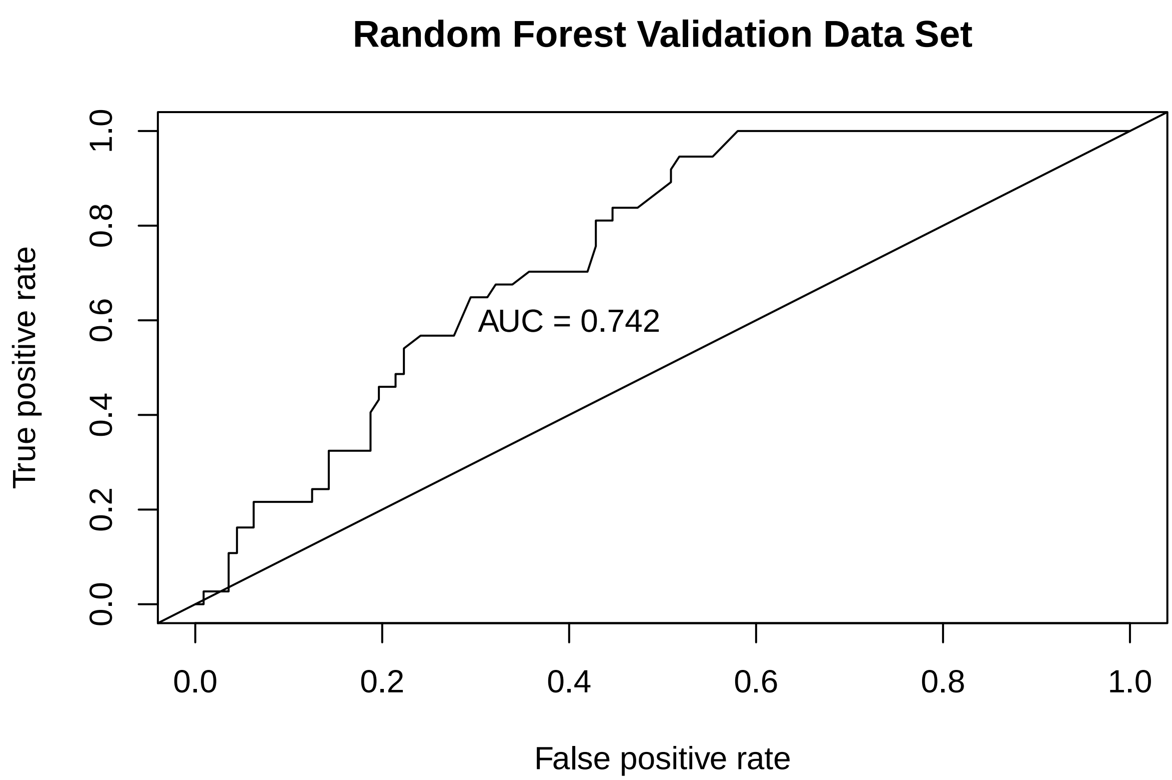



```r
#confusion matrix Training
pdata_logical_train <-  (rf.fit.pred.train[,2] >= 0.5)
confusionMatrix(data = as.factor(as.numeric(pdata_logical_train)), reference = as.factor(as.numeric(tra
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 263  76
##          1   0  12
##
##                Accuracy : 0.7835
##                  95% CI : (0.7367, 0.8254)
##     No Information Rate : 0.7493
```

```
##     P-Value [Acc > NIR] : 0.07672
##
##                   Kappa : 0.1913
##  Mcnemar's Test P-Value : < 2e-16
##
##             Sensitivity : 1.0000
##             Specificity : 0.1364
##          Pos Pred Value : 0.7758
##          Neg Pred Value : 1.0000
##              Prevalence : 0.7493
##          Detection Rate : 0.7493
##    Detection Prevalence : 0.9658
##       Balanced Accuracy : 0.5682
##
##        'Positive' Class : 0
##
```

```
rf.fit.pred.test <- predict(rf.fit, newdata=validationData, type="prob")
pred.test.rf <- prediction(rf.fit.pred.test[,2], validationData$FRACTURE)
roc.perf = performance(pred.test.rf, measure = "tpr", x.measure = "fpr")
auc.train <- performance(pred.test.rf, measure = "auc")
auc.train <- auc.train@y.values
plot(roc.perf, main="Random Forest Validation Data Set")
abline(a=0, b= 1)
text(x = .40, y = .6,paste("AUC = ", round(auc.train[[1]],3), sep = ""))
```



```r
#confusion matrix Test
pdata_logical <-  rf.fit.pred.test[,2] > 0.5
confusionMatrix(data = as.factor(as.numeric(pdata_logical)), reference = as.factor(as.numeric(validatio
```

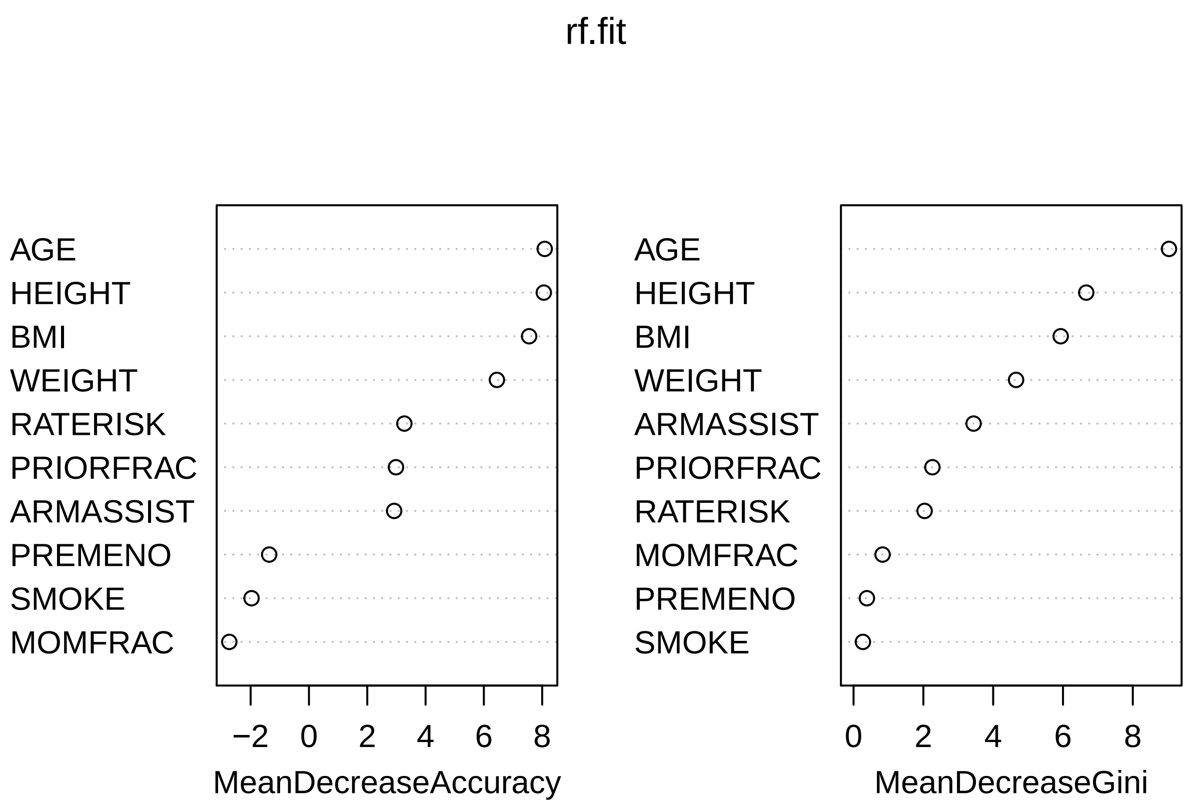
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 111  37
##          1   1   0
##
##                Accuracy : 0.745
##                  95% CI : (0.6672, 0.8128)
##     No Information Rate : 0.7517
##     P-Value [Acc > NIR] : 0.6175
##
##                   Kappa : -0.0132
##  Mcnemar's Test P-Value : 1.365e-08
##
##             Sensitivity : 0.9911
##             Specificity : 0.0000
##          Pos Pred Value : 0.7500
##          Neg Pred Value : 0.0000
##              Prevalence : 0.7517
##          Detection Rate : 0.7450
##    Detection Prevalence : 0.9933
##       Balanced Accuracy : 0.4955
##
##        'Positive' Class : 0
##
```

```r
#confusion matrix Test Lower Cutoff
pdata_logical_lowercf <-  rf.fit.pred.test[,2] >= 0.3
confusionMatrix(data = as.factor(as.numeric(pdata_logical_lowercf)), reference = as.factor(as.numeric(va
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 102  29
##          1  10   8
##
##                Accuracy : 0.7383
##                  95% CI : (0.66, 0.8068)
##     No Information Rate : 0.7517
##     P-Value [Acc > NIR] : 0.686582
##
##                   Kappa : 0.1533
##  Mcnemar's Test P-Value : 0.003948
##
##             Sensitivity : 0.9107
##             Specificity : 0.2162
##          Pos Pred Value : 0.7786
```

```
##          Neg Pred Value : 0.4444
##              Prevalence : 0.7517
##          Detection Rate : 0.6846
##    Detection Prevalence : 0.8792
##       Balanced Accuracy : 0.5635
##
##        'Positive' Class : 0
##
```

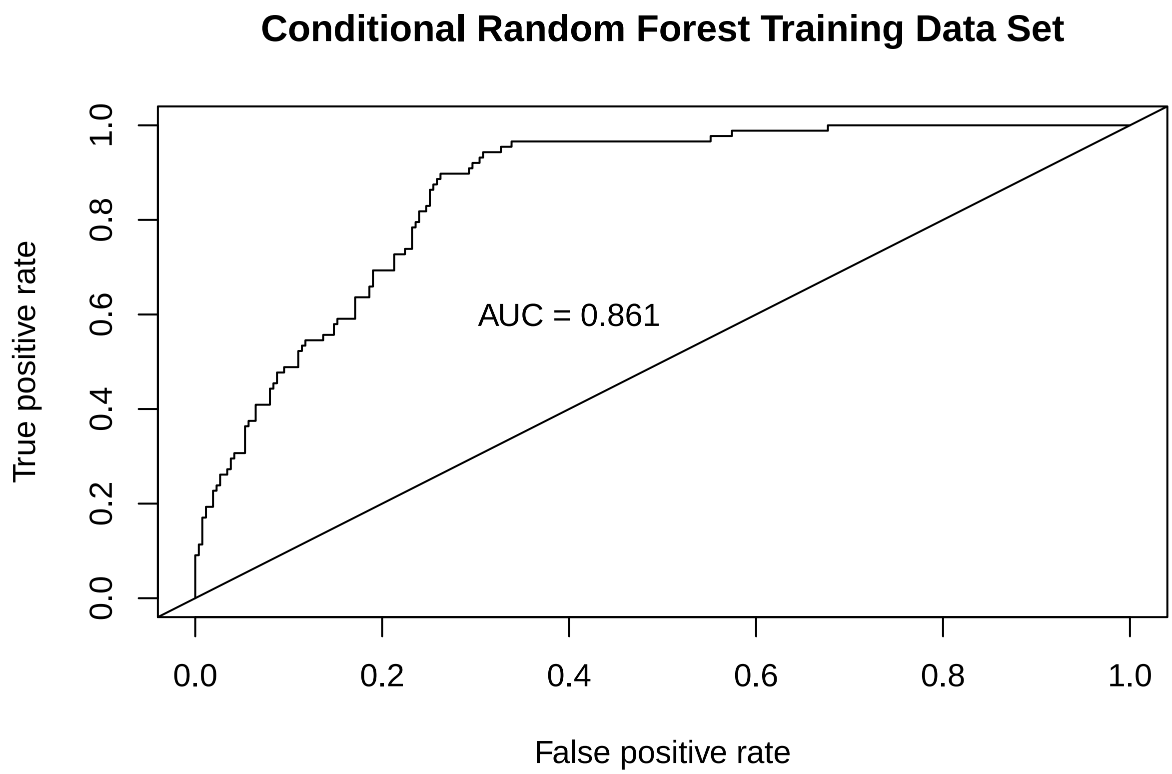
```
varImpPlot(rf.fit)
```



Running Conditional Random Forest Fit

```
set.seed(999)

crf.fit <- cforest(FRACTURE ~ ., data=trainingData, control=cforest_unbiased(ntree=500))

crf.fit.pred.train <- predict(crf.fit, newdata=trainingData, OOB = TRUE, type="prob")
unlist.Pred.train <- matrix(unlist(crf.fit.pred.train), ncol=2,  byrow = TRUE)
pred.train.crf <- prediction(unlist.Pred.train[,2], trainingData$FRACTURE)
roc.perf = performance(pred.train.crf, measure = "tpr", x.measure = "fpr")
auc.train <- performance(pred.train.crf, measure = "auc")
auc.train <- auc.train@y.values
```

```r
plot(roc.perf, main="Conditional Random Forest Training Data Set")
abline(a=0, b= 1)
text(x = .40, y = .6,paste("AUC = ", round(auc.train[[1]],3), sep = ""))
```

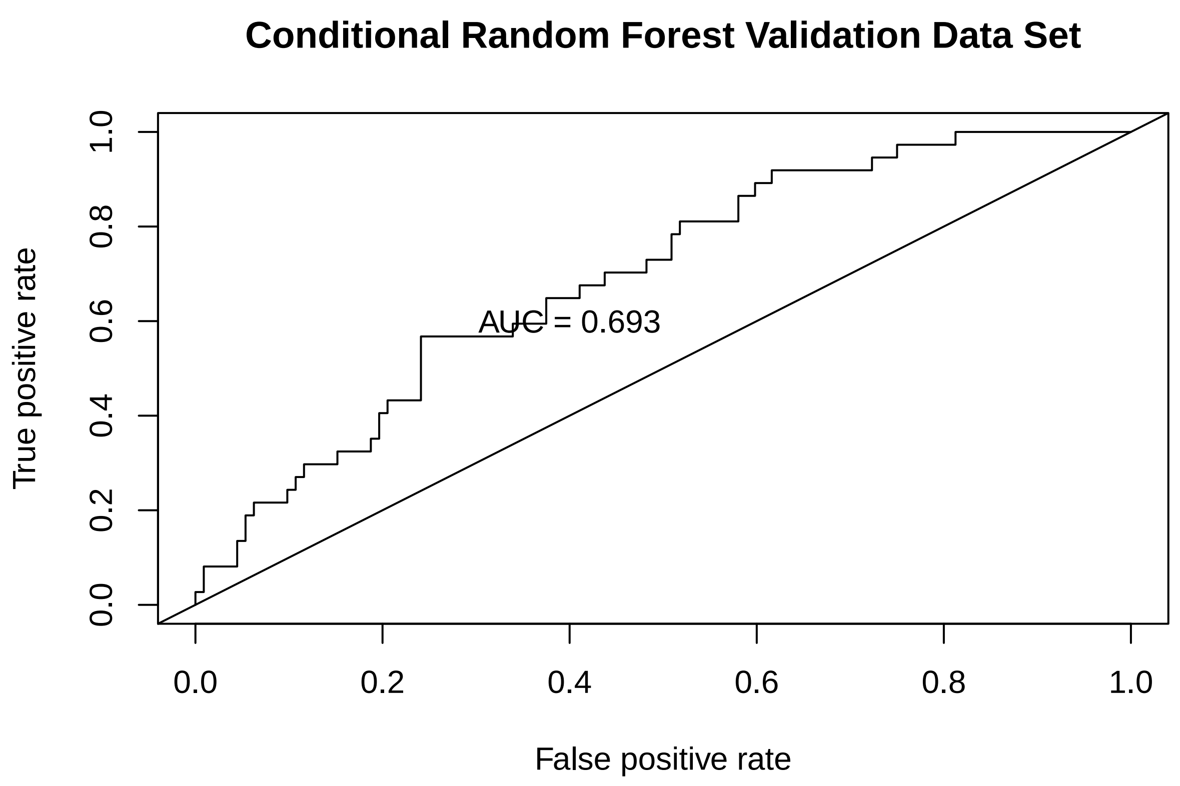



```r
#confusion matrix Training
pdata_logical_train <-  (unlist.Pred.train[,2] >= 0.5)
confusionMatrix(data = as.factor(as.numeric(pdata_logical_train)), reference = as.factor(as.numeric(tra
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 258  70
##          1   5  18
##
##                Accuracy : 0.7863
##                  95% CI : (0.7397, 0.8281)
##     No Information Rate : 0.7493
##     P-Value [Acc > NIR] : 0.06
##
##                   Kappa : 0.246
##  Mcnemar's Test P-Value : 1.467e-13
##
##             Sensitivity : 0.9810
##             Specificity : 0.2045
```

```
##          Pos Pred Value : 0.7866
##          Neg Pred Value : 0.7826
##              Prevalence : 0.7493
##          Detection Rate : 0.7350
##    Detection Prevalence : 0.9345
##       Balanced Accuracy : 0.5928
##
##        'Positive' Class : 0
##
```

```
crf.fit.pred.test <- predict(crf.fit, newdata=validationData, OOB = T, type="prob")
unlist.Pred.test <- matrix(unlist(crf.fit.pred.test), ncol=2,  byrow = TRUE)
pred.test.crf <- prediction(unlist.Pred.test[,2], validationData$FRACTURE)
roc.perf = performance(pred.test.crf, measure = "tpr", x.measure = "fpr")
auc.train <- performance(pred.test.crf, measure = "auc")
auc.train <- auc.train@y.values
plot(roc.perf, main="Conditional Random Forest Validation Data Set")
abline(a=0, b= 1)
text(x = .40, y = .6,paste("AUC = ", round(auc.train[[1]],3), sep = ""))
```




```
#confusion matrix
pdata_logical <-   unlist.Pred.test[,2] > 0.5
confusionMatrix(data = as.factor(as.numeric(pdata_logical)), reference = as.factor(as.numeric(validatio
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction   0   1
##          0 106  31
##          1   6   6
##
##                Accuracy : 0.7517
##                  95% CI : (0.6743, 0.8187)
##     No Information Rate : 0.7517
##     P-Value [Acc > NIR] : 0.544
##
##                   Kappa : 0.1403
##  Mcnemar's Test P-Value : 7.961e-05
##
##             Sensitivity : 0.9464
##             Specificity : 0.1622
##          Pos Pred Value : 0.7737
##          Neg Pred Value : 0.5000
##              Prevalence : 0.7517
##          Detection Rate : 0.7114
##    Detection Prevalence : 0.9195
##       Balanced Accuracy : 0.5543
##
##        'Positive' Class : 0
##
```

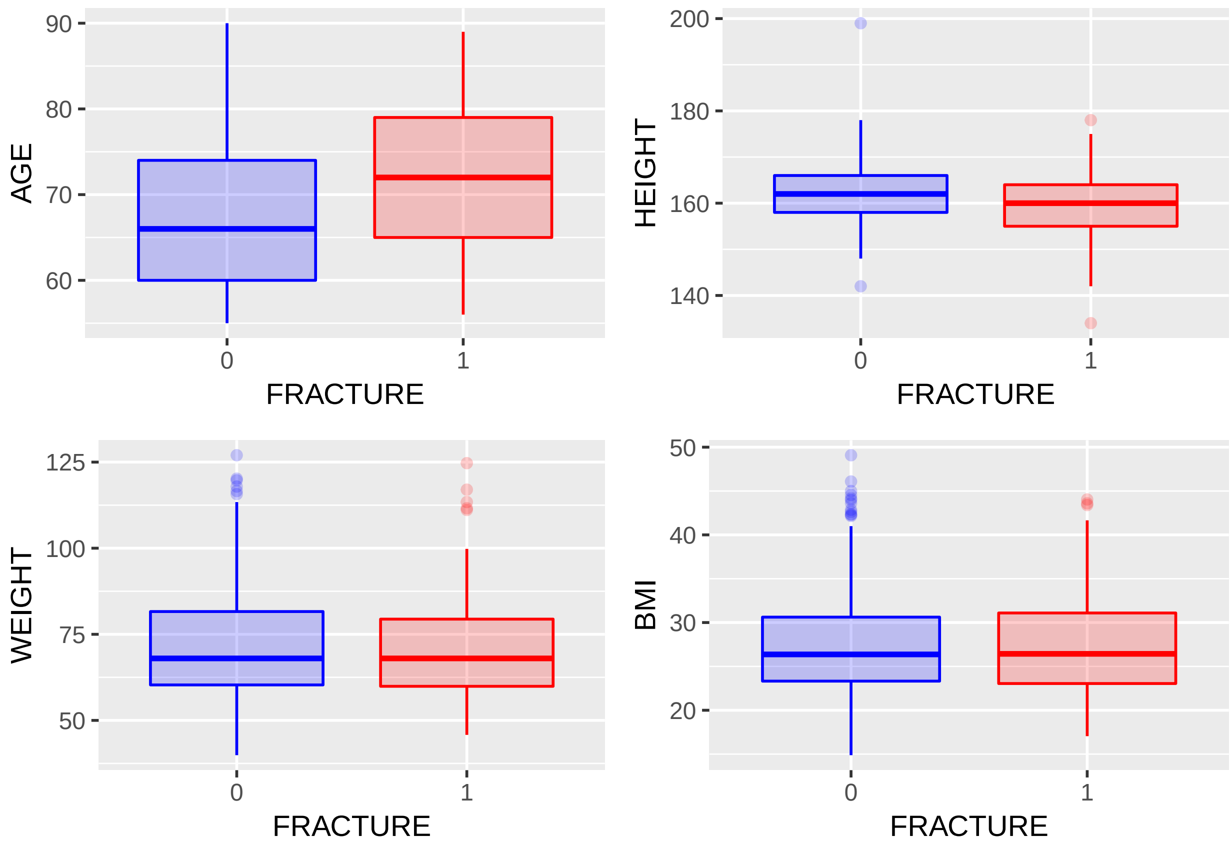
```
relativeImp <- varimp(crf.fit)
sort(relativeImp, decreasing = T)
```

```
##           AGE        HEIGHT     ARMASSIST           BMI     PRIORFRAC
##  8.372093e-03  7.581395e-03  6.124031e-03  1.674419e-03  4.496124e-04
##        WEIGHT      RATERISK         SMOKE       PREMENO       MOMFRAC
##  2.790698e-04 -7.751938e-05 -7.751938e-05 -3.255814e-04 -7.751938e-04
```

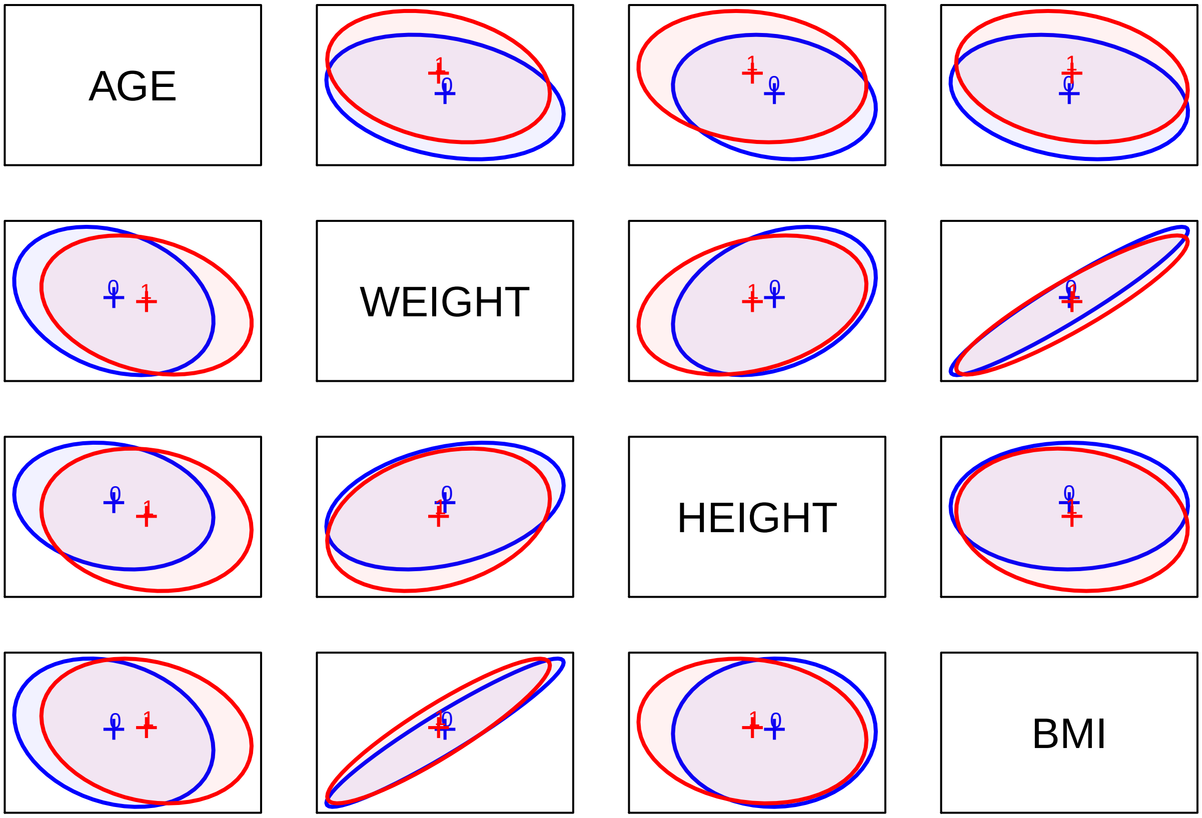
LDA AND QDA Model fit

```
library(MASS)
library(gridExtra)

## Assumption of Eq Variance / CoVariance
box.AGE <- ggplot(dataset, aes(x = FRACTURE, y = AGE, col = FRACTURE, fill = FRACTURE)) +
  geom_boxplot(alpha = 0.2) +
  theme(legend.position = "none") +
  scale_color_manual(values = c("blue", "red")) +
  scale_fill_manual(values = c("blue", "red"))

box.HEIGHT <- ggplot(dataset, aes(x = FRACTURE, y = HEIGHT, col = FRACTURE, fill = FRACTURE)) +
  geom_boxplot(alpha = 0.2) +
  theme(legend.position = "none") +
  scale_color_manual(values = c("blue", "red")) +
  scale_fill_manual(values = c("blue", "red"))
```

```
box.WEIGHT <- ggplot(dataset, aes(x = FRACTURE, y = WEIGHT, col = FRACTURE, fill = FRACTURE)) +
  geom_boxplot(alpha = 0.2) +
  theme(legend.position = "none") +
  scale_color_manual(values = c("blue", "red")) +
  scale_fill_manual(values = c("blue", "red"))

box.BMI <- ggplot(dataset, aes(x = FRACTURE, y = BMI, col = FRACTURE, fill = FRACTURE)) +
  geom_boxplot(alpha = 0.2) +
  theme(legend.position = "none") +
  scale_color_manual(values = c("blue", "red")) +
  scale_fill_manual(values = c("blue", "red"))

grid.arrange(box.AGE, box.HEIGHT, box.WEIGHT, box.BMI, nrow = 2, ncol = 2)
```




```
covEllipses(dataset[,c(5:8)], dataset$FRACTURE, fill = TRUE, pooled = FALSE,  col = c("blue", "red"), va
```




```r
#
# Conducting Levene Test
leveneTest(AGE ~ FRACTURE, dataset) # Came Back Not Significant
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##        Df F value Pr(>F)
## group   1   1.522 0.2179
##       498
```

```r
leveneTest(HEIGHT ~ FRACTURE, dataset)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##        Df F value Pr(>F)
## group   1  0.9566 0.3285
##       498
```

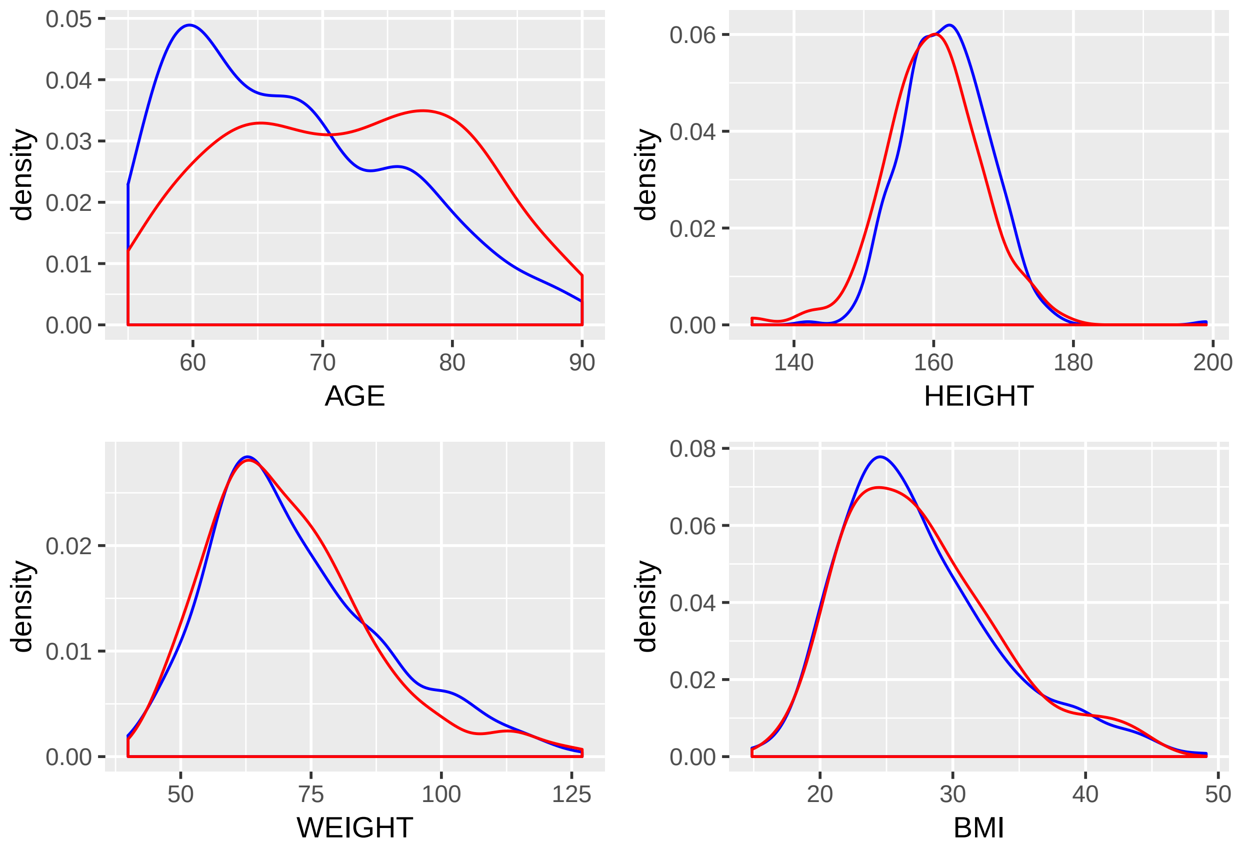
```r
leveneTest(WEIGHT ~ FRACTURE, dataset)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##        Df F value Pr(>F)
## group   1  0.9475 0.3308
##       498
```

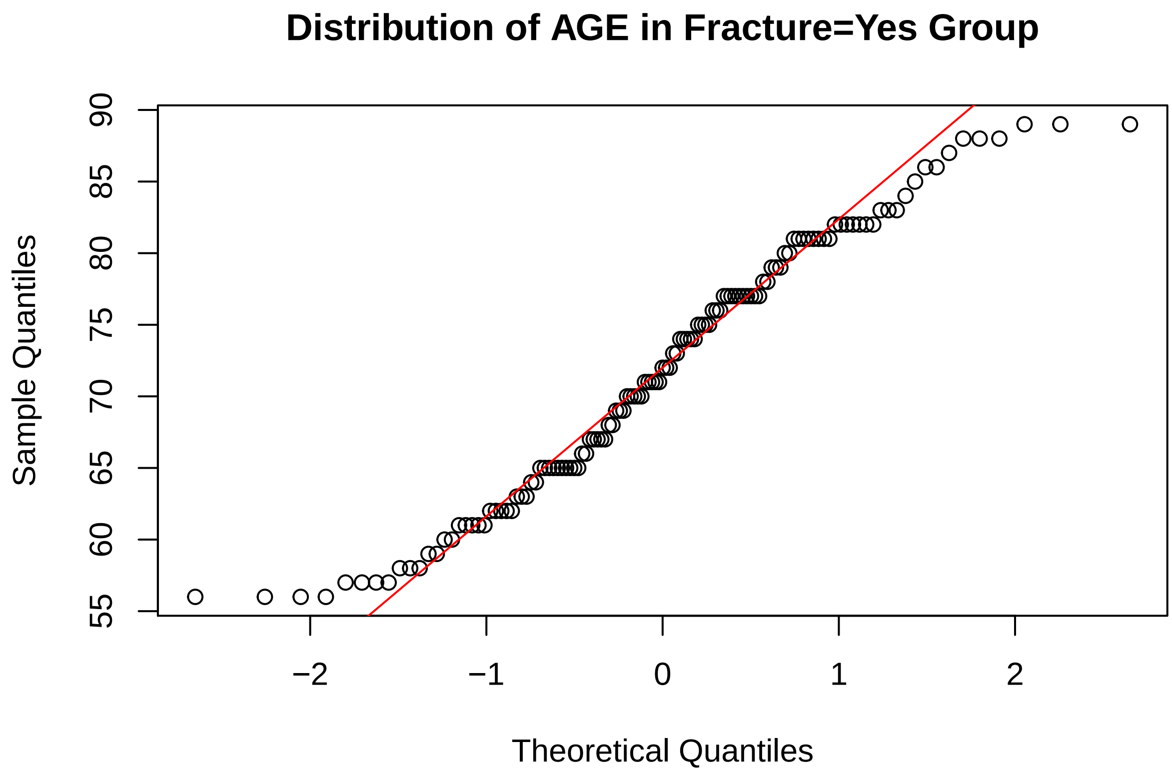
```r
leveneTest(BMI ~ FRACTURE, dataset)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##        Df F value Pr(>F)
## group   1  0.0188 0.8911
##       498
```

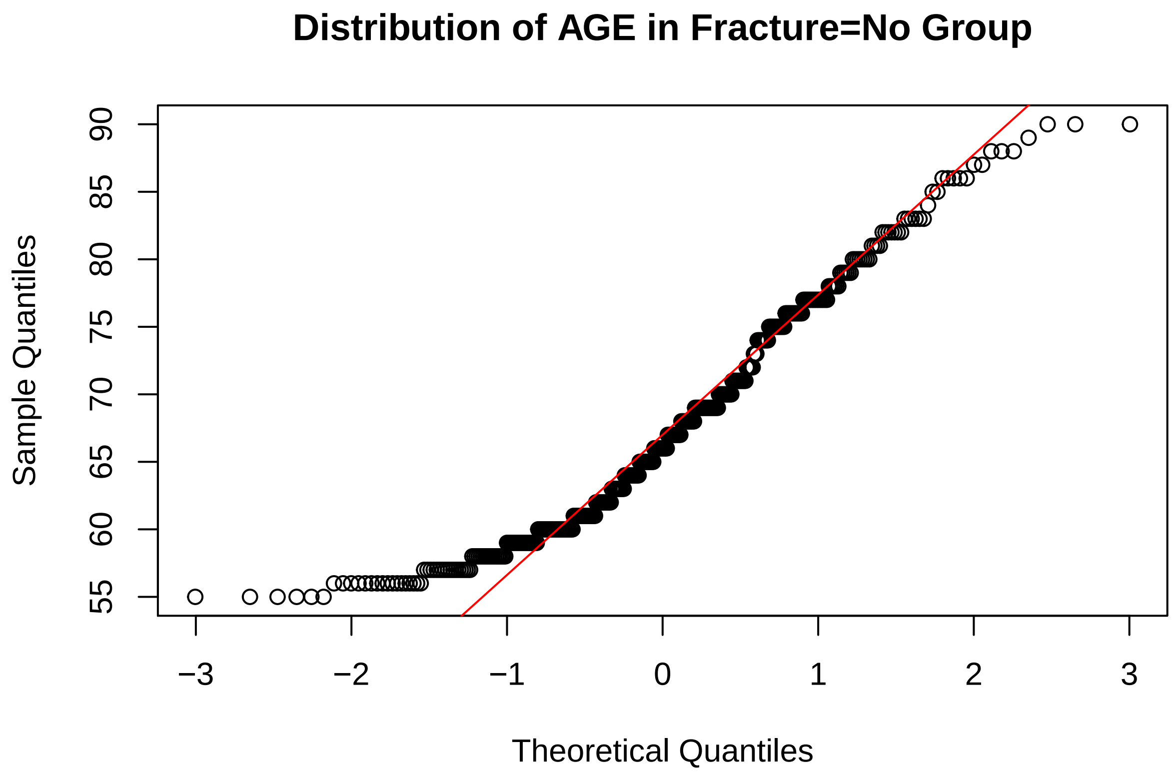
```r
# Came Back Not Significant, Confirms findings from previous plots

density.AGE <- ggplot(dataset, aes(x = AGE, y = ..density.., col = FRACTURE)) +
  geom_density(aes(y = ..density..)) +
  scale_color_manual(values = c("blue", "red")) +
  theme(legend.position = "none")

density.HEIGHT <- ggplot(dataset, aes(x = HEIGHT, y = ..density.., col = FRACTURE)) +
  geom_density(aes(y = ..density..)) +
  scale_color_manual(values = c("blue", "red")) +
  theme(legend.position = "none")

density.WEIGHT <- ggplot(dataset, aes(x = WEIGHT, y = ..density.., col = FRACTURE)) +
  geom_density(aes(y = ..density..)) +
  scale_color_manual(values = c("blue", "red")) +
  theme(legend.position = "none")

density.BMI <- ggplot(dataset, aes(x = BMI, y = ..density.., col = FRACTURE)) +
  geom_density(aes(y = ..density..)) +
  scale_color_manual(values = c("blue", "red")) +
  theme(legend.position = "none")

grid.arrange(density.AGE, density.HEIGHT, density.WEIGHT, density.BMI, nrow = 2, ncol = 2)
```



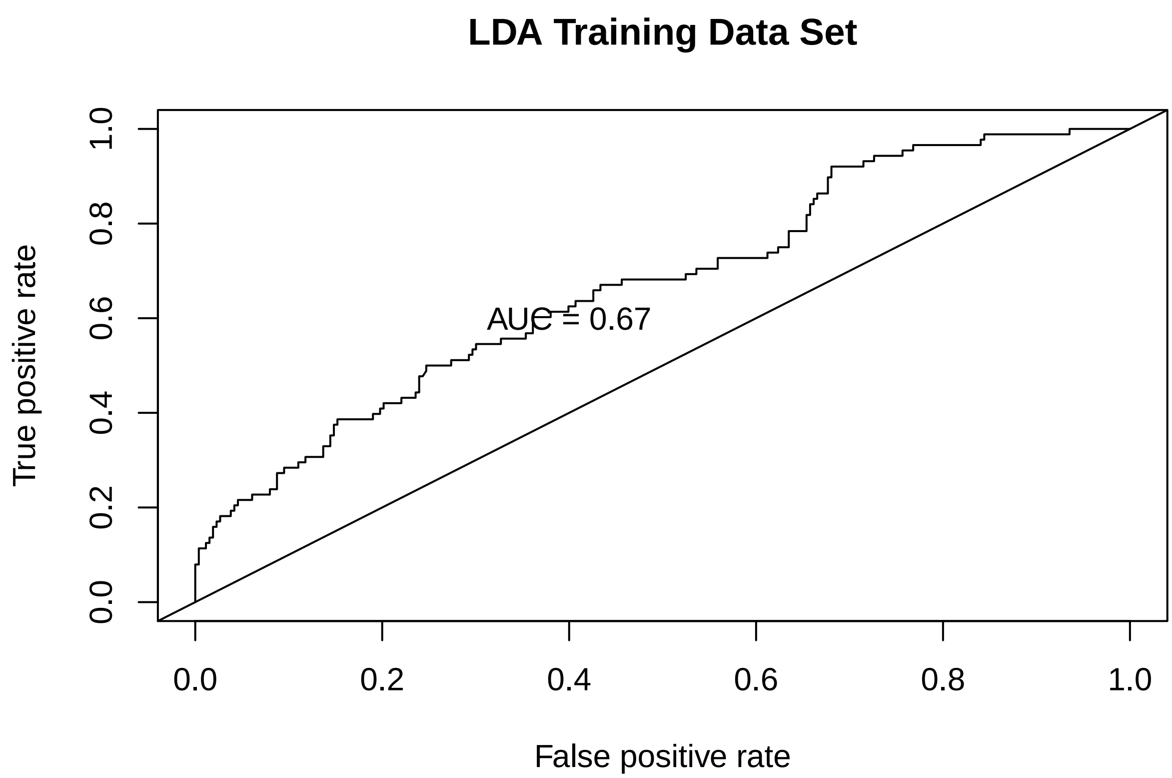

```r
# Check QQ Plot for AGE to ascertain Normality in BOTH Groups
frac.yes <- subset(dataset, FRACTURE == 1)
frac.no <- subset(dataset, FRACTURE == 0)
# Plot
qqnorm(frac.yes$AGE, main = "Distribution of AGE in Fracture=Yes Group"); qqline(frac.yes$AGE, col = 2)
```




```r
qqnorm(frac.no$AGE, main = "Distribution of AGE in Fracture=No Group"); qqline(frac.no$AGE, col = 2)
```




```
## Assumptions for Normality and of Equal Variance-Coavariance matrices Are Successfully Met.
## Run the LDA Now

set.seed(999)

lda.fit <- lda(FRACTURE ~ AGE + HEIGHT + WEIGHT + BMI, data = trainingData)

#ROC on training data set
ldaprd <- predict(lda.fit, newdata = trainingData)$posterior
ldaprd <- ldaprd[,2]
pred.train <- prediction(ldaprd, trainingData$FRACTURE)
roc.perf = performance(pred.train, measure = "tpr", x.measure = "fpr")
auc.train <- performance(pred.train, measure = "auc")
auc.train <- auc.train@y.values

#Plot ROC on Training Data
plot(roc.perf,main="LDA Training Data Set")
abline(a=0, b= 1) #Ref line indicating poor performance
text(x = .40, y = .6,paste("AUC = ", round(auc.train[[1]],3), sep = ""))
```

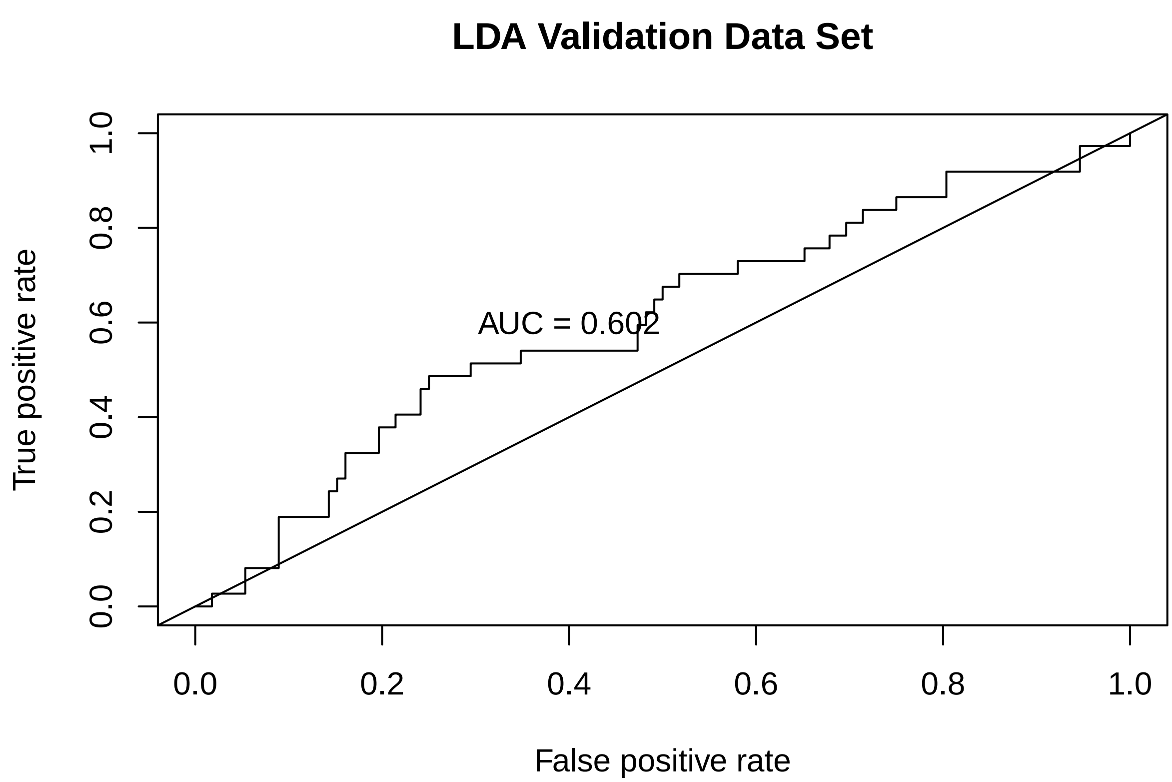



```r
prd <- predict(lda.fit, newdata = trainingData)$class
confusionMatrix(data = prd, reference = trainingData$FRACTURE)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 258  76
##          1   5  12
##
##                Accuracy : 0.7692
##                  95% CI : (0.7216, 0.8123)
##     No Information Rate : 0.7493
##     P-Value [Acc > NIR] : 0.2128
##
##                   Kappa : 0.1604
##  Mcnemar's Test P-Value : 7.381e-15
##
##             Sensitivity : 0.9810
##             Specificity : 0.1364
##          Pos Pred Value : 0.7725
##          Neg Pred Value : 0.7059
##              Prevalence : 0.7493
##          Detection Rate : 0.7350
##    Detection Prevalence : 0.9516
```

```
##        Balanced Accuracy : 0.5587
##
##         'Positive' Class : 0
##
```

```r
#ROC on test data set
ldaprd.test <- predict(lda.fit, newdata = validationData)$posterior
ldaprd.test <- ldaprd.test[,2]
pred.test <- prediction(ldaprd.test, validationData$FRACTURE)
roc.perf = performance(pred.test, measure = "tpr", x.measure = "fpr")
auc.train <- performance(pred.test, measure = "auc")
auc.train <- auc.train@y.values

#Plot ROC on Training Data
plot(roc.perf,main="LDA Validation Data Set")
abline(a=0, b= 1) #Ref line indicating poor performance
text(x = .40, y = .6,paste("AUC = ", round(auc.train[[1]],3), sep = ""))
```

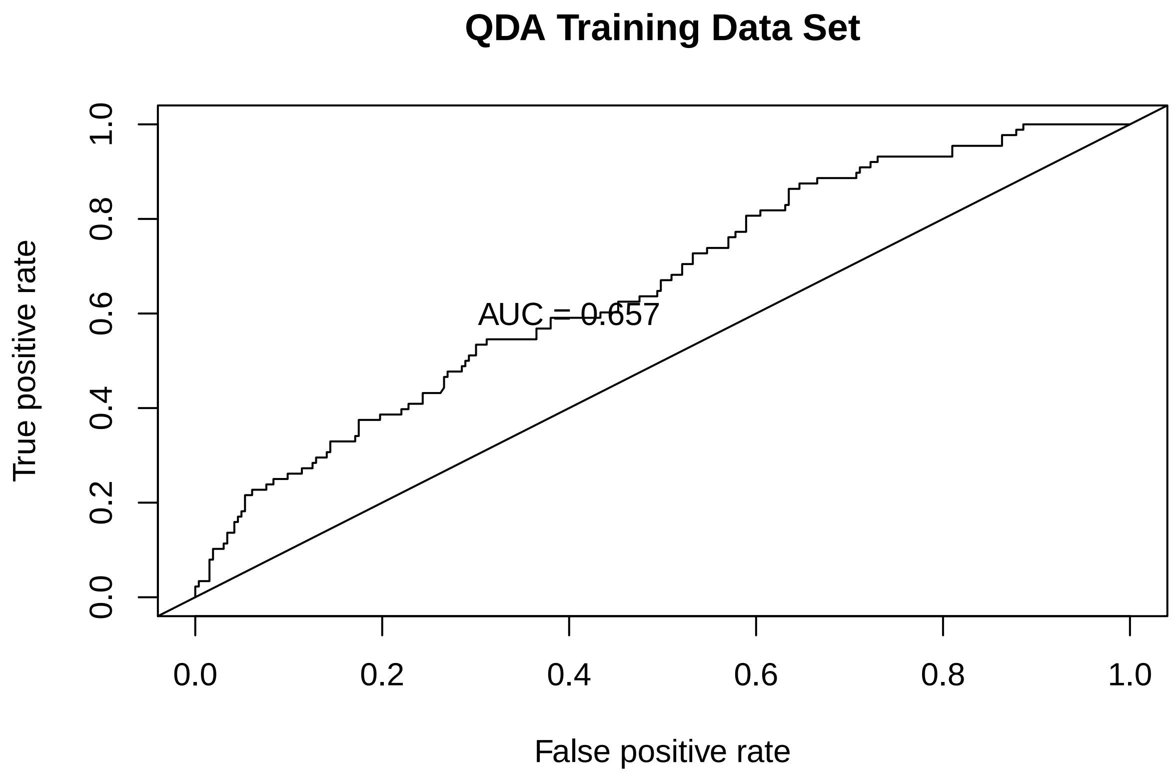



```r
prd.test <- predict(lda.fit, newdata = validationData)$class
confusionMatrix(data = prd.test, reference = validationData$FRACTURE)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   0   1
```

```
##          0 106  34
##          1   6   3
##
##                Accuracy : 0.7315
##                  95% CI : (0.6529, 0.8008)
##     No Information Rate : 0.7517
##     P-Value [Acc > NIR] : 0.7493
##
##                   Kappa : 0.0368
##  Mcnemar's Test P-Value : 1.963e-05
##
##             Sensitivity : 0.94643
##             Specificity : 0.08108
##          Pos Pred Value : 0.75714
##          Neg Pred Value : 0.33333
##              Prevalence : 0.75168
##          Detection Rate : 0.71141
##    Detection Prevalence : 0.93960
##       Balanced Accuracy : 0.51375
##
##        'Positive' Class : 0
##
```

```r
## Running QDA to see if it improves AUC

qda.fit <- qda(FRACTURE ~ AGE + HEIGHT + WEIGHT + BMI, data = trainingData)

#ROC on training data set
qdaprd <- predict(qda.fit, newdata = trainingData)$posterior
qdaprd <- qdaprd[,2]
pred.train <- prediction(qdaprd, trainingData$FRACTURE)
roc.perf = performance(pred.train, measure = "tpr", x.measure = "fpr")
auc.train <- performance(pred.train, measure = "auc")
auc.train <- auc.train@y.values

#Plot ROC on Training Data
plot(roc.perf,main="QDA Training Data Set")
abline(a=0, b= 1) #Ref line indicating poor performance
text(x = .40, y = .6,paste("AUC = ", round(auc.train[[1]],3), sep = ""))
```

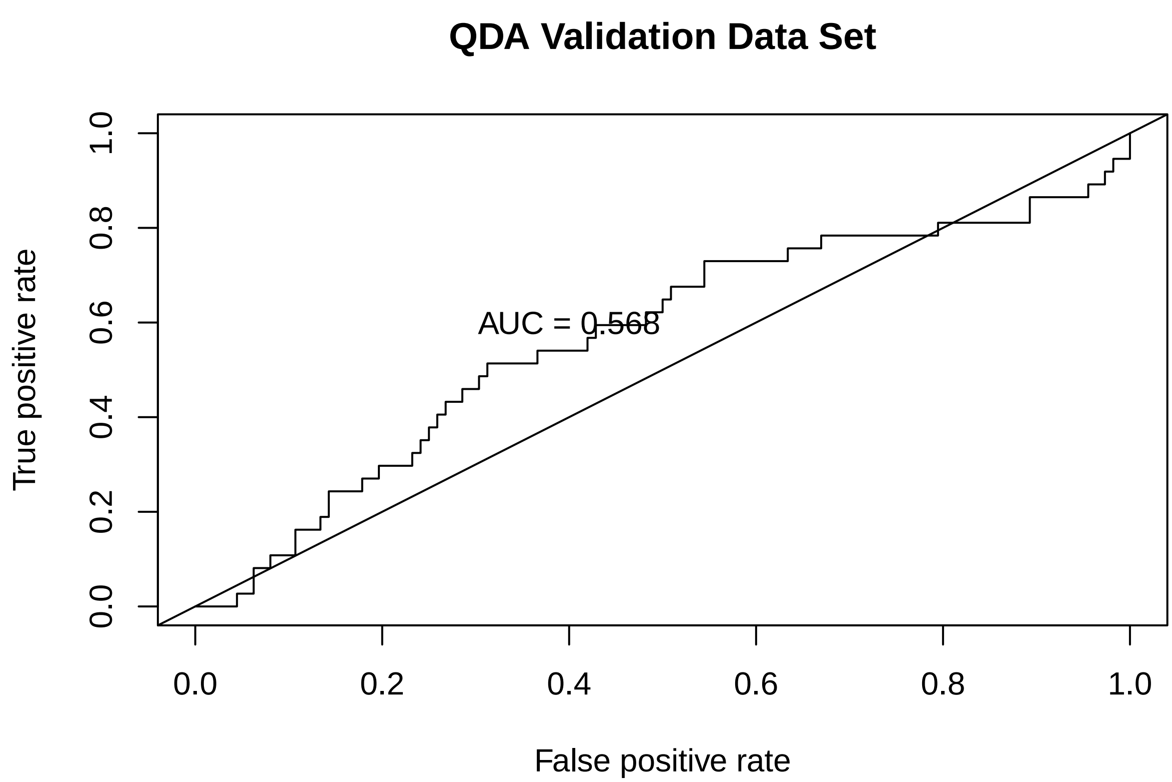



```r
prd <- predict(qda.fit, newdata = trainingData)$class
confusionMatrix(data = prd, reference = trainingData$FRACTURE)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 249  71
##          1  14  17
##
##                Accuracy : 0.7578
##                  95% CI : (0.7095, 0.8017)
##     No Information Rate : 0.7493
##     P-Value [Acc > NIR] : 0.3827
##
##                   Kappa : 0.1784
##  Mcnemar's Test P-Value : 1.247e-09
##
##             Sensitivity : 0.9468
##             Specificity : 0.1932
##          Pos Pred Value : 0.7781
##          Neg Pred Value : 0.5484
##              Prevalence : 0.7493
##          Detection Rate : 0.7094
##    Detection Prevalence : 0.9117
```

```
##        Balanced Accuracy : 0.5700
##
##         'Positive' Class : 0
##
```

```
#ROC on test data set
qdaprd.test <- predict(qda.fit, newdata = validationData)$posterior
qdaprd.test <- qdaprd.test[,2]
pred.test <- prediction(qdaprd.test, validationData$FRACTURE)
roc.perf = performance(pred.test, measure = "tpr", x.measure = "fpr")
auc.train <- performance(pred.test, measure = "auc")
auc.train <- auc.train@y.values

#Plot ROC on Training Data
plot(roc.perf,main="QDA Validation Data Set")
abline(a=0, b= 1) #Ref line indicating poor performance
text(x = .40, y = .6,paste("AUC = ", round(auc.train[[1]],3), sep = ""))
```




```
prd.test <- predict(qda.fit, newdata = validationData)$class
confusionMatrix(data = prd.test, reference = validationData$FRACTURE)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   0   1
```

```
##          0 103  33
##          1   9   4
##
##                Accuracy : 0.7181
##                  95% CI : (0.6387, 0.7887)
##     No Information Rate : 0.7517
##     P-Value [Acc > NIR] : 0.8512971
##
##                   Kappa : 0.0355
##  Mcnemar's Test P-Value : 0.0003867
##
##             Sensitivity : 0.9196
##             Specificity : 0.1081
##          Pos Pred Value : 0.7574
##          Neg Pred Value : 0.3077
##              Prevalence : 0.7517
##          Detection Rate : 0.6913
##    Detection Prevalence : 0.9128
##       Balanced Accuracy : 0.5139
##
##        'Positive' Class : 0
##
```