

# EDA\_Classfication.R

abhinavmishra

2022-11-03

```
##          A script for exploratory data analysis, and          -
##          classification training three classifiers             -
##    to diagnose heart disease based on the Heart Disease Data Set -

#####
##          Author: Abhinav Mishra                               ##
#####

##          Loading/Installing packages required                 -

#install.packages(c("MLeval", "caret", "ggvis","skimr", "tidyverse","ggvis", "e1071"))
library(tidyverse)

## -- Attaching packages ----- tidyverse 1.3.2 --
## v ggplot2 3.3.6      v purrr   0.3.5
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.3      v forcats 0.5.2
## -- Conflicts ----- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()

library(skimr)
library(ggvis)

##
## Attaching package: 'ggvis'
##
## The following object is masked from 'package:ggplot2':
##
##     resolution

library(caret)

## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift

library(ggvis)
library(caret)
library(MLeval)
```

```
##                               Loading data from the file                               -

processedWithHeader_cleveland <- read_csv("~/Documents/Freie/IFA/WiSe 22-23/Data Science/Week 2/processedWithHeader_cleveland.csv",
                                           show_col_types = FALSE, na = "?")

heart_data <- na.omit(data.frame(processedWithHeader_cleveland))
data <- data.frame(processedWithHeader_cleveland)
heart_data$ID <- seq.int(nrow(heart_data))
```

```
##                               Passing as factors                               -
```

```
#
heart_data$fbs <- as.factor(heart_data$fbs)
heart_data$restecg <- as.factor(heart_data$restecg)
heart_data$exang <- as.factor(heart_data$exang)
heart_data$slope <- as.factor(heart_data$slope)
#heart_data$cp <- as.factor(heart_data$cp)
```

```
##                               Data wrangling with preparation                               -
```

```
heart_data[heart_data$sex == 0, ]$sex <- "F"
heart_data[heart_data$sex == 1, ]$sex <- "M"
heart_data$sex <- as.factor(heart_data$sex)

heart_data[heart_data$goal == 0, ]$goal <- "healthy"
heart_data[heart_data$goal == 1, ]$goal <- "unhealthy"
heart_data[heart_data$goal == 2, ]$goal <- "unhealthy"
heart_data[heart_data$goal == 3, ]$goal <- "unhealthy"
heart_data[heart_data$goal == 4, ]$goal <- "unhealthy"

write.table(heart_data, file = 'heart_data')
table(heart_data$goal)
```

```
##
##   healthy unhealthy
##      160       137
```

```
##                               Descriptive Statistics + NA values omit                               -
```

```
str(heart_data)

## 'data.frame':   297 obs. of  15 variables:
## $ age      : num  63 67 67 37 41 56 62 57 63 53 ...
## $ sex      : Factor w/ 2 levels "F","M": 2 2 2 2 1 2 1 1 2 2 ...
## $ cp       : num  1 4 4 3 2 2 4 4 4 4 ...
## $ trestbps : num  145 160 120 130 130 120 140 120 130 140 ...
## $ chol     : num  233 286 229 250 204 236 268 354 254 203 ...
## $ fbs      : Factor w/ 2 levels "0","1": 2 1 1 1 1 1 1 1 1 2 ...
## $ restecg  : Factor w/ 3 levels "0","1","2": 3 3 3 1 3 1 3 1 3 3 ...
## $ thalach  : num  150 108 129 187 172 178 160 163 147 155 ...
## $ exang    : Factor w/ 2 levels "0","1": 1 2 2 1 1 1 1 2 1 2 ...
## $ oldpeak  : num  2.3 1.5 2.6 3.5 1.4 0.8 3.6 0.6 1.4 3.1 ...
## $ slope    : Factor w/ 3 levels "1","2","3": 3 2 2 3 1 1 3 1 2 3 ...
## $ ca       : num  0 3 2 0 0 0 2 0 1 0 ...
```

```
## $ thal      : num  6 3 7 3 3 3 3 7 7 ...
## $ goal      : chr  "healthy" "unhealthy" "unhealthy" "healthy" ...
## $ ID        : int   1 2 3 4 5 6 7 8 9 10 ...
## - attr(*, "na.action")= 'omit' Named int [1:6] 88 167 193 267 288 303
##   ..- attr(*, "names")= chr [1:6] "88" "167" "193" "267" ...
```

```
summary(heart_data)
```

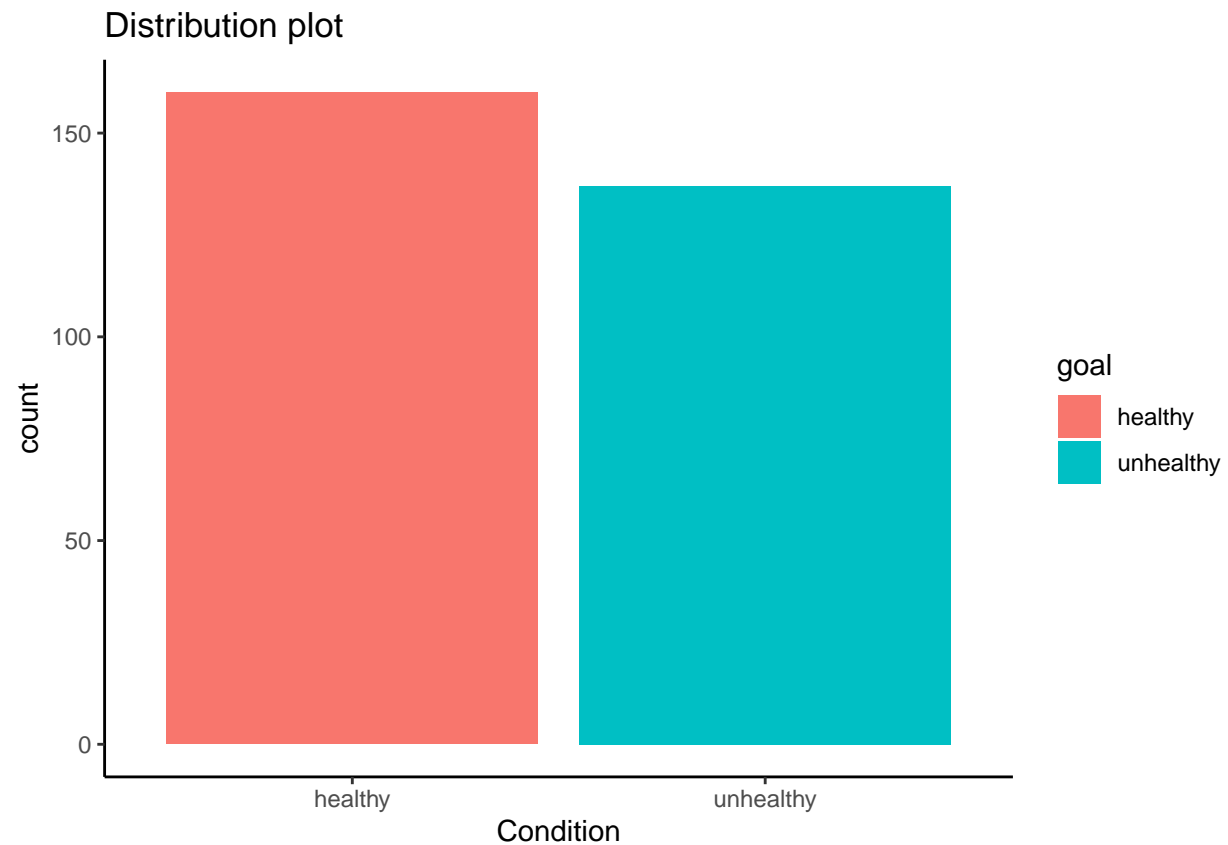
```
##      age      sex      cp      trestbps      chol
## Min.   :29.00 F: 96   Min.   :1.000   Min.   : 94.0   Min.   :126.0
## 1st Qu.:48.00 M:201   1st Qu.:3.000   1st Qu.:120.0   1st Qu.:211.0
## Median :56.00      Median :3.000   Median :130.0   Median :243.0
## Mean   :54.54      Mean   :3.158   Mean   :131.7   Mean   :247.4
## 3rd Qu.:61.00      3rd Qu.:4.000   3rd Qu.:140.0   3rd Qu.:276.0
## Max.   :77.00      Max.   :4.000   Max.   :200.0   Max.   :564.0
## fbs      restecg      thalach      exang      oldpeak      slope
## 0:254    0:147   Min.   : 71.0   0:200   Min.   :0.000   1:139
## 1: 43    1:  4   1st Qu.:133.0   1: 97   1st Qu.:0.000   2:137
##          2:146   Median :153.0      Median :0.800   3: 21
##          Mean   :149.6      Mean   :1.056
##          3rd Qu.:166.0      3rd Qu.:1.600
##          Max.   :202.0      Max.   :6.200
##      ca      thal      goal      ID
## Min.   :0.0000   Min.   :3.000   Length:297      Min.   : 1
## 1st Qu.:0.0000   1st Qu.:3.000   Class :character 1st Qu.: 75
## Median :0.0000   Median :3.000   Mode  :character Median :149
## Mean   :0.6768   Mean   :4.731      Mean   :149
## 3rd Qu.:1.0000   3rd Qu.:7.000      3rd Qu.:223
## Max.   :3.0000   Max.   :7.000      Max.   :297
```

```
sum(is.na(heart_data))
```

```
## [1] 0
```

```
##      Descriptive plots      -
```

```
ggplot(heart_data, aes(x = goal, fill = goal)) +
  geom_bar() + theme_classic() +
  labs(title='Distribution plot') +
  xlab("Condition")
```



```
table(heart_data$goal)
```

```
##
##   healthy unhealthy
##      160      137
```

```
round(prop.table(table(heart_data$goal)) * 100, digits = 1)
```

```
##
##   healthy unhealthy
##      53.9      46.1
```

```
##                               Scatter plots by condition                               -
```

```
#heart_data %>% ggvis(~age, ~trestbps, fill = ~goal) %>% layer_points()
```

```
#heart_data %>% ggvis(~age, ~trestbps, fill = ~sex) %>% layer_points()
```

```
#heart_data %>% ggvis(~age, ~trestbps, fill = ~cp) %>% layer_points()
```

```
##                               Data partition                               -
```

```
test_index <- createDataPartition(y = heart_data$goal, times = 1,
                                   p = 0.2, list= FALSE)
```

```
heart_data$goal <- as.factor(heart_data$goal)
```

```
train_data <- heart_data[-test_index, ]
```

```
test_data <- heart_data[test_index, ]
```

```

##                               Classifiers                               -

##      1. Logistic regression: Fit the logistic regression model,      -
##      that is a GLM using a binomial link using the caret function train() -

set.seed(112)
log_fit <- train(goal ~.-ID ,
                 data = train_data,
                 method = "glm",
                 family = "binomial")

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

log_pred <- predict(log_fit, test_data)
confusionMatrix(log_pred, test_data$goal)

## Confusion Matrix and Statistics
##
##              Reference
## Prediction  healthy unhealthy
## healthy      27          5
## unhealthy     5         23
##
##              Accuracy : 0.8333
##              95% CI : (0.7148, 0.9171)
##      No Information Rate : 0.5333
##      P-Value [Acc > NIR] : 1.056e-06
##
##              Kappa : 0.6652
##
##  Mcnemar's Test P-Value : 1
##
##      Sensitivity : 0.8438
##      Specificity : 0.8214
##      Pos Pred Value : 0.8438
##      Neg Pred Value : 0.8214
##      Prevalence : 0.5333
##      Detection Rate : 0.4500
##      Detection Prevalence : 0.5333
##      Balanced Accuracy : 0.8326
##
##      'Positive' Class : healthy
##

##                               2. Random forest                               -

set.seed(112)
rf_fit <- train(goal ~.-ID ,
               data = train_data,
               method = "rf")

rf_pred <- predict(rf_fit, test_data)
confusionMatrix(rf_pred, test_data$goal)

## Confusion Matrix and Statistics

```

```
##
##           Reference
## Prediction  healthy unhealthy
##   healthy      27         5
##   unhealthy     5        23
##
##           Accuracy : 0.8333
##           95% CI : (0.7148, 0.9171)
##   No Information Rate : 0.5333
##   P-Value [Acc > NIR] : 1.056e-06
##
##           Kappa : 0.6652
##
## Mcnemar's Test P-Value : 1
##
##           Sensitivity : 0.8438
##           Specificity : 0.8214
##   Pos Pred Value : 0.8438
##   Neg Pred Value : 0.8214
##   Prevalence : 0.5333
##   Detection Rate : 0.4500
##   Detection Prevalence : 0.5333
##   Balanced Accuracy : 0.8326
##
##   'Positive' Class : healthy
##
```

```
## 3. Boosted logistic regression: using decision stumps (one node decision trees) -
##           as weak learners. It implements a internal version of decision -
##           stump classifier instead of using -
##           calls to rpart. Also, training and testing phases of the -
##           classification process are split into separate functions. -
```

```
set.seed(112)
blog_fit <- train(goal ~.-ID,
                  data = train_data,
                  method = "LogitBoost")
blog_pred <- predict(blog_fit, test_data)
confusionMatrix(blog_pred, test_data$goal)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  healthy unhealthy
##   healthy      24         6
##   unhealthy     8        22
##
##           Accuracy : 0.7667
##           95% CI : (0.6396, 0.8662)
##   No Information Rate : 0.5333
##   P-Value [Acc > NIR] : 0.0001655
##
##           Kappa : 0.5333
##
## Mcnemar's Test P-Value : 0.7892680
```

```
##
##      Sensitivity : 0.7500
##      Specificity : 0.7857
##      Pos Pred Value : 0.8000
##      Neg Pred Value : 0.7333
##      Prevalence : 0.5333
##      Detection Rate : 0.4000
##      Detection Prevalence : 0.5000
##      Balanced Accuracy : 0.7679
##
##      'Positive' Class : healthy
##
```

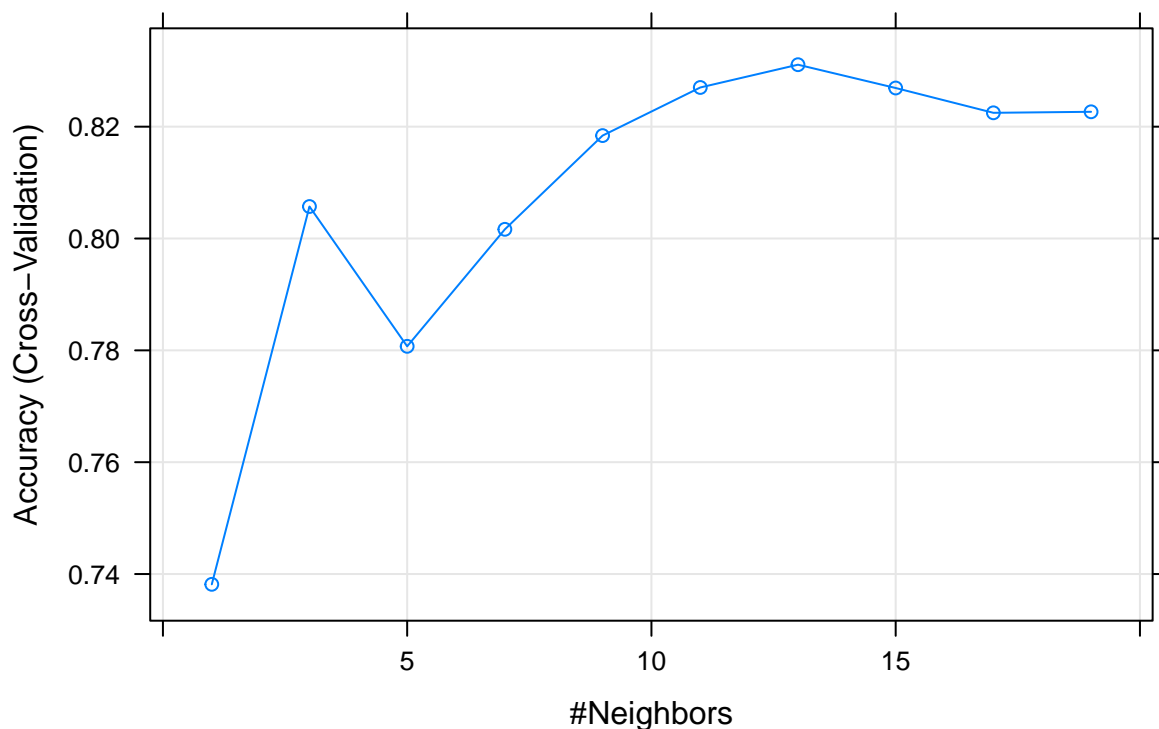
#### ## 4. KNN -

```
ctrl <- trainControl(method = "cv", verboseIter = FALSE, number = 5)

knn_fit <- train(goal ~. -ID , data = train_data,
  method = "knn", preProcess = c("center","scale"),
  trControl = ctrl , tuneGrid = expand.grid(k = seq(1, 20, 2)))

plot(knn_fit, main = "K-nearest neighbour")
```

### K-nearest neighbour



```
knn_pred <- predict(knn_fit, test_data)
confusionMatrix(knn_pred, test_data$goal)
```

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

```
##           Reference
## Prediction  healthy unhealthy
##   healthy      28         4
##   unhealthy     4        24
##
##           Accuracy : 0.8667
##           95% CI : (0.7541, 0.9406)
##   No Information Rate : 0.5333
##   P-Value [Acc > NIR] : 4.403e-08
##
##           Kappa : 0.7321
##
## Mcnemar's Test P-Value : 1
##
##           Sensitivity : 0.8750
##           Specificity : 0.8571
##   Pos Pred Value : 0.8750
##   Neg Pred Value : 0.8571
##   Prevalence : 0.5333
##   Detection Rate : 0.4667
##   Detection Prevalence : 0.5333
##   Balanced Accuracy : 0.8661
##
##   'Positive' Class : healthy
##
```

# ``` ## Performance Comparison - ```

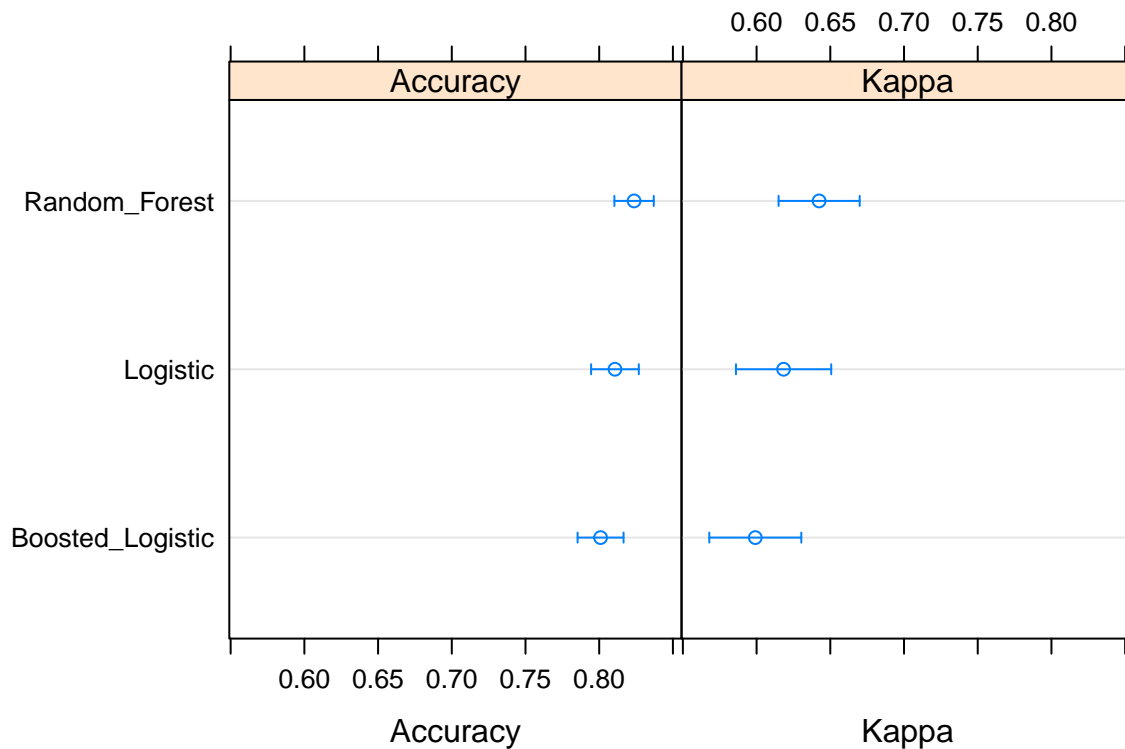
```
results <- resamples(list(Logistic = log_fit,
                          Random_Forest = rf_fit,
                          Boosted_Logistic = blog_fit))
summary(results)
```

```
##
## Call:
## summary.resamples(object = results)
##
## Models: Logistic, Random_Forest, Boosted_Logistic
## Number of resamples: 25
##
## Accuracy
##           Min.   1st Qu.   Median     Mean   3rd Qu.     Max.
## Logistic      0.7261905 0.7956989 0.8111111 0.8106845 0.8333333 0.8888889
## Random_Forest  0.7500000 0.8111111 0.8247423 0.8236557 0.8452381 0.8888889
## Boosted_Logistic 0.7142857 0.7777778 0.8048780 0.8009295 0.8222222 0.8780488
##
## NA's
## Logistic      0
## Random_Forest  0
## Boosted_Logistic 0
##
## Kappa
##           Min.   1st Qu.   Median     Mean   3rd Qu.     Max.
## Logistic      0.4511364 0.5886618 0.6240786 0.6183196 0.6673238 0.7776680
## Random_Forest  0.4848131 0.6192609 0.6428417 0.6424229 0.6906808 0.7783251
## Boosted_Logistic 0.4153132 0.5484195 0.6109134 0.5991288 0.6400791 0.7489284
```



```
##
## Logistic      NA's 0
## Random_Forest 0
## Boosted_Logistic 0
```

```
dotplot(results)
```



**Confidence Level: 0.95**

```
cf_rf <- confusionMatrix(rf_pred, test_data$goal)
cf_log <- confusionMatrix(log_pred, test_data$goal)
cf_blog <- confusionMatrix(log_pred, test_data$goal)
cf_knn <- confusionMatrix(knn_pred, test_data$goal)

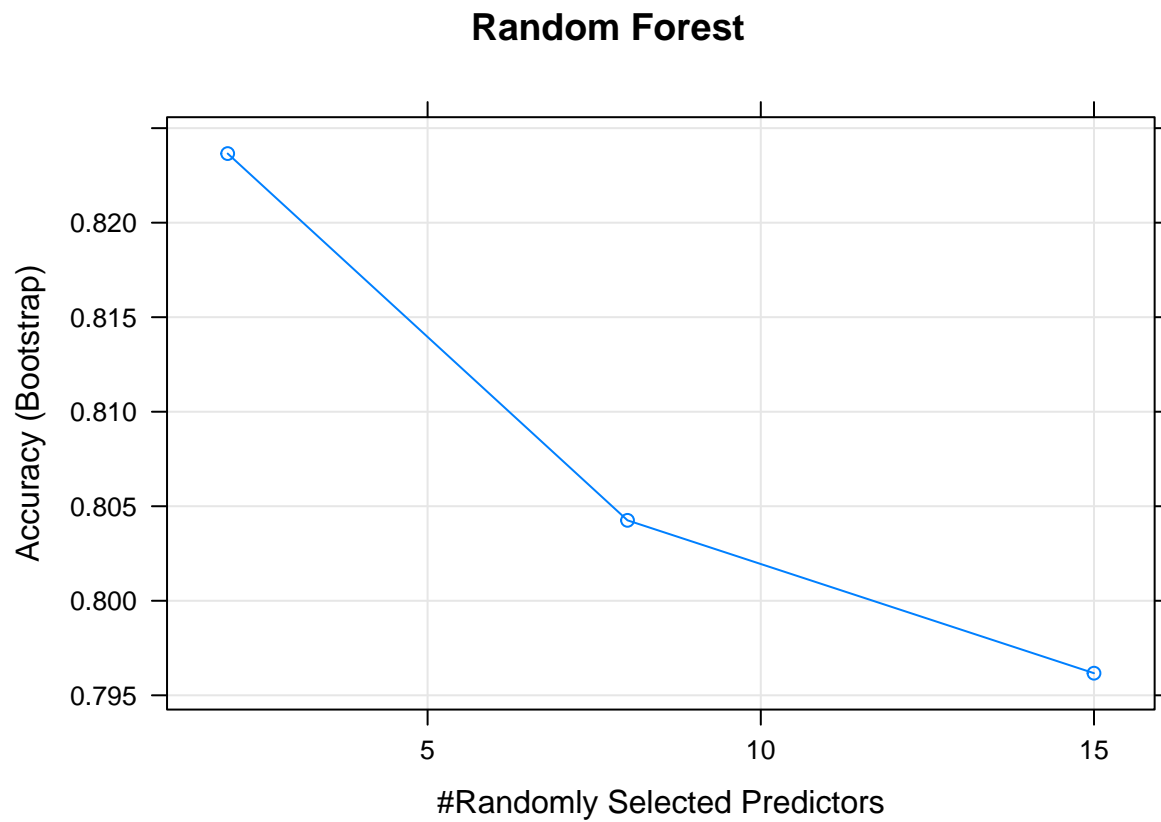
Accuracy <- 100*rbind(cf_log[["overall"]][["Accuracy"]],
  cf_rf[["overall"]][["Accuracy"]],
  cf_blog[["overall"]][["Accuracy"]],
  cf_knn[["overall"]][["Accuracy"]])

Specificity <- 100*rbind(cf_log[["byClass"]][["Specificity"]],
  cf_rf[["byClass"]][["Specificity"]],
  cf_blog[["byClass"]][["Specificity"]],
  cf_knn[["byClass"]][["Specificity"]])

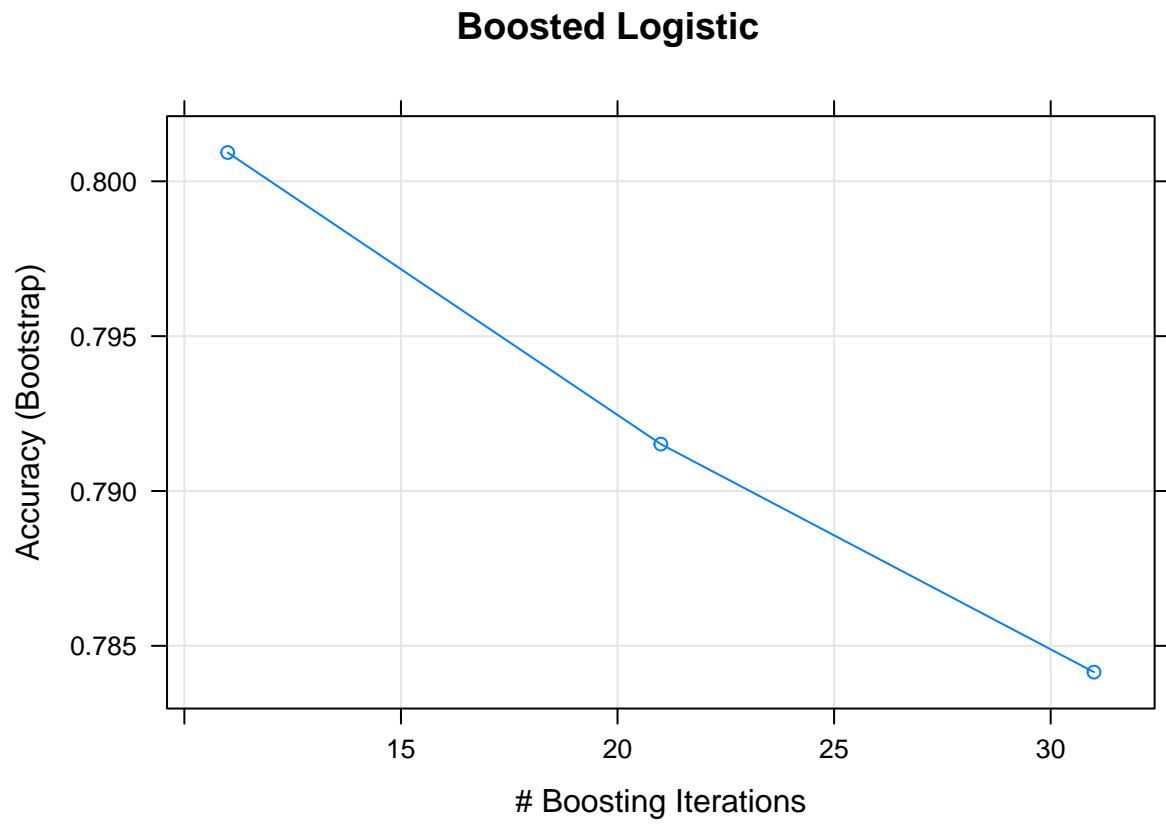
Sensitivity <- 100*rbind(cf_log[["byClass"]][["Sensitivity"]],
  cf_rf[["byClass"]][["Sensitivity"]],
  cf_blog[["byClass"]][["Sensitivity"]],
  cf_knn[["byClass"]][["Sensitivity"]])
```

```
pf_result <- t(data.frame(Accuracy, Specificity, Sensitivity))
colnames(pf_result) <- c("Logistic", "Random Forest", "Boosted Logit", "KNN")
pf_result <- as.matrix(pf_result)

plot(rf_fit, main = "Random Forest")
```

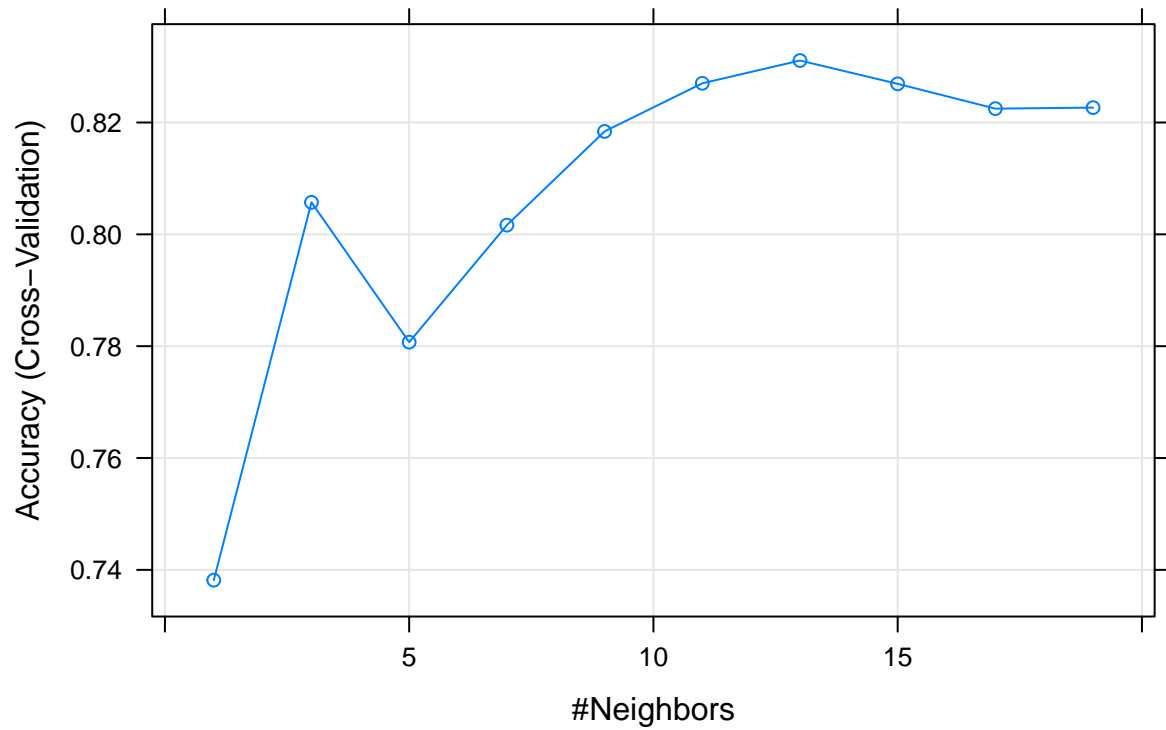


```
plot(blog_fit, main = "Boosted Logistic")
```

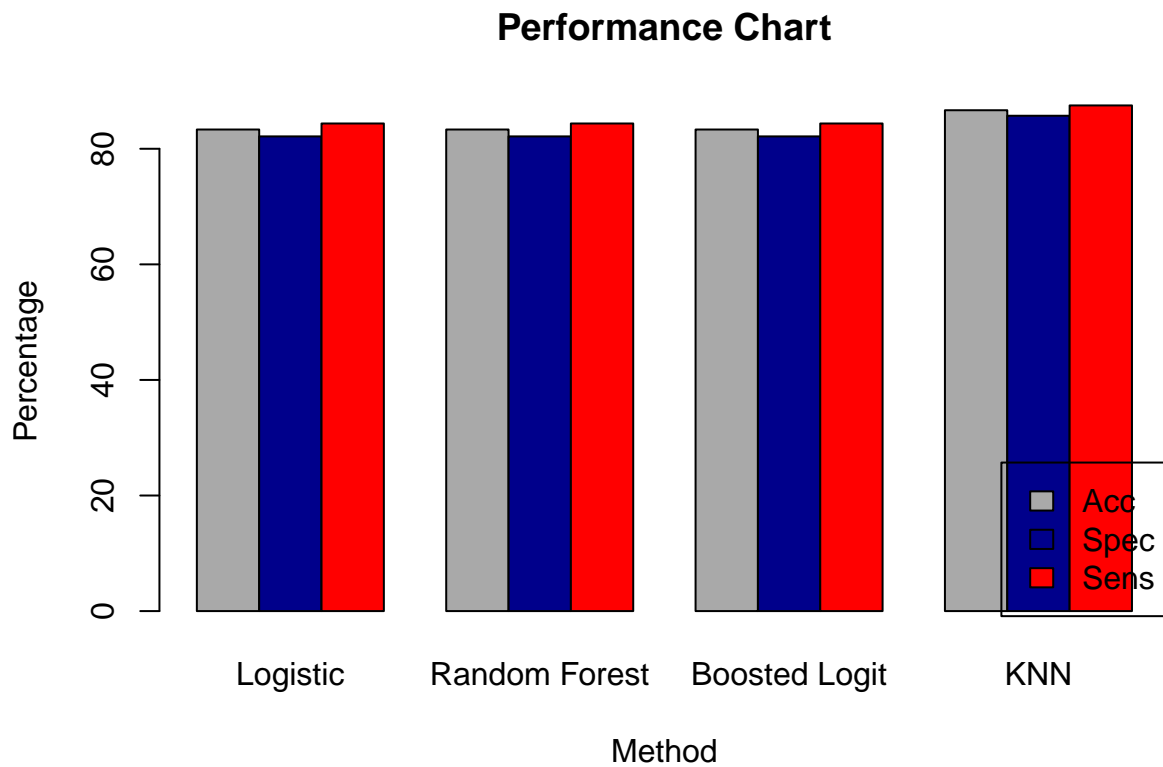


```
plot(knn_fit, main = "K-nearest neighbour")
```

## K-nearest neighbour



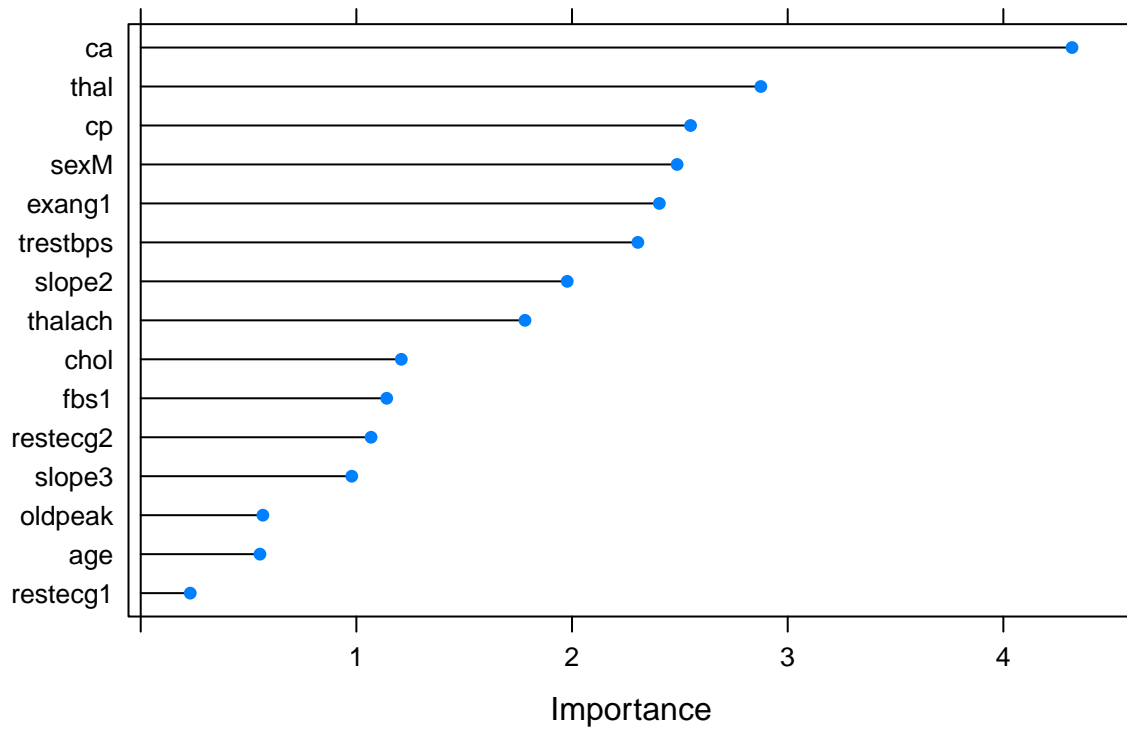
```
barplot(height = pf_result, beside = TRUE,  
        col = c("darkgrey", "darkblue", "red"),  
        xlab = "Method", legend.text = c("Acc", "Spec", "Sens"),  
        args.legend = list(x = "bottomright"),  
        main = "Performance Chart", ylab = "Percentage")
```



```
## Feature extraction -
feat_log <- varImp(log_fit, scale = FALSE)
feat_rf <- varImp(rf_fit, scale = FALSE)
feat_blog <- varImp(blog_fit, scale = FALSE)
feat_knn <- varImp(knn_fit, scale = FALSE)

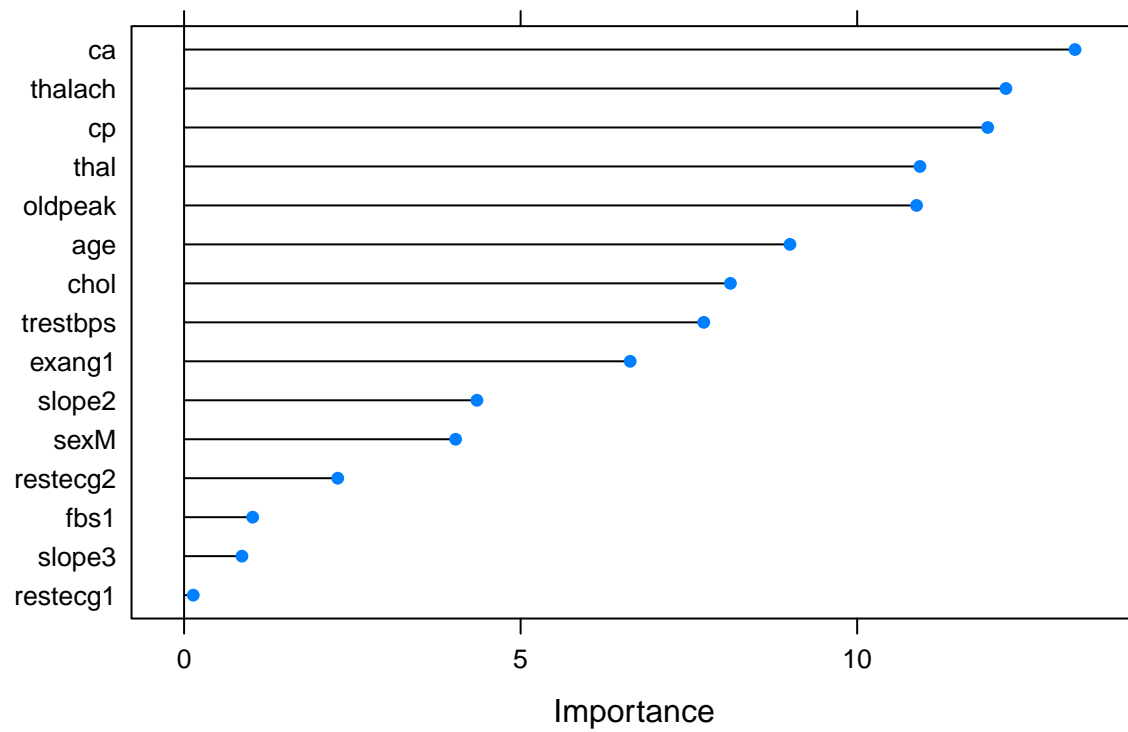
plot(feat_log, main = "Logistic regression: features")
```

## Logistic regression: features



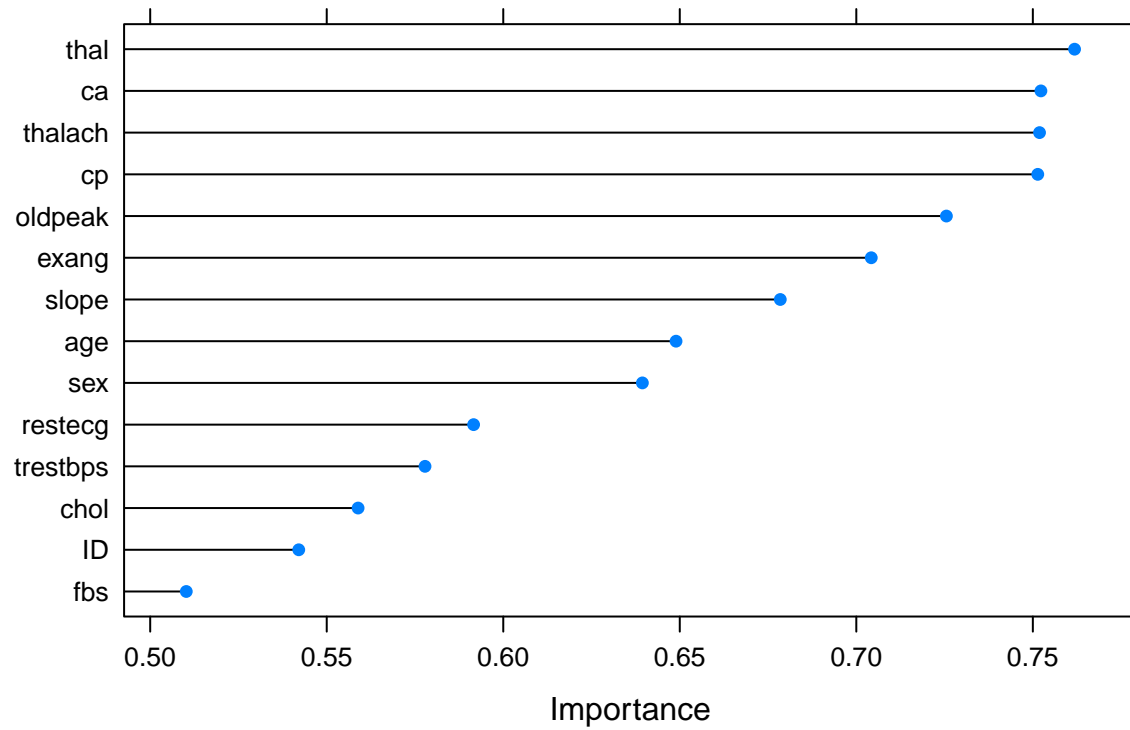
```
plot(feet_rf, main = "Random forest: features")
```

## Random forest: features



```
plot(feat_blog, main = "Boosted Logistic regression: features")
```

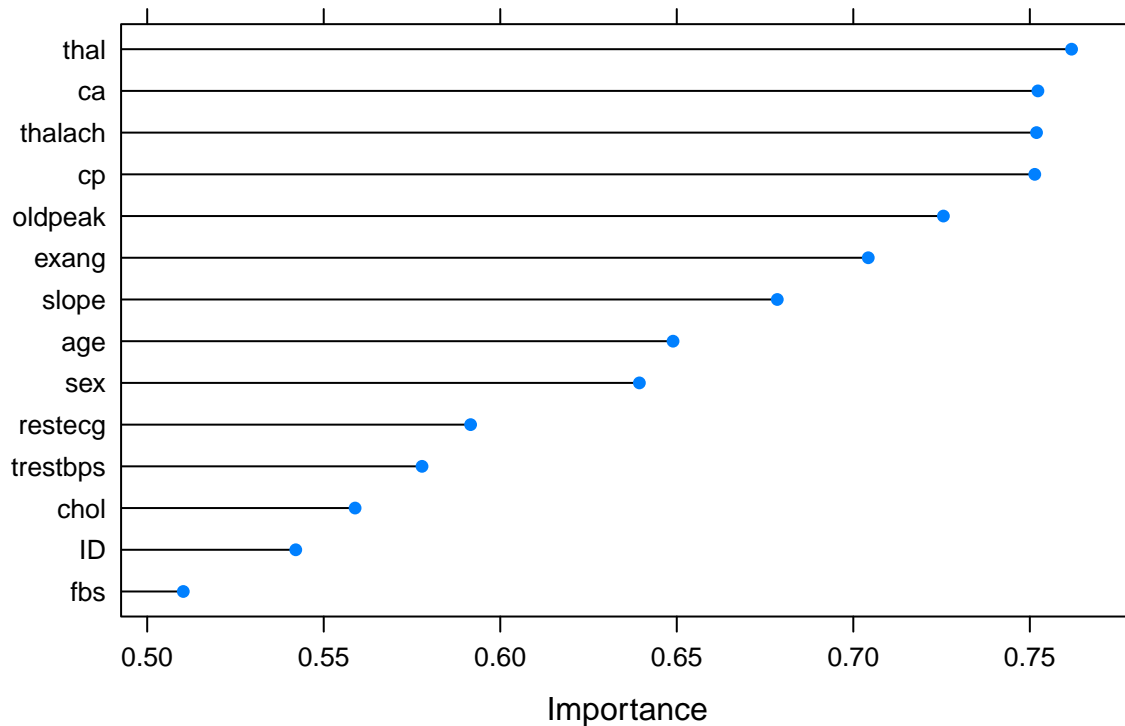
## Boosted Logistic regression: features



```
plot(feet_knn, main = "KNN: features")
```



## KNN: features



```
##      Model Evaluation with ROC, calibration, precision      -
##      recall gain, and Obs vs. Pred probabilities curve     -
```

```
cont <- trainControl(method="cv",
                     summaryFunction=twoClassSummary,
                     classProbs=T,
                     savePredictions = T)

log <- train(goal ~.-ID ,
            data = train_data,
            method = "glm", preProc=c("center", "scale"),
            family = "binomial", trControl=cont)
```

```
## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" was not
## in the result set. ROC will be used instead.
```

```
rf <- train(goal ~.-ID ,
            data = train_data, preProc=c("center", "scale"),
            method = "rf", trControl=cont)
```

```
## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" was not
## in the result set. ROC will be used instead.
```

```
blog <- train(goal ~.-ID,
              data = train_data, preProc=c("center", "scale"),
              method = "LogitBoost", trControl=cont)
```

```
## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" was not
```

```

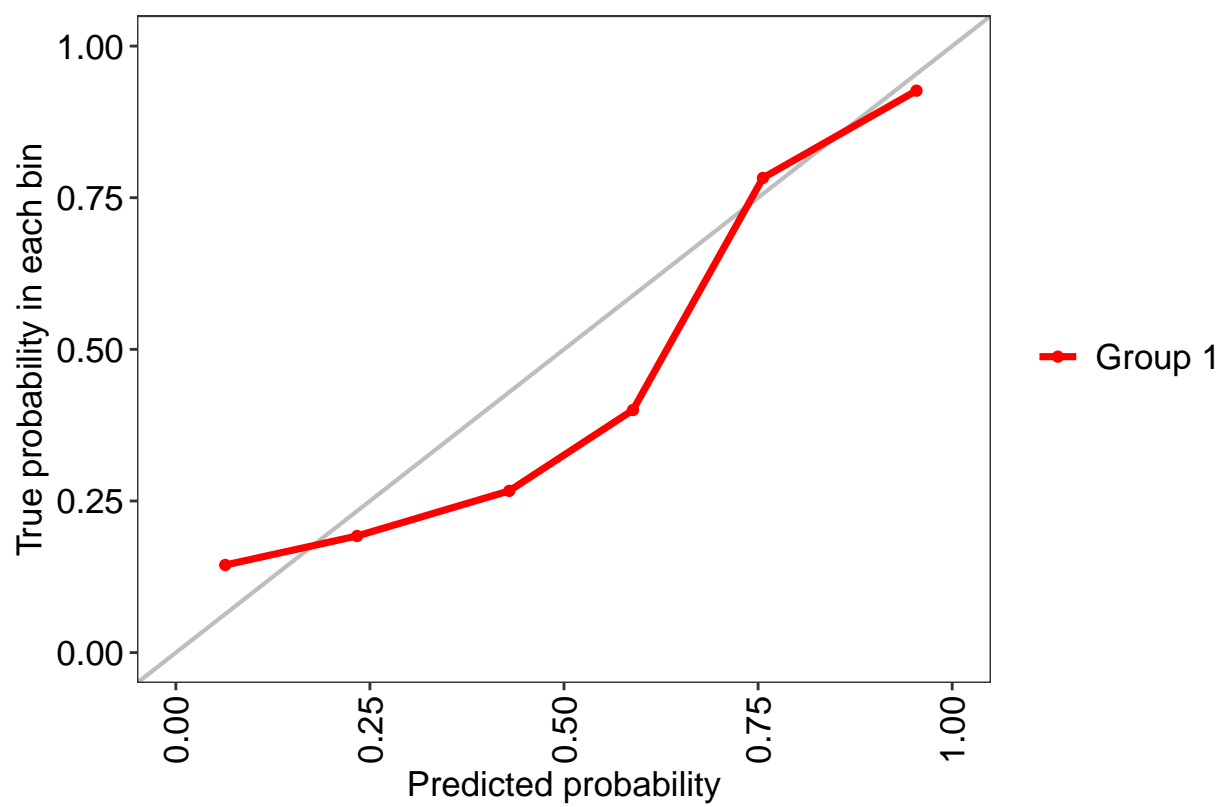
## in the result set. ROC will be used instead.
knn <- train(goal ~. -ID , data = train_data,
             method = "knn", preProcess = c("center","scale"),
             trControl = cont , tuneGrid = expand.grid(k = seq(1, 20, 2)))

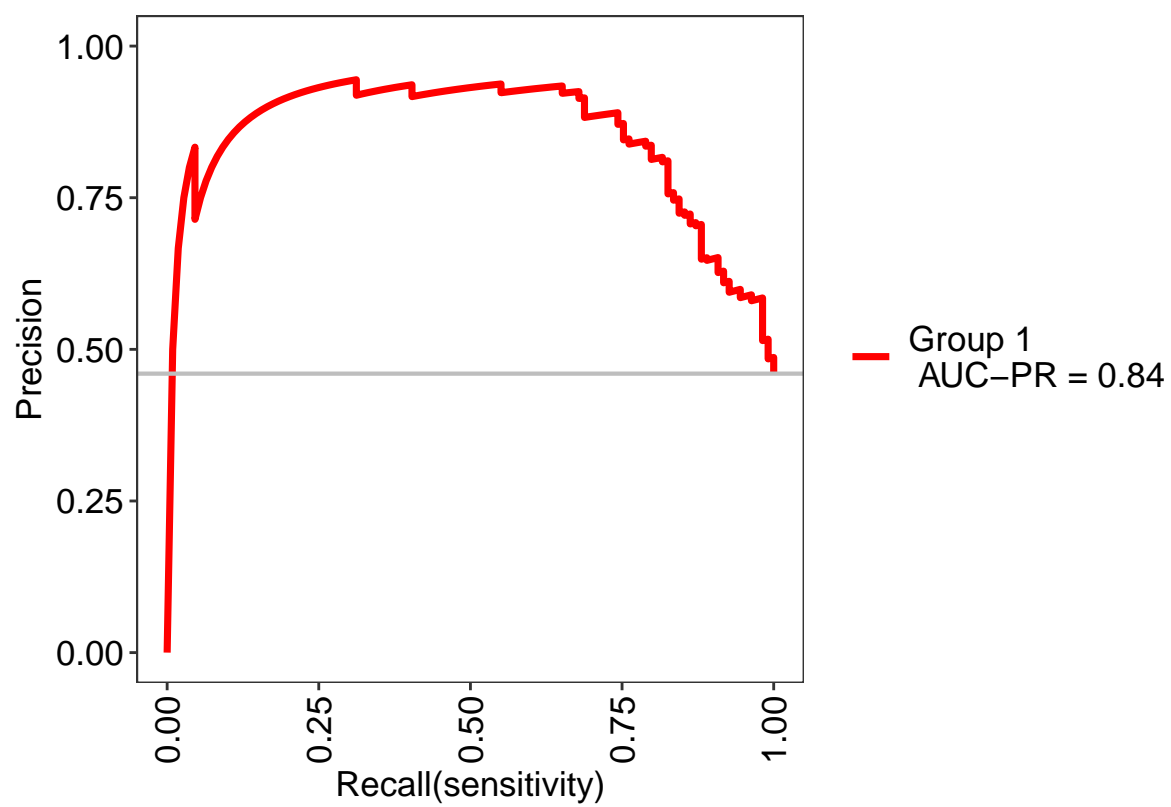
## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" was not
## in the result set. ROC will be used instead.

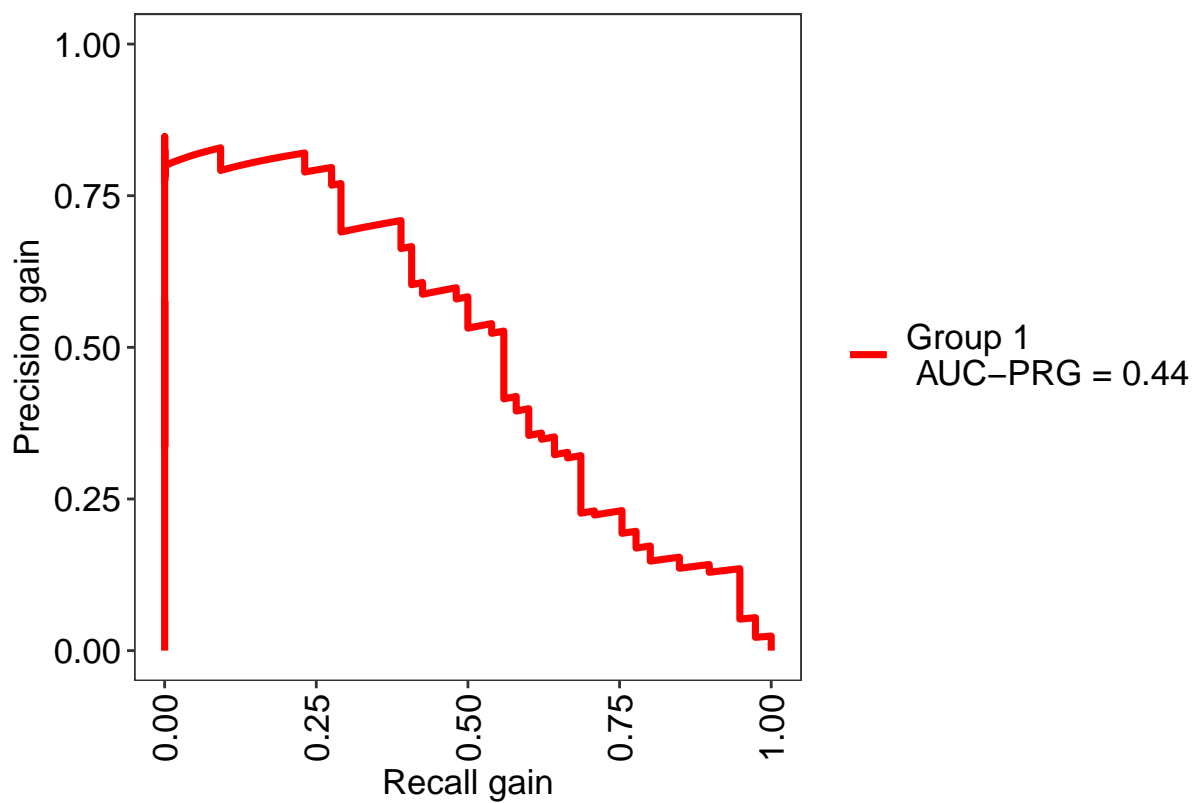
#####
##          Logistic Classifier: Performance Plots          ##
#####
res.log <- evalm(log)

## ***MLeval: Machine Learning Model Evaluation***
## Input: caret train function object
## Not averaging probs.
## Group 1 type: cv
## Observations: 237
## Number of groups: 1
## Observations per group: 237
## Positive: unhealthy
## Negative: healthy
## Group: Group 1
## Positive: 109
## Negative: 128
## ***Performance Metrics***

```

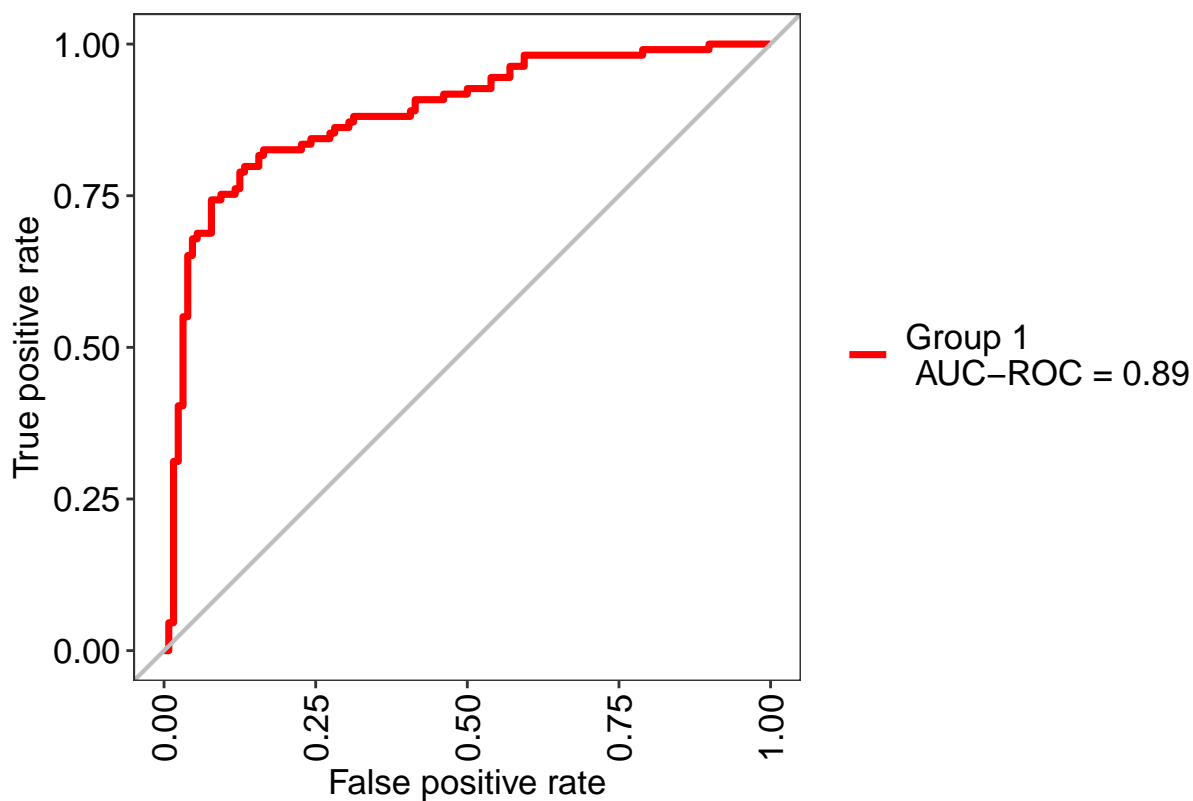






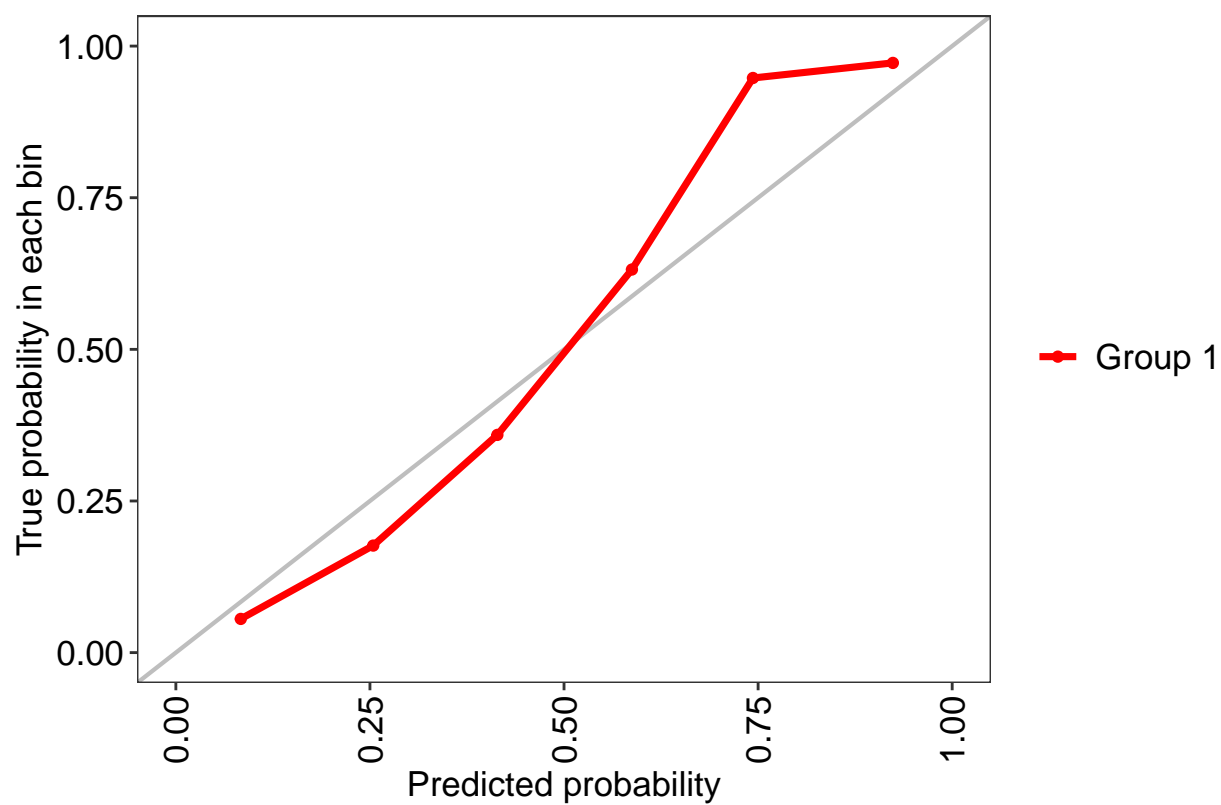
## Group 1 Optimal Informedness = 0.665352637614679

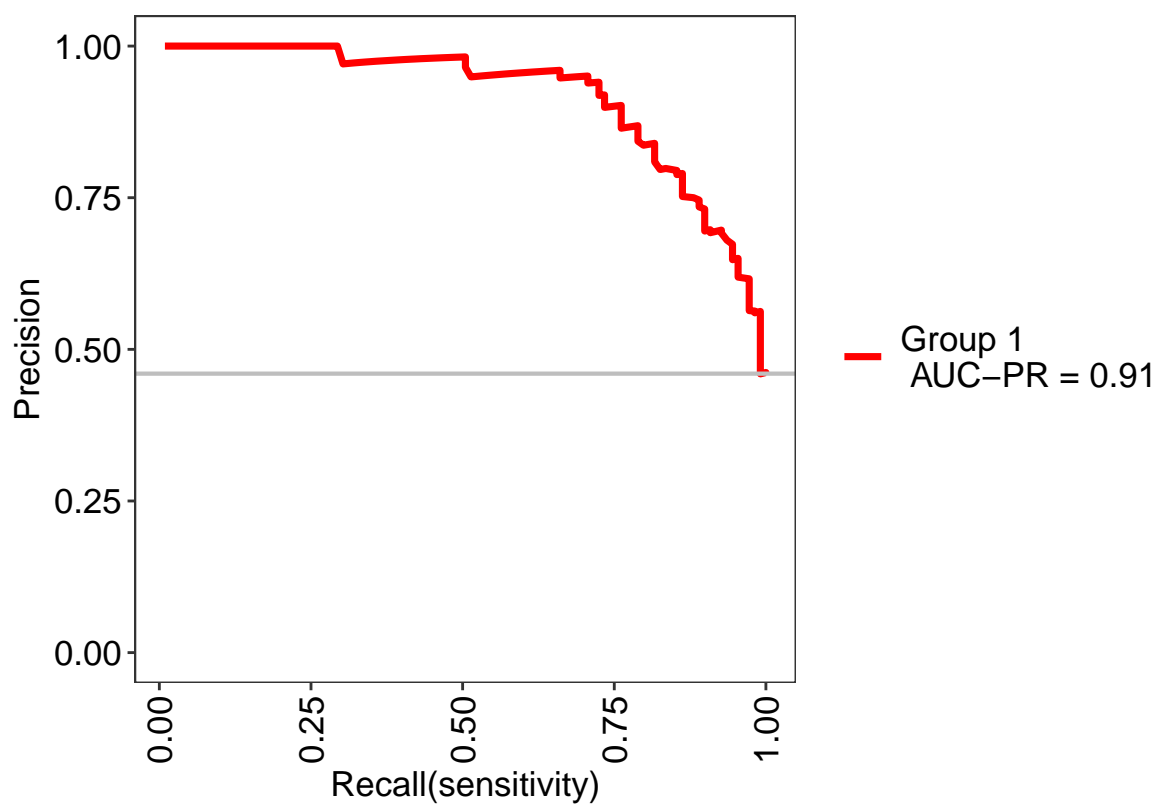
## Group 1 AUC-ROC = 0.89



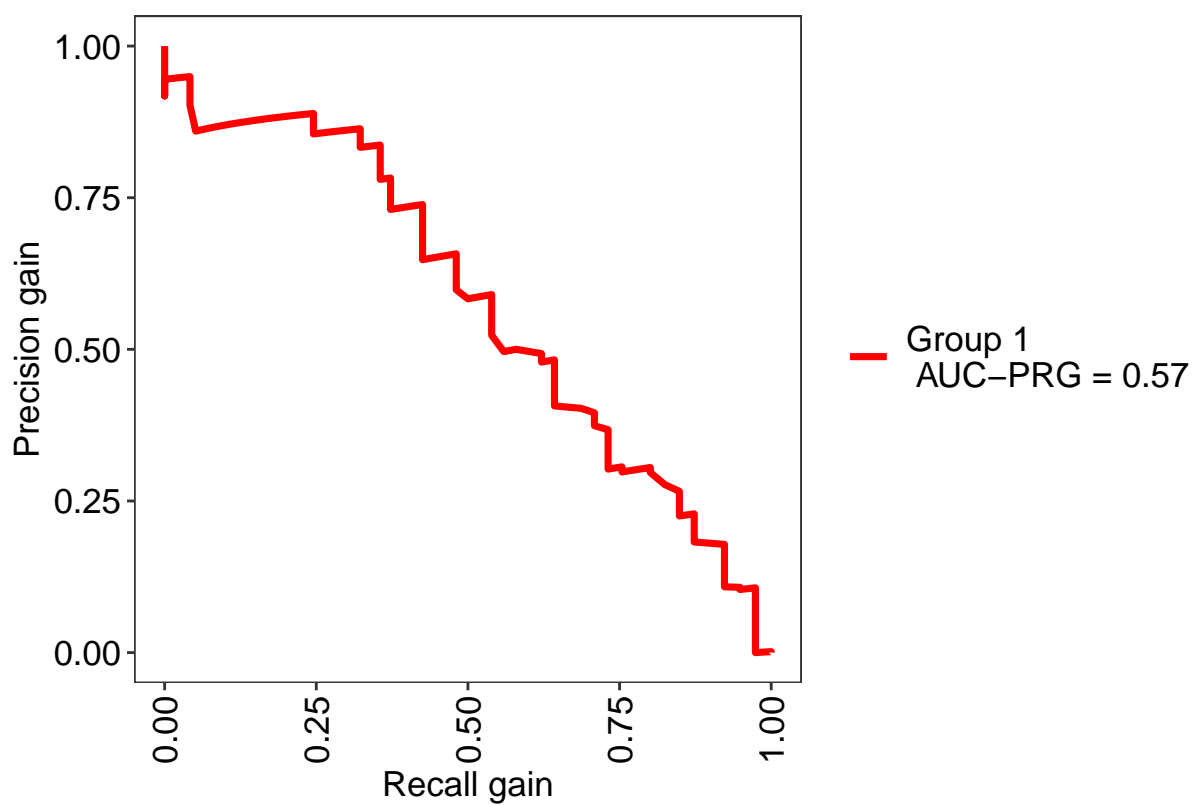
```
#####
##      Random Forest Classifier: Performance Plots      ##
#####
res.rf <- evalm(rf)

## ***MLevel: Machine Learning Model Evaluation***
## Input: caret train function object
## Not averaging probs.
## Group 1 type: cv
## Observations: 237
## Number of groups: 1
## Observations per group: 237
## Positive: unhealthy
## Negative: healthy
## Group: Group 1
## Positive: 109
## Negative: 128
## ***Performance Metrics***
```



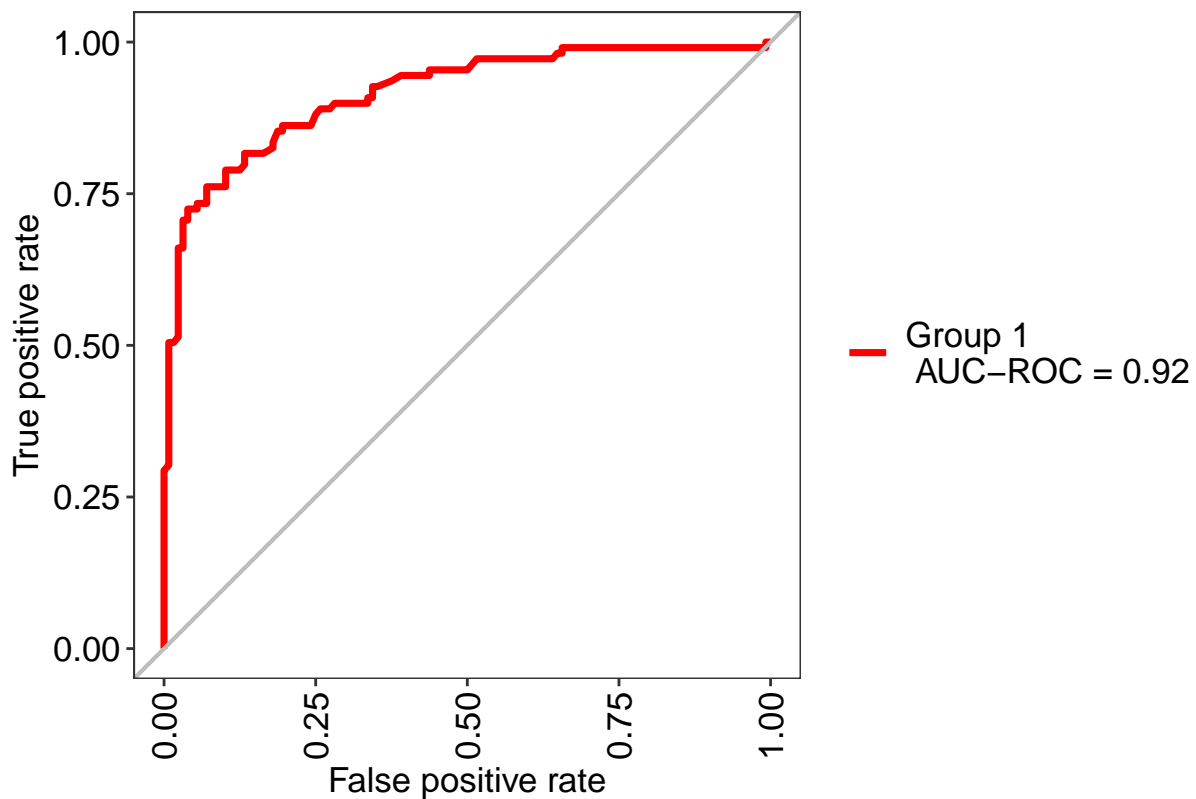






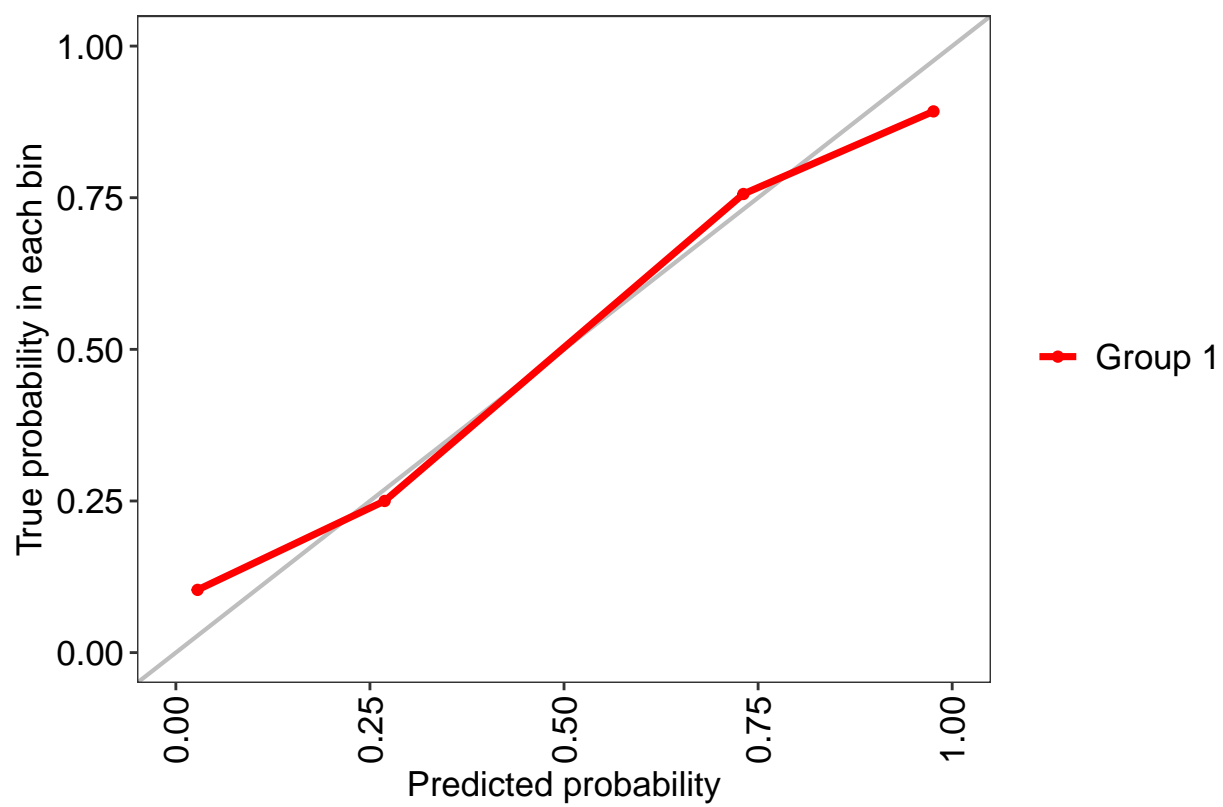
## Group 1 Optimal Informedness = 0.691155389908257

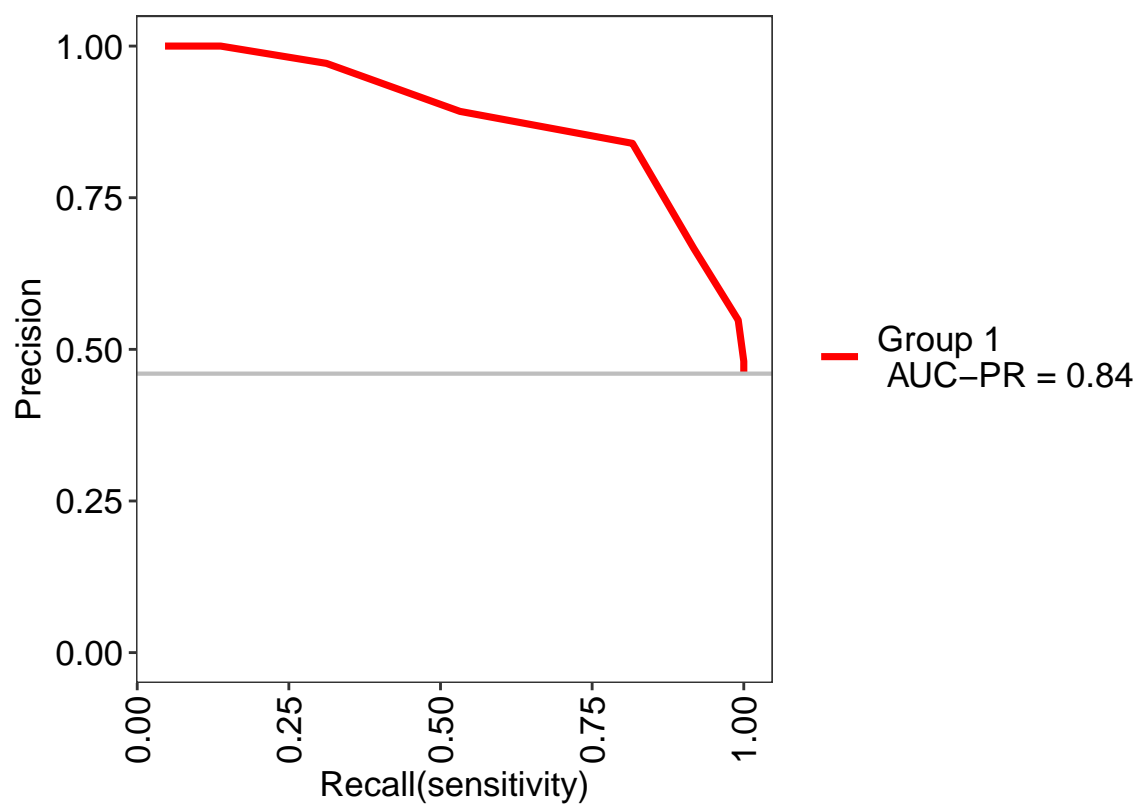
## Group 1 AUC-ROC = 0.92

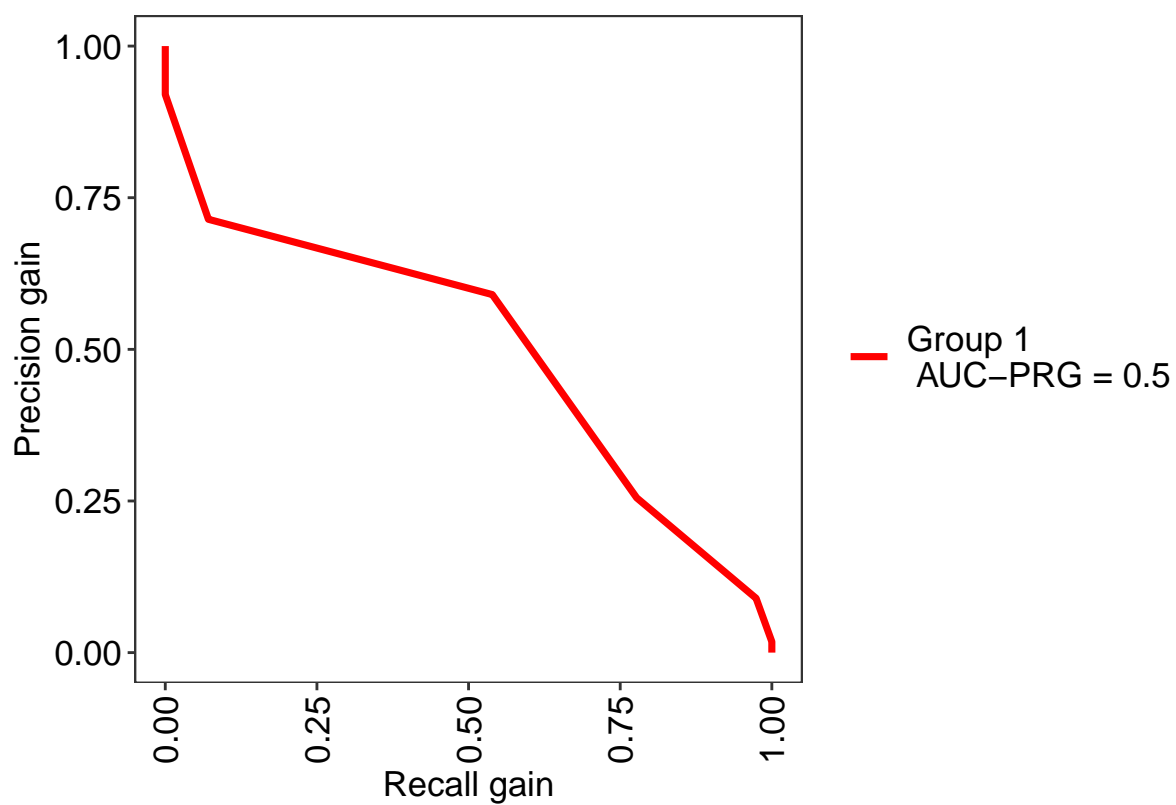


```
#####
##           Boosted logit Classifier: Performance Plots           ##
#####
res.blog <- evalm(blog)

## ***MLeval: Machine Learning Model Evaluation***
## Input: caret train function object
## Not averaging probs.
## Group 1 type: cv
## Observations: 237
## Number of groups: 1
## Observations per group: 237
## Positive: unhealthy
## Negative: healthy
## Group: Group 1
## Positive: 109
## Negative: 128
## ***Performance Metrics***
```

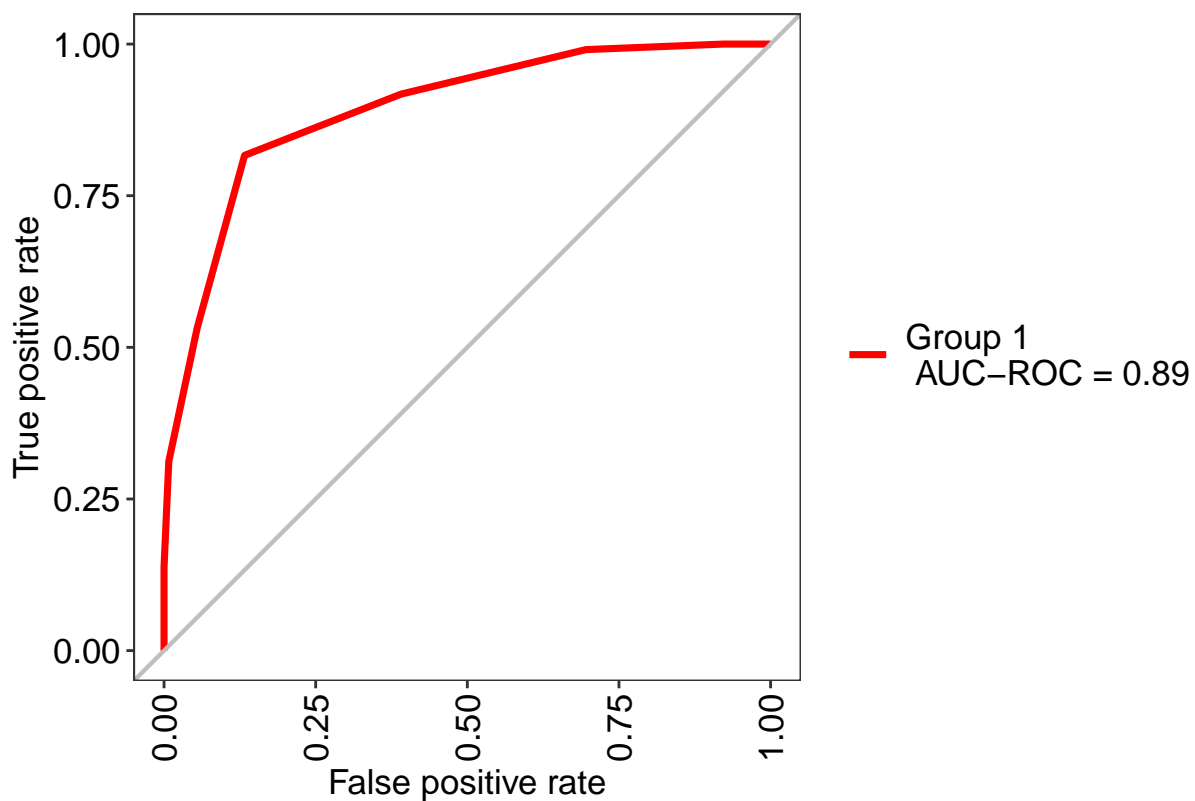






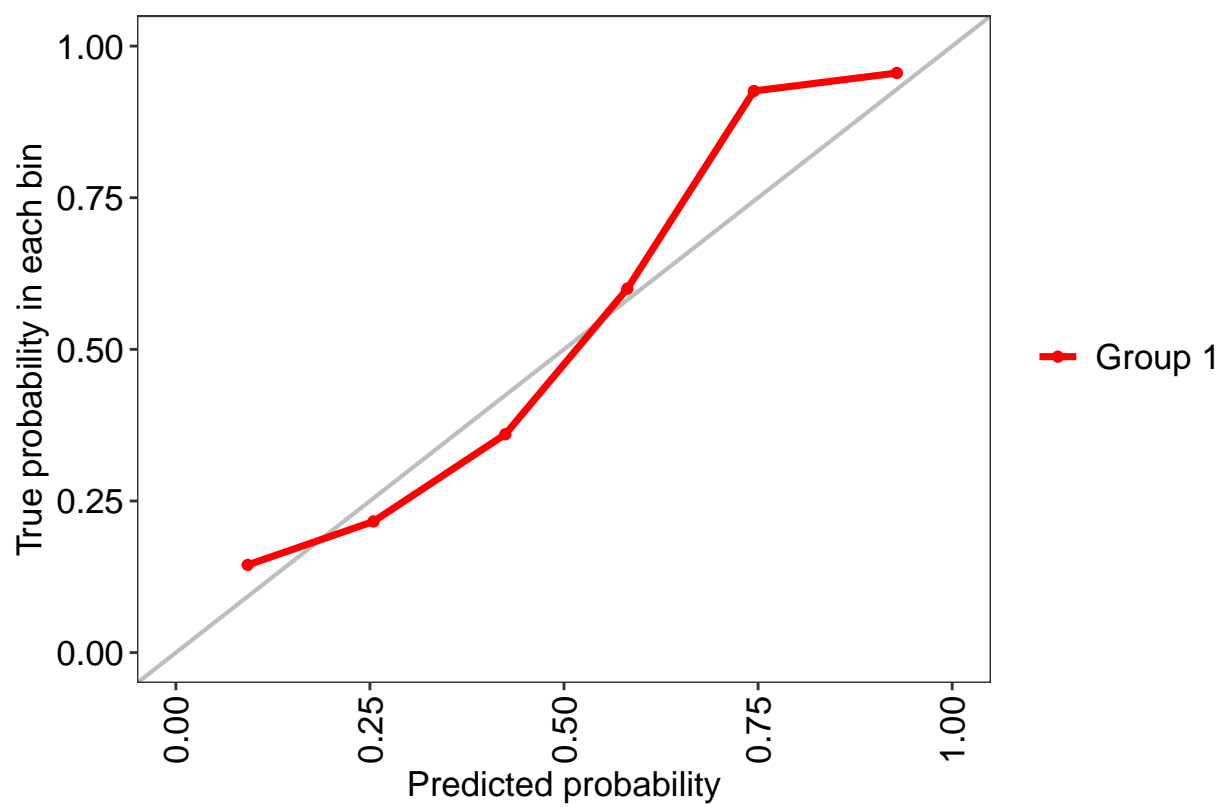
## Group 1 Optimal Informedness = 0.68370126146789

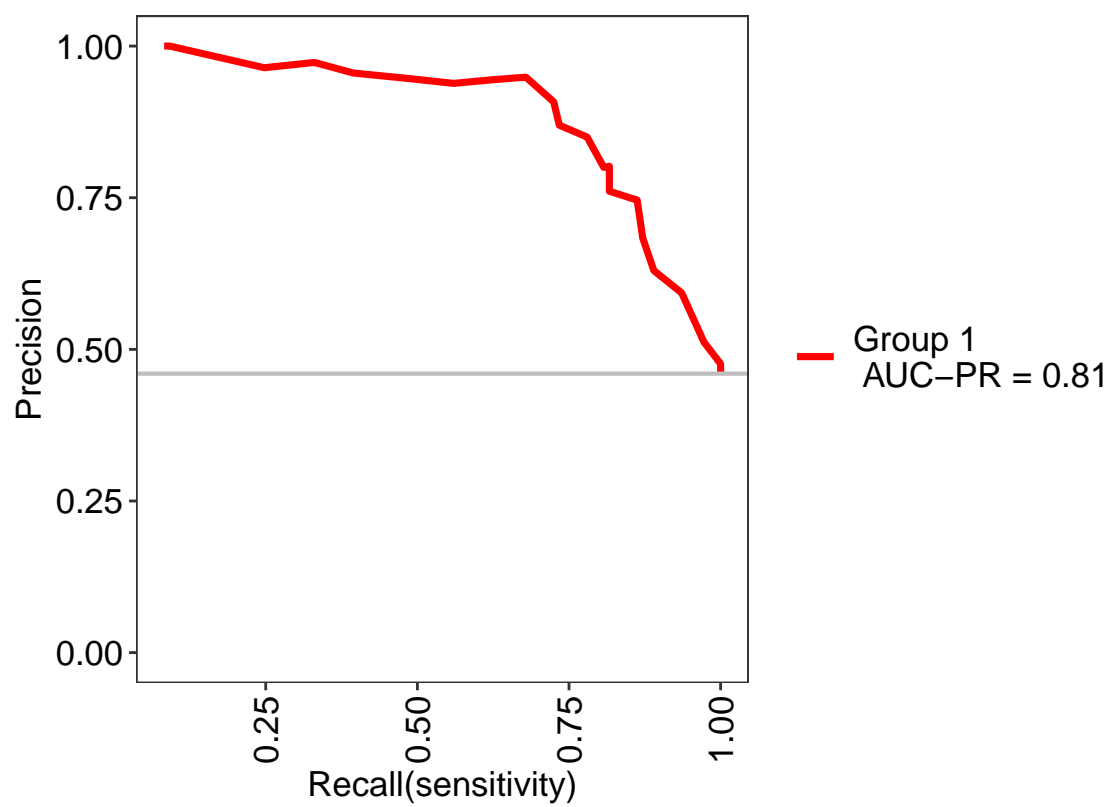
## Group 1 AUC-ROC = 0.89



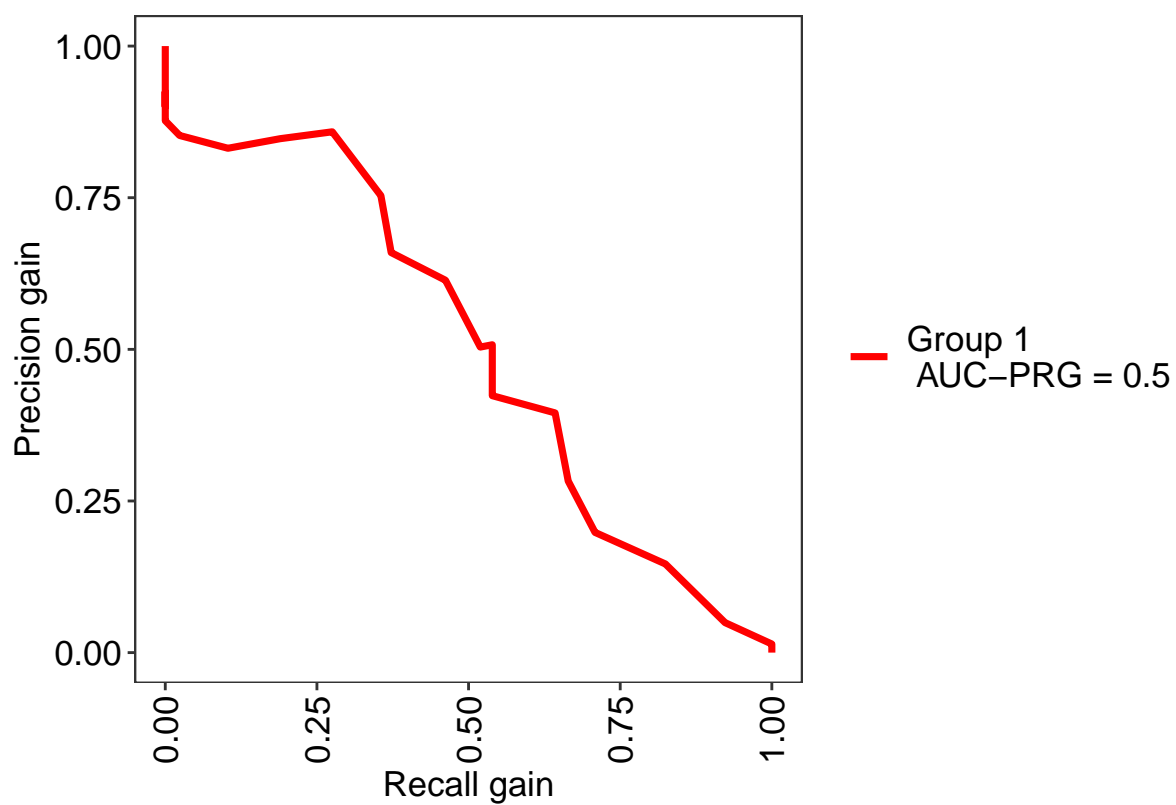
```
#####
##          k-NN Classifier: Performance Plots          ##
#####
res.knn <- evalm(knn)

## ***MLevel: Machine Learning Model Evaluation***
## Input: caret train function object
## Not averaging probs.
## Group 1 type: cv
## Observations: 237
## Number of groups: 1
## Observations per group: 237
## Positive: unhealthy
## Negative: healthy
## Group: Group 1
## Positive: 109
## Negative: 128
## ***Performance Metrics***
```









## Group 1 Optimal Informedness = 0.662629013761468

## Group 1 AUC-ROC = 0.89

