EDA_Classfication.R

abhinavmishra

2022-11-15

```
##            A script for exploratory data analysis, and            -
##             classification training four 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", "RColorBrewer"))

library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.4.0      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(MLeval)
library(RColorBrewer)
```

```
##                   Loading data from the file                        -
```

```
processedWithHeader_cleveland <- read_csv("~/Documents/Freie/IFA/WiSe 22-23/Data Science/Week 2/processe
                                          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 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)
```

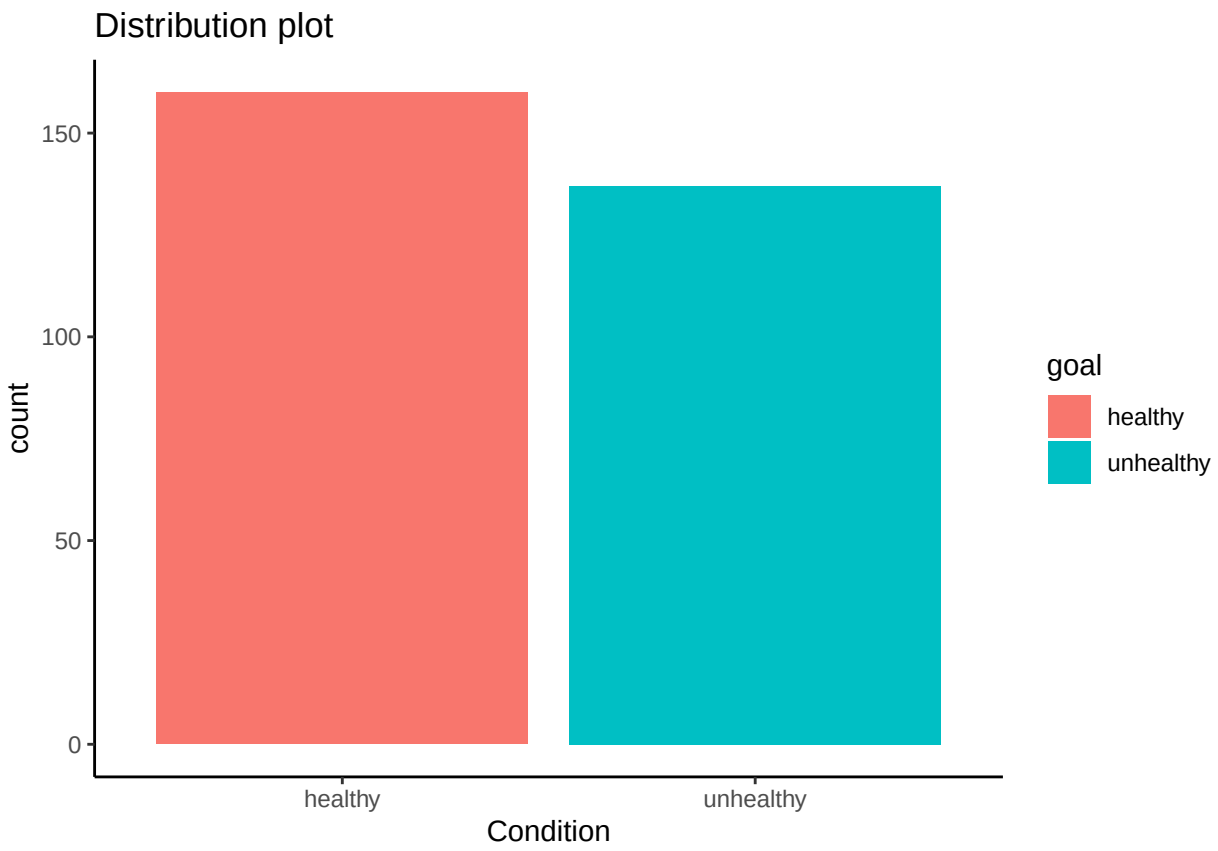
```
##       age        sex           cp          trestbps          chol
##  Min.   :29.00   F: 96   Min.   :1.000   Min.   : 94.0   Min.   :126.0
##  1st Qu.:48.00   M:201   1st Qu.:3.000   1st Qu.:120.0   1st Qu.:211.0
##  Median :56.00           Median :3.000   Median :130.0   Median :243.0
##  Mean   :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")
log_pred <- predict(log_fit, test_data)
confusionMatrix(log_pred, test_data$goal)
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  healthy unhealthy
##   healthy        27         3
##   unhealthy       5        25
##
##                Accuracy : 0.8667
##                  95% CI : (0.7541, 0.9406)
##     No Information Rate : 0.5333
##     P-Value [Acc > NIR] : 4.403e-08
##
##                   Kappa : 0.7333
##
##  Mcnemar's Test P-Value : 0.7237
##
##             Sensitivity : 0.8438
##             Specificity : 0.8929
##          Pos Pred Value : 0.9000
##          Neg Pred Value : 0.8333
##              Prevalence : 0.5333
##          Detection Rate : 0.4500
##    Detection Prevalence : 0.5000
##       Balanced Accuracy : 0.8683
##
##        '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         4
```

```
##   unhealthy       5        24
##
##                Accuracy : 0.85
##                  95% CI : (0.7343, 0.929)
##     No Information Rate : 0.5333
##     P-Value [Acc > NIR] : 2.293e-07
##
##                   Kappa : 0.6993
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8438
##             Specificity : 0.8571
##          Pos Pred Value : 0.8710
##          Neg Pred Value : 0.8276
##              Prevalence : 0.5333
##          Detection Rate : 0.4500
##    Detection Prevalence : 0.5167
##       Balanced Accuracy : 0.8504
##
##        '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.              -
```

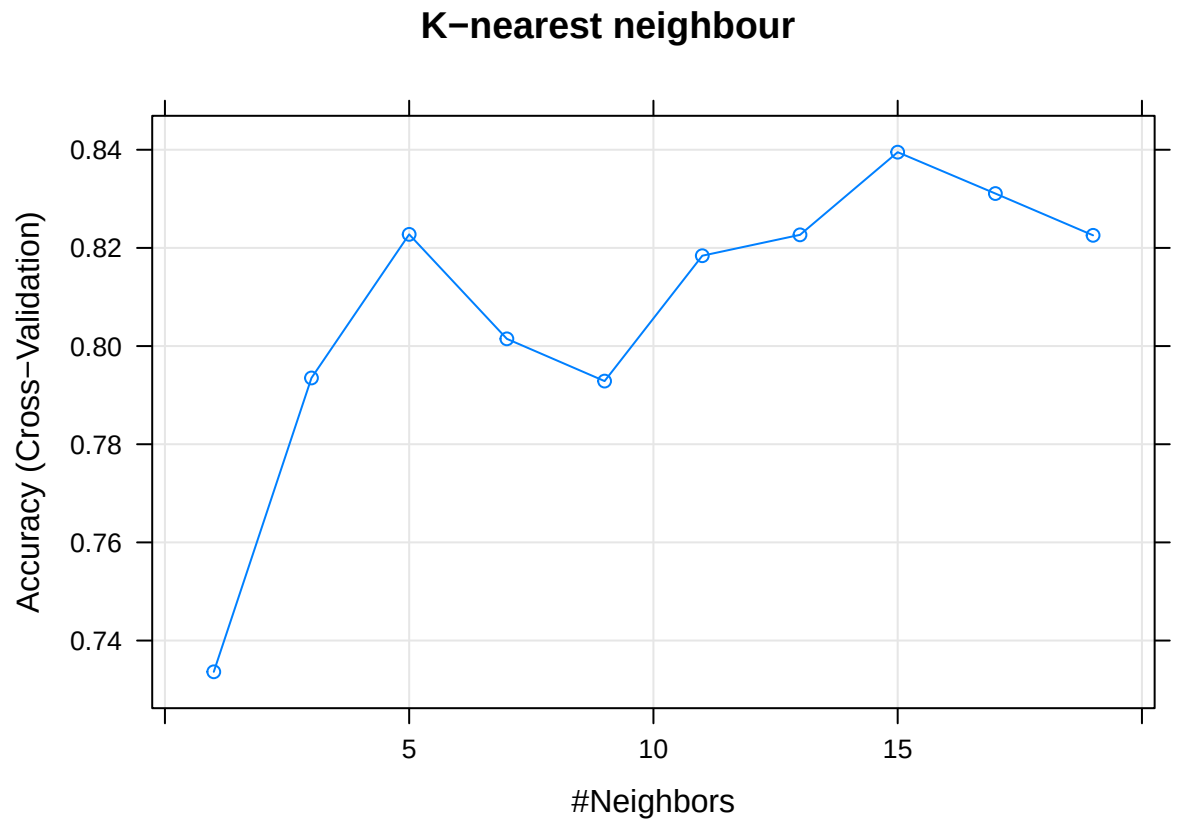
```
set.seed(112)
blog_fit <- train(goal ~.-ID,
                  data = train_data,
                  method = "LogitBoost")
blog_pred <- predict(blog_fit, test_data)
confusionMatrix(blog_pred, test_data$goal)
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  healthy unhealthy
##   healthy        22         6
##   unhealthy      10        22
##
##                Accuracy : 0.7333
##                  95% CI : (0.6034, 0.8393)
##     No Information Rate : 0.5333
##     P-Value [Acc > NIR] : 0.001201
##
##                   Kappa : 0.469
##
##  Mcnemar's Test P-Value : 0.453255
##
##             Sensitivity : 0.6875
##             Specificity : 0.7857
##          Pos Pred Value : 0.7857
```

```
##          Neg Pred Value : 0.6875
##              Prevalence : 0.5333
##          Detection Rate : 0.3667
##    Detection Prevalence : 0.4667
##       Balanced Accuracy : 0.7366
##
##        '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")
```




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

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  healthy unhealthy
##   healthy        28         5
##   unhealthy       4        23
```

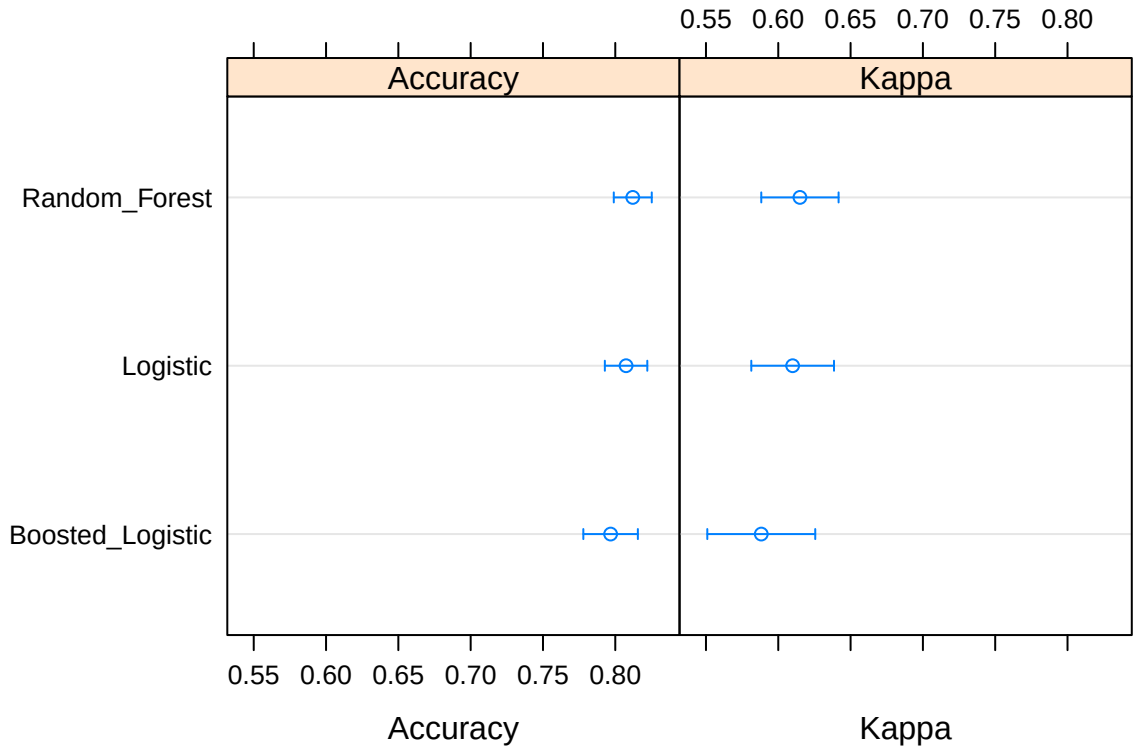
```
##
##                Accuracy : 0.85
##                  95% CI : (0.7343, 0.929)
##     No Information Rate : 0.5333
##     P-Value [Acc > NIR] : 2.293e-07
##
##                   Kappa : 0.698
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8750
##             Specificity : 0.8214
##          Pos Pred Value : 0.8485
##          Neg Pred Value : 0.8519
##              Prevalence : 0.5333
##          Detection Rate : 0.4667
##    Detection Prevalence : 0.5500
##       Balanced Accuracy : 0.8482
##
##        '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         0.7311828 0.7820513 0.8170732 0.8074614 0.8333333 0.8777778
## Random_Forest    0.7333333 0.7931034 0.8160920 0.8120950 0.8333333 0.8666667
## Boosted_Logistic 0.7111111 0.7666667 0.7954545 0.7967605 0.8333333 0.8666667
##                  NA's
## Logistic            0
## Random_Forest       0
## Boosted_Logistic    0
##
## Kappa
##                      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## Logistic         0.4576627 0.5641026 0.6274985 0.6099358 0.6576525 0.7385103
## Random_Forest    0.4666667 0.5645161 0.6245955 0.6149778 0.6606335 0.7322757
## Boosted_Logistic 0.4222222 0.5267902 0.5882353 0.5882428 0.6620690 0.7258883
##                  NA's
## Logistic            0
## Random_Forest       0
## Boosted_Logistic    0
```

```
dotplot(results)
```

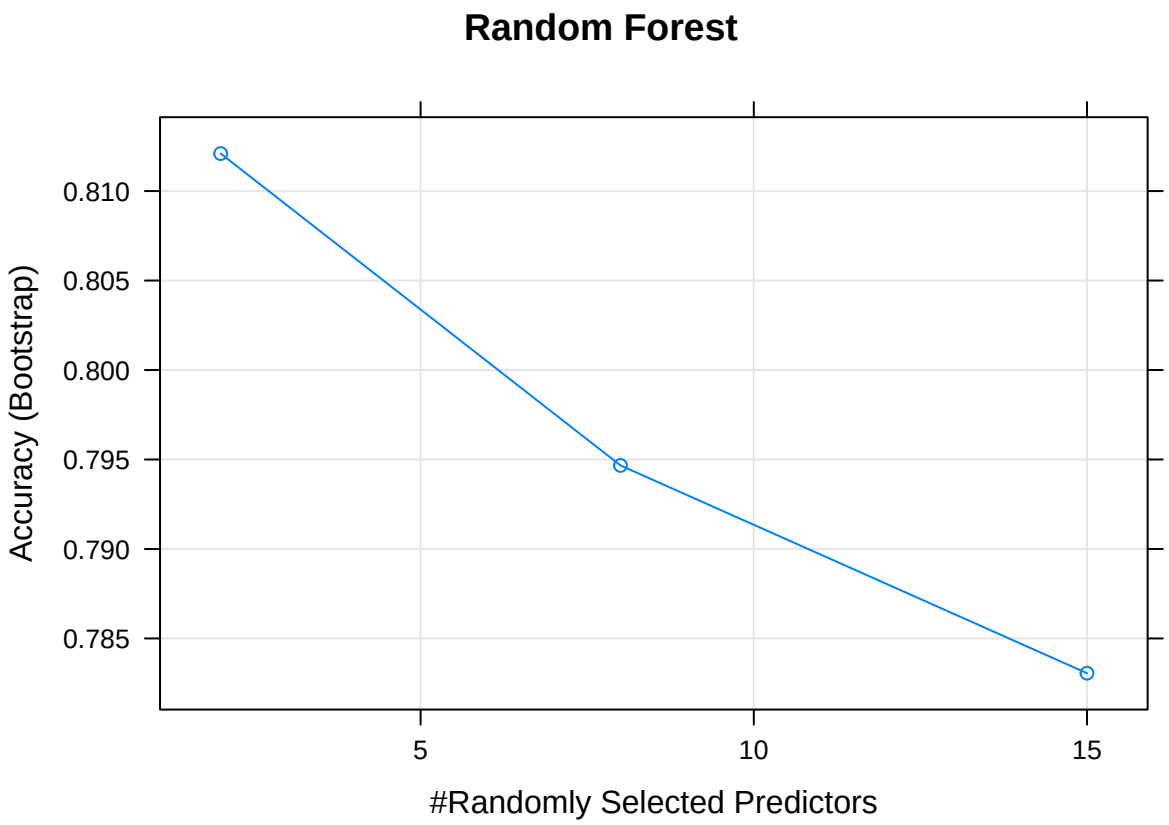


Confidence Level: 0.95

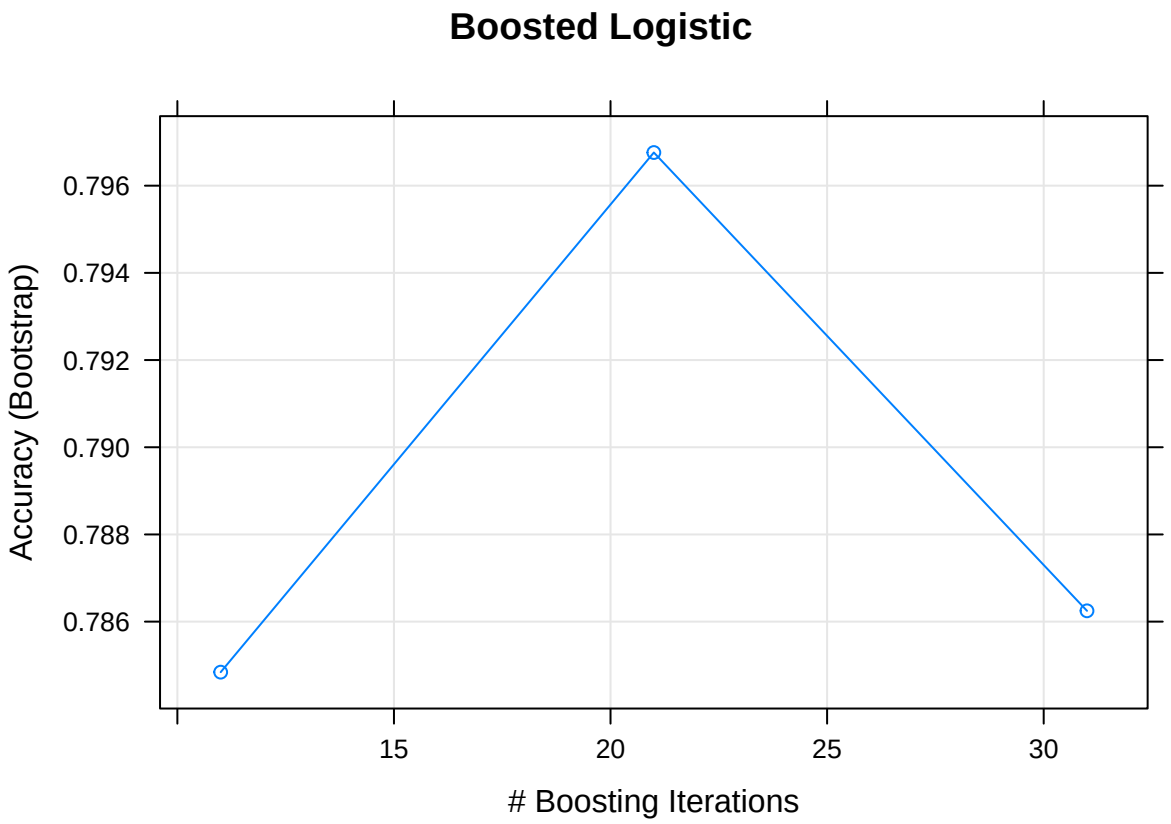
```
cf_rf <- confusionMatrix(rf_pred, test_data$goal)
cf_log <- confusionMatrix(log_pred, test_data$goal)
cf_blog <- confusionMatrix(blog_pred, test_data$goal)
cf_knn <- confusionMatrix(knn_pred, test_data$goal)

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("Log", "RF", "LogitB", "KNN")
pf_result <- as.matrix(pf_result)
pf_result
```

```
##                  Log       RF   LogitB      KNN
## Accuracy    86.66667 85.00000 73.33333 85.00000
## Specificity 89.28571 85.71429 78.57143 82.14286
## Sensitivity 84.37500 84.37500 68.75000 87.50000
```

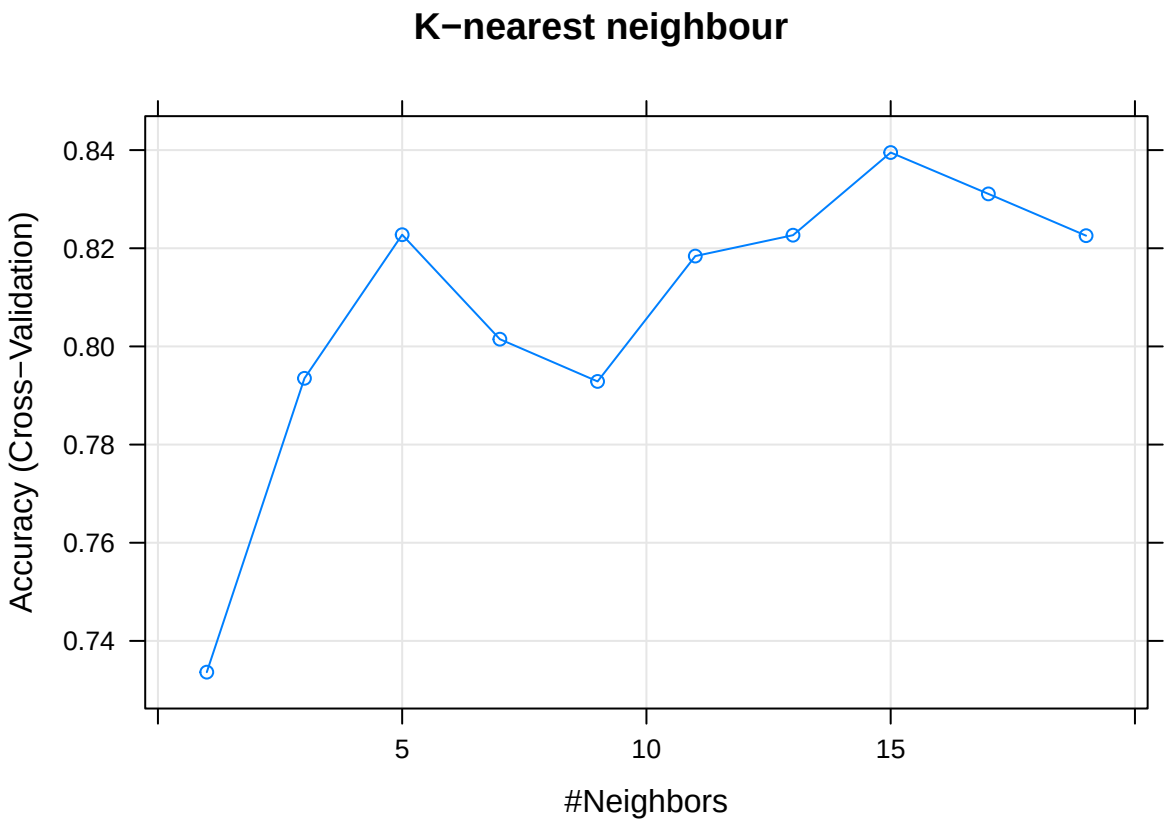
```
plot(rf_fit, main = "Random Forest")
```



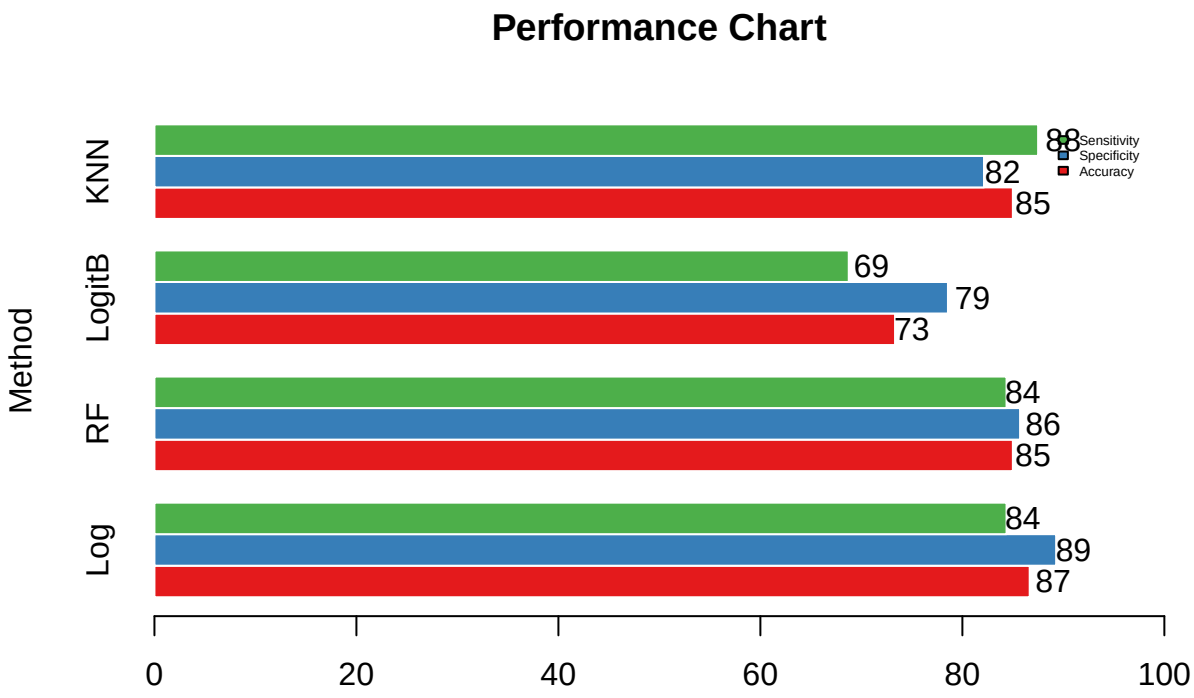

```
plot(blog_fit, main = "Boosted Logistic")
```




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

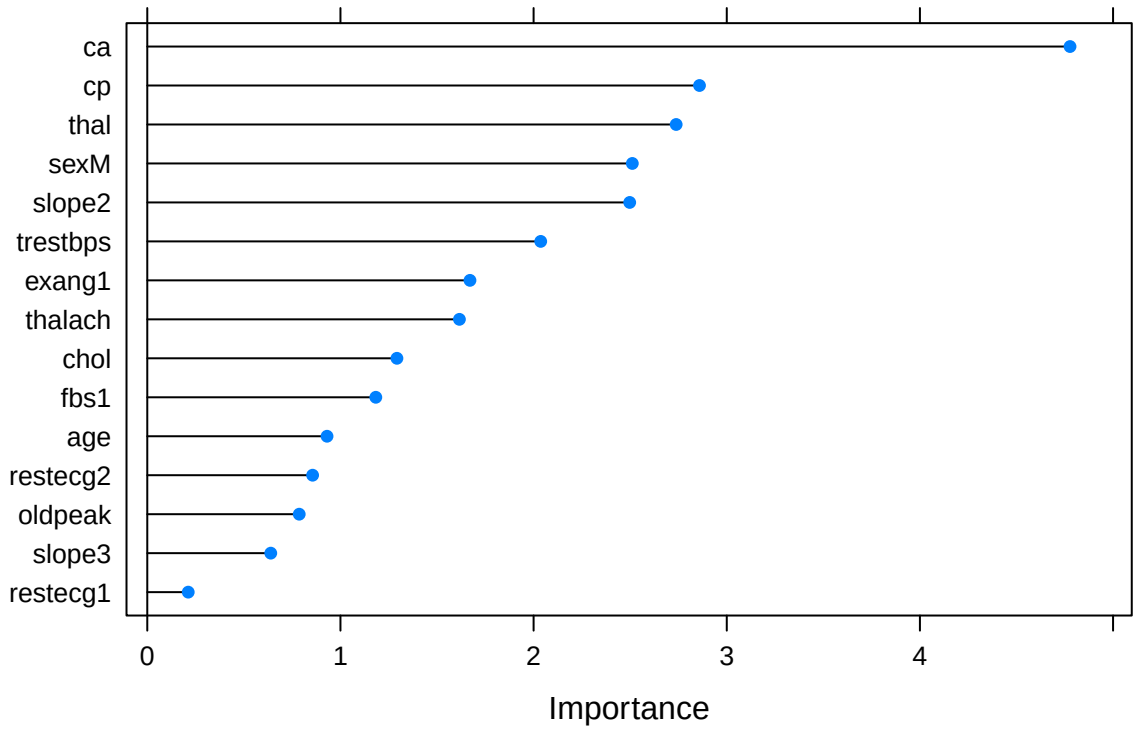



```
y <- barplot(pf_result, beside = TRUE, horiz = TRUE,
             col=brewer.pal(3,"Set1"),border="white",
             legend.text = c("Accuracy", "Specificity", "Sensitivity"),
             args.legend = list(bty = "n", cex = 0.4), xlim=c(0,100),
             main = "Performance Chart", ylab = "Method")

x <- round(pf_result)

text(x+2,y,labels=as.character(x))
```

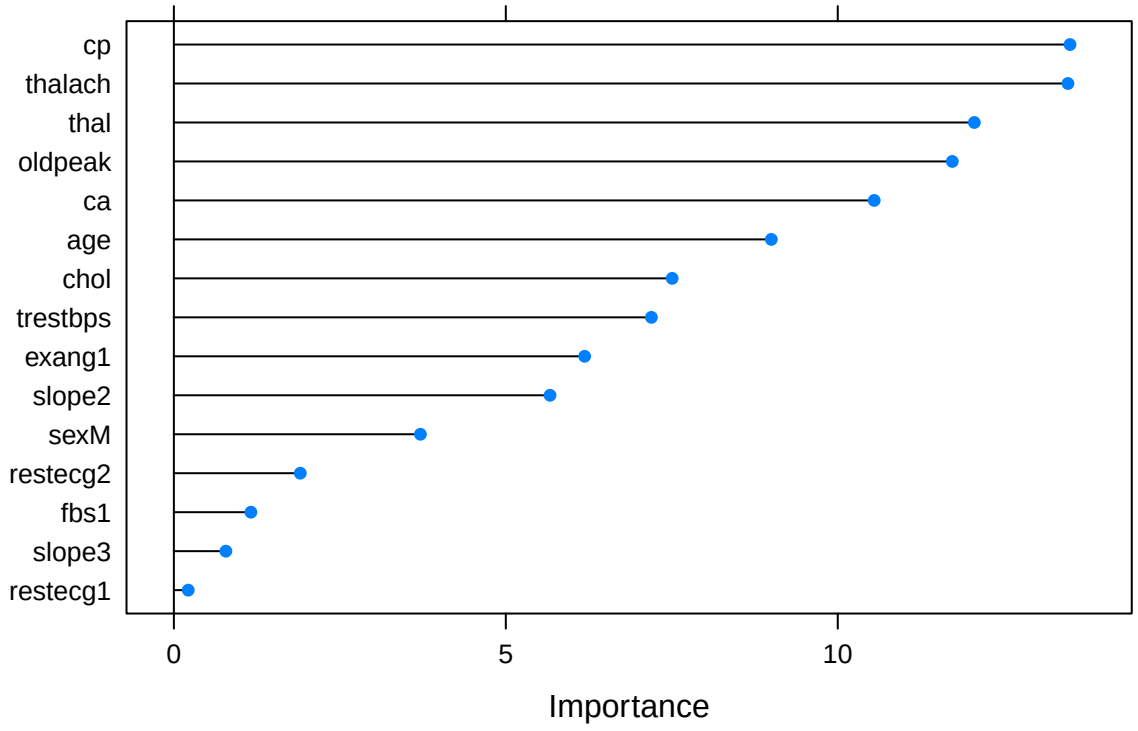



```
##                          Feature extraction                          -
```

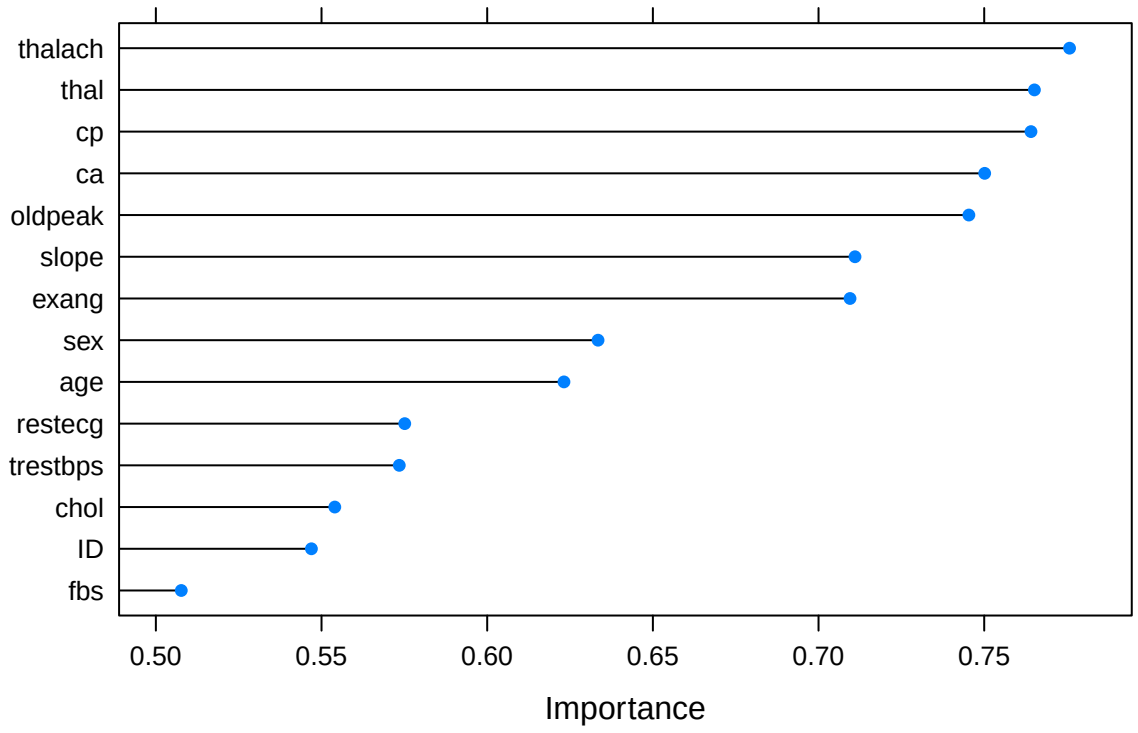
```
feat_log <- varImp(log_fit, scale = FALSE)
feat_rf <- varImp(rf_fit, scale = FALSE)
feat_blog <- varImp(blog_fit, scale = FALSE)
feat_knn <- varImp(knn_fit, scale = FALSE)

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



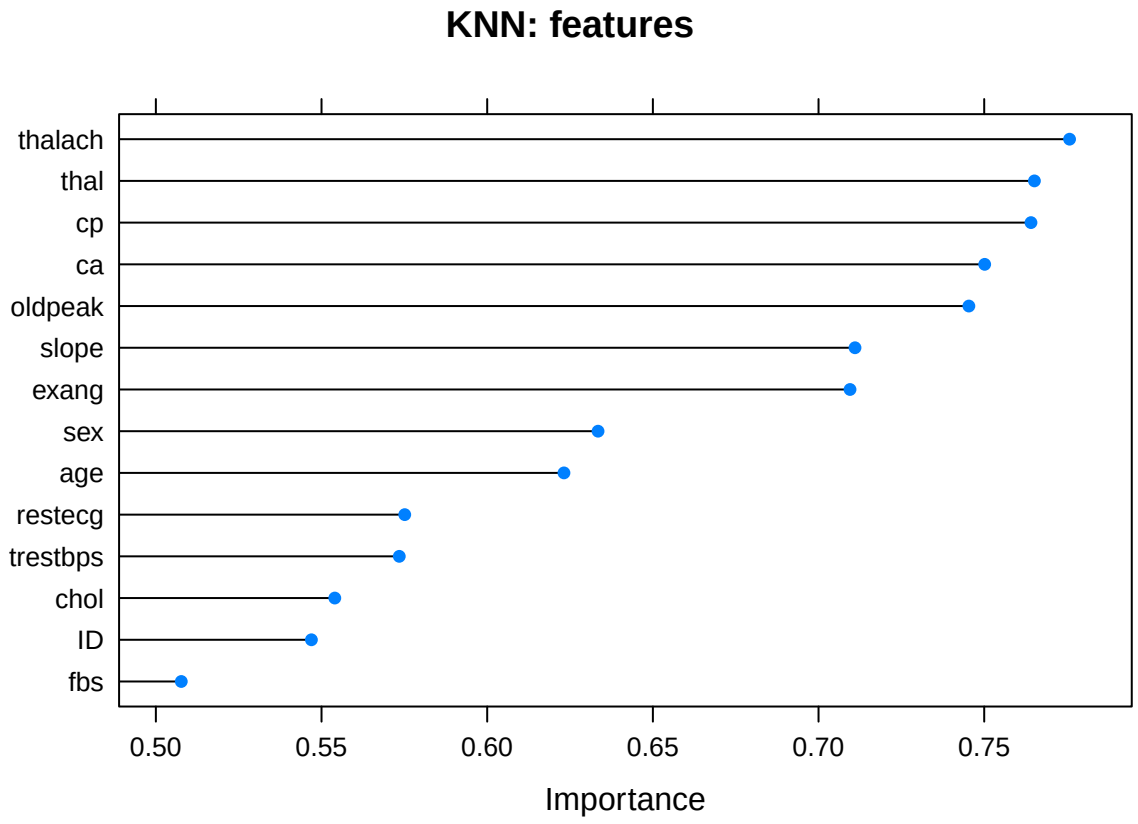

```
plot(feat_rf, main = "Random forest: features")
```




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




```
plot(feat_knn, main = "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)
```

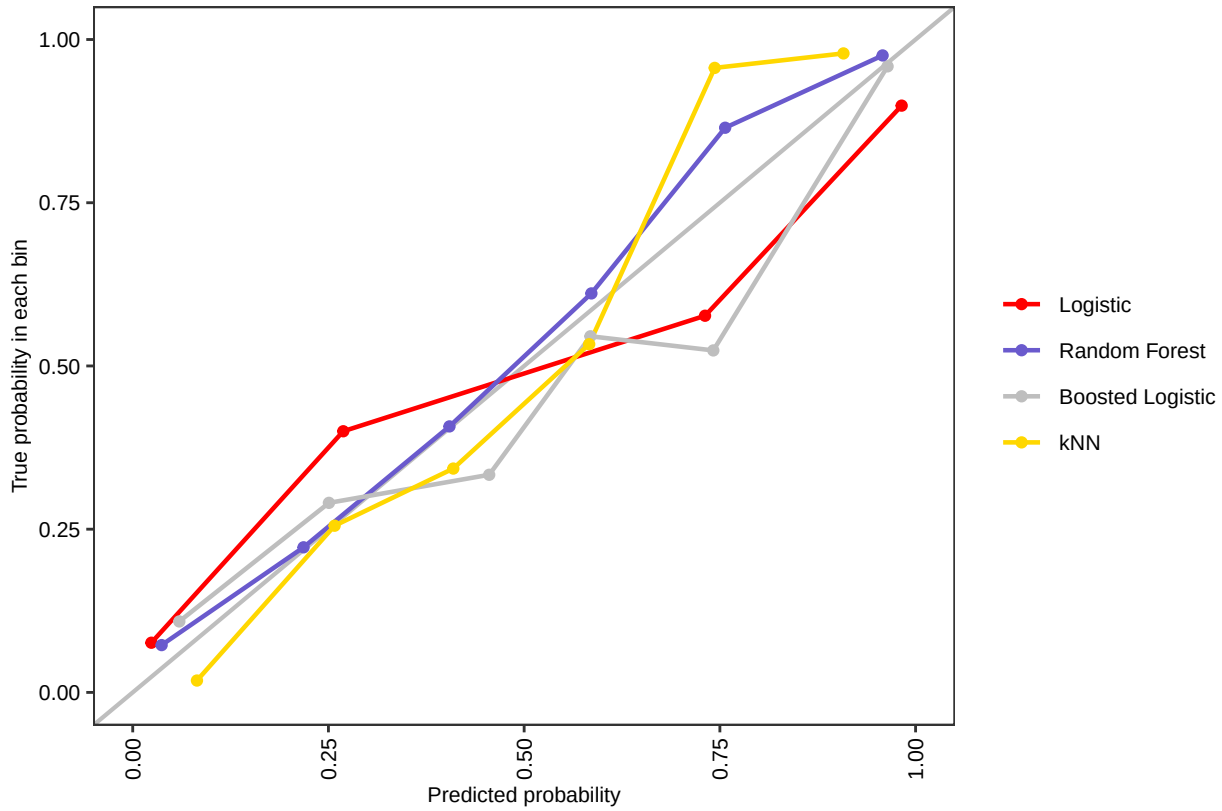
```
## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" was not
```

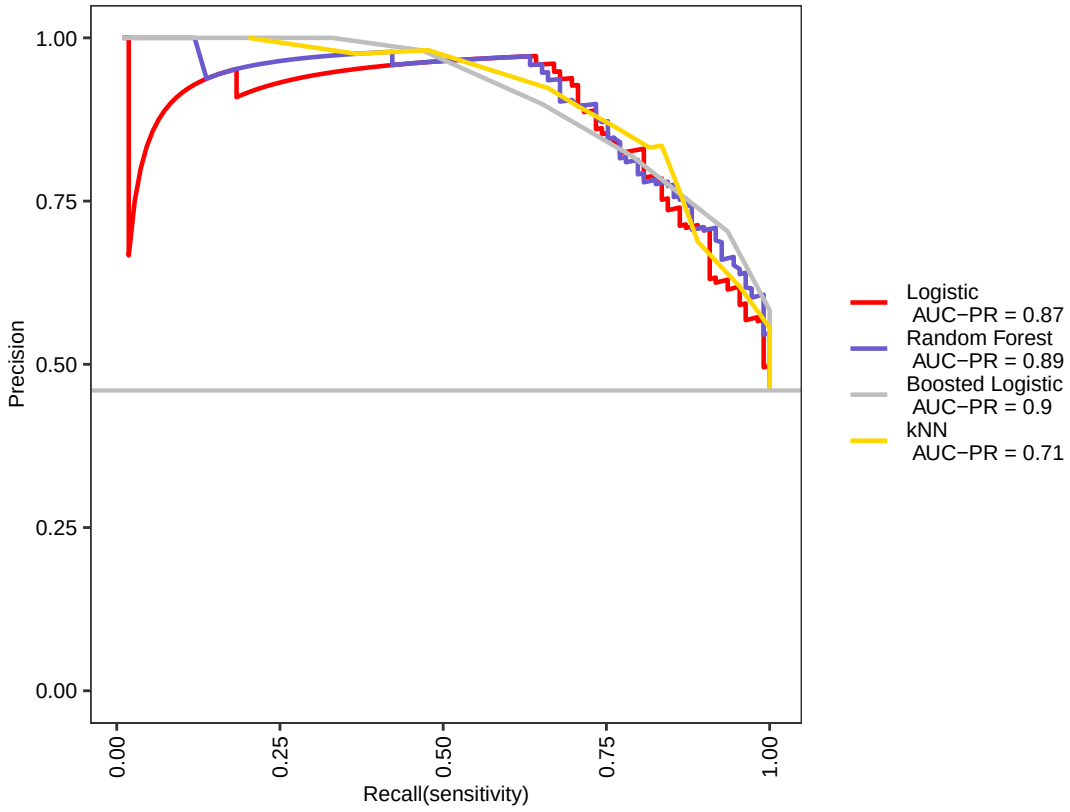
```
## in the result set. ROC will be used instead.
```

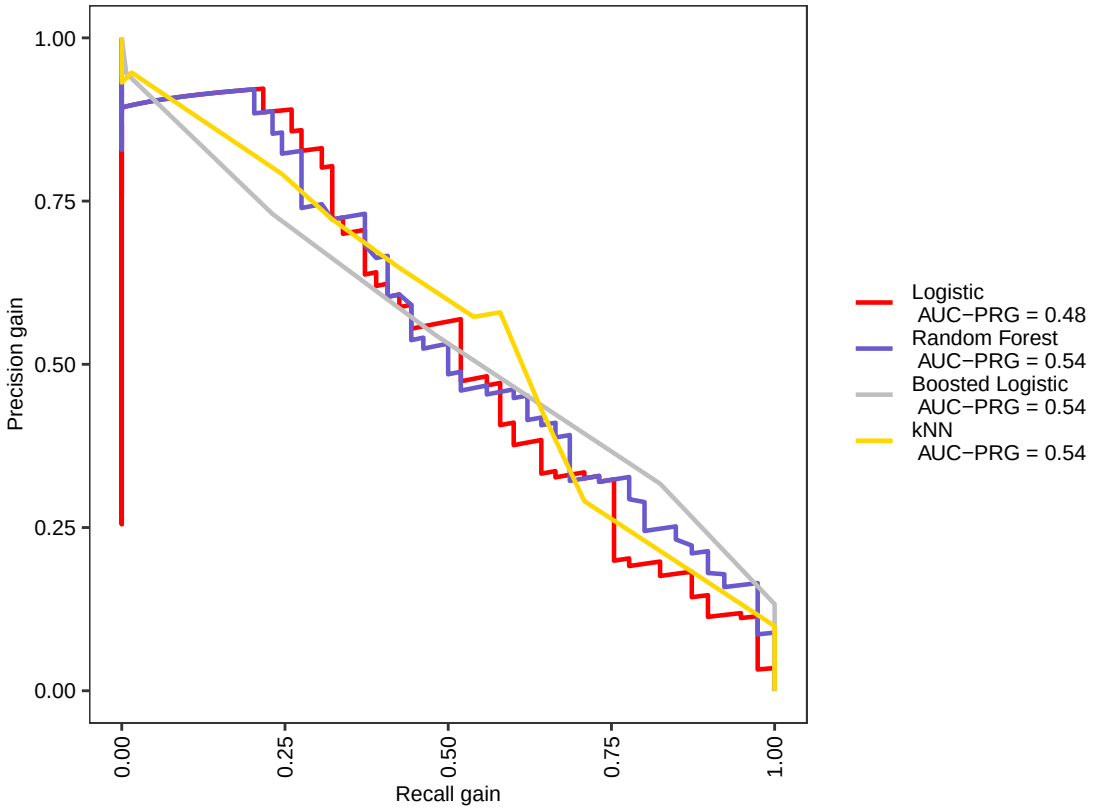
```
knn <- train(goal ~. -ID , data = train_data,
             method = "knn", preProcess = c("center","scale"),
             trControl = cont , tuneGrid = expand.grid(k = seq(1, 20, 2)))
```

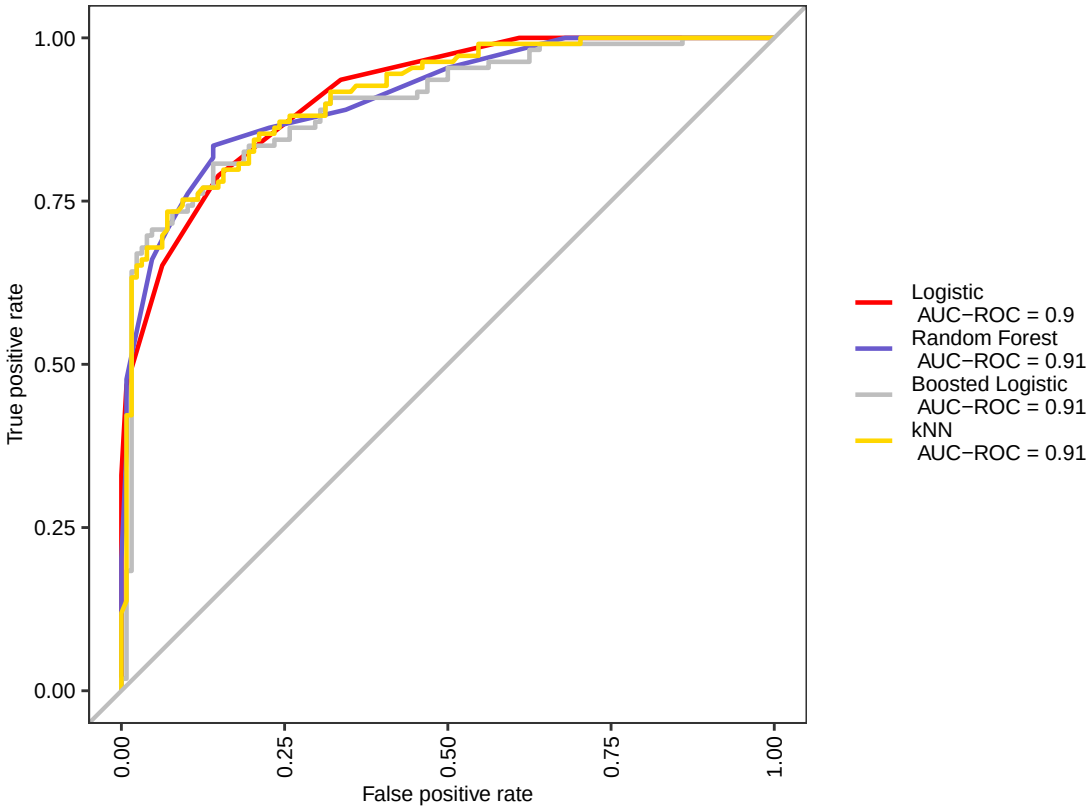
```
## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" was not
## in the result set. ROC will be used instead.
```

```
#####################################################################
##                              AUC-ROC                            ##
#####################################################################
metric <- evalm(list(log,rf,blog,knn),
      gnames=c('Logistic','Random Forest',
               'Boosted Logistic', 'kNN'),
      rlinethick=0.8,fsize=8, silent = TRUE)
```




1.00
0.75
0.50
0.25
0.00
Precision
0.00
0.25
0.50
0.75
1.00
Recall(sensitivity)
Logistic
AUC−PR = 0.87
Random Forest
AUC−PR = 0.89
Boosted Logistic
AUC−PR = 0.9
kNN
AUC−PR = 0.71


1.00
0.75
0.50
0.25
0.00
Precision gain
0.00
0.25
0.50
0.75
1.00
Recall gain
Logistic
AUC−PRG = 0.48
Random Forest
AUC−PRG = 0.54
Boosted Logistic
AUC−PRG = 0.54
kNN
AUC−PRG = 0.54




```
ROC <- data.frame(round(rbind(log[["results"]][["ROC"]],
max(rf[["results"]][["ROC"]]),
max(blog[["results"]][["ROC"]]),
max(knn[["results"]][["ROC"]])),2))

colnames(ROC) <- "AUC-ROC"
row.names(ROC) <- c("Logistic","Random Forest","Boosted Logit","kNN")

ROC
```

```
##               AUC-ROC
## Logistic         0.90
## Random Forest    0.91
## Boosted Logit    0.92
## kNN              0.91
```

```
##                              EOL                              -
```