

heart_Classfication_rerun.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", "mice", "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(mice)
```

```

##
## Attaching package: 'mice'
##
## The following object is masked from 'package:stats':
##
##     filter
##
## The following objects are masked from 'package:base':
##
##     cbind, rbind
library(RColorBrewer)

## Loading data from the file -

processedWithHeader_cleveland <- read_csv("~/Documents/Freie/IFA/WiSe 22-23/Data Science/Week 4/processedWithHeader_cleveland.csv",
                                           show_col_types = FALSE, na = "?")

heart_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.csv')
table(heart_data$goal)

##
##     healthy unhealthy
##      164      139

## Descriptive Statistics + NA values omit -

str(heart_data)

```

```
## 'data.frame': 303 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 ...
```

```
summary(heart_data)
```

```
##      age      sex      cp      trestbps      chol
## Min.   :29.00   F: 97   Min.   :1.000   Min.   : 94.0   Min.   :126.0
## 1st Qu.:48.00   M:206   1st Qu.:3.000   1st Qu.:120.0   1st Qu.:211.0
## Median :56.00           Median :3.000   Median :130.0   Median :241.0
## Mean   :54.44           Mean   :3.158   Mean   :131.7   Mean   :246.7
## 3rd Qu.:61.00           3rd Qu.:4.000   3rd Qu.:140.0   3rd Qu.:275.0
## Max.   :77.00           Max.   :4.000   Max.   :200.0   Max.   :564.0
##
## fbs      restecg      thalach      exang      oldpeak      slope
## 0:258    0:151   Min.   : 71.0   0:204   Min.   :0.00   1:142
## 1: 45    1: 4    1st Qu.:133.5   1: 99   1st Qu.:0.00   2:140
##          2:148   Median :153.0           Median :0.80   3: 21
##          Mean   :149.6           Mean   :1.04
##          3rd Qu.:166.0           3rd Qu.:1.60
##          Max.   :202.0           Max.   :6.20
##
##      ca      thal      goal      ID
## Min.   :0.0000   Min.   :3.000   Length:303   Min.   : 1.0
## 1st Qu.:0.0000   1st Qu.:3.000   Class :character   1st Qu.: 76.5
## Median :0.0000   Median :3.000   Mode  :character   Median :152.0
## Mean   :0.6722   Mean   :4.734           Mean   :152.0
## 3rd Qu.:1.0000   3rd Qu.:7.000           3rd Qu.:227.5
## Max.   :3.0000   Max.   :7.000           Max.   :303.0
## NA's    :4      NA's    :2
```

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

```
## [1] 6
```

```
##      Missing Value: Impute      -
```

```
colnames(heart_data)[apply(heart_data, 2, anyNA)]
```

```
## [1] "ca" "thal"
```

```
### ca: number of major vessels (0-3) colored by flourosopy
### thal: 3 = normal; 6 = fixed defect; 7 = reversable defect
```

```
summary(heart_data$ca)
```

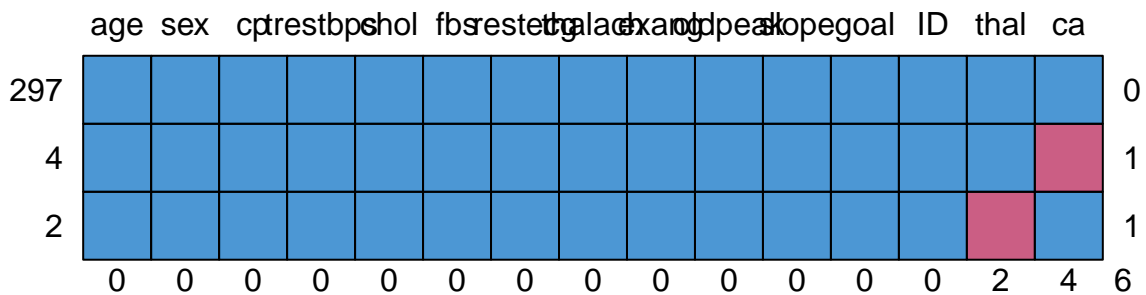
```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.     NA's
## 0.0000  0.0000  0.0000  0.6722  1.0000  3.0000         4
```

```
summary(heart_data$thal)
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.     NA's
## 3.000  3.000  3.000  4.734  7.000  7.000         2
```

```
### missing block
```

```
md.pattern(heart_data)
```



```
##      age sex cp trestbps chol fbs restecg thalach exang oldpeak slope goal ID
## 297  1  1  1      1      1  1      1      1      1      1      1      1
## 4    1  1  1      1      1  1      1      1      1      1      1      1
## 2    1  1  1      1      1  1      1      1      1      1      1      1
##      0  0  0      0      0  0      0      0      0      0      0      0
##      thal ca
## 297    1  1 0
## 4      1  0 1
## 2      0  1 1
##      2  4 6
```

```
### imputation
```

```
### m = 5 (iteration models)
```

```
impute <- mice(heart_data, m = 5, maxit = 50, method = "pmm",
               seed = 112, printFlag = FALSE)
```

```
## Warning: Number of logged events: 1
### 5 imputed dataset values
impute$imp$ca

##      1 2 3 4 5
## 167 0 0 0 0 2
## 193 1 0 0 0 0
## 288 1 0 1 1 0
## 303 0 0 0 0 0

impute$imp$thal

##      1 2 3 4 5
## 88   3 3 3 3 3
## 267 7 7 7 7 7

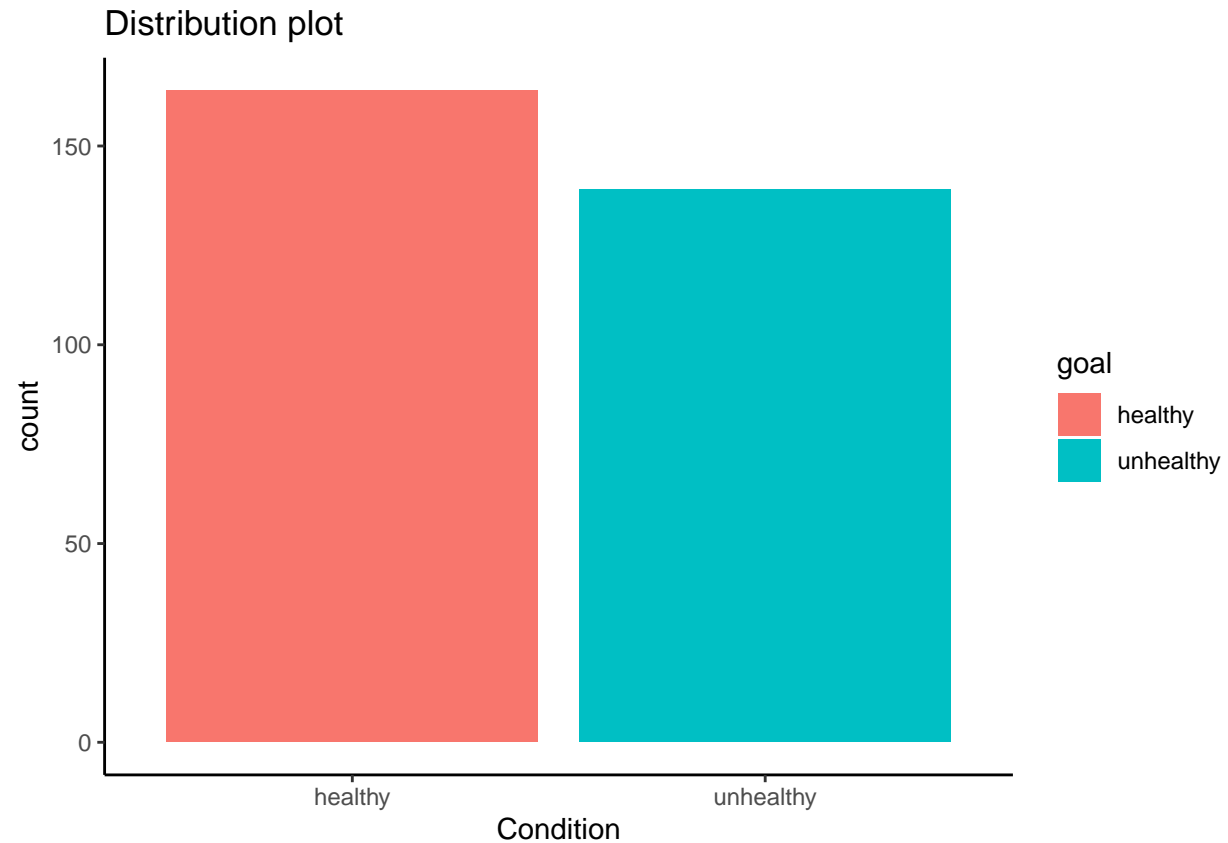
### appending the imputed values (second out of five)
heart_data <- complete(impute, 2)
### no missing values now
sum(is.na(heart_data))

## [1] 0

#####
##      No evidence of imbalanced classes (both datasets),      ##
##      hence, no change the composition of the training set      ##
#####

##                               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
##      164      139
```

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

```
##
##   healthy unhealthy
##      54.1      45.9
```

```
##                               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      31         4
## unhealthy     2        24
##
##              Accuracy : 0.9016
##              95% CI : (0.7981, 0.963)
##      No Information Rate : 0.541
##      P-Value [Acc > NIR] : 1.252e-09
##
##              Kappa : 0.8009
##
##      Mcnemar's Test P-Value : 0.6831
##
##              Sensitivity : 0.9394
##              Specificity : 0.8571
##              Pos Pred Value : 0.8857
##              Neg Pred Value : 0.9231
##              Prevalence : 0.5410
##              Detection Rate : 0.5082
##      Detection Prevalence : 0.5738
##              Balanced Accuracy : 0.8983
##
##      '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      31         6
```

```
## unhealthy      2      22
##
##              Accuracy : 0.8689
##              95% CI : (0.7578, 0.9416)
##      No Information Rate : 0.541
##      P-Value [Acc > NIR] : 5.049e-08
##
##              Kappa : 0.733
##
## McNemar's Test P-Value : 0.2888
##
##      Sensitivity : 0.9394
##      Specificity : 0.7857
##      Pos Pred Value : 0.8378
##      Neg Pred Value : 0.9167
##      Prevalence : 0.5410
##      Detection Rate : 0.5082
##      Detection Prevalence : 0.6066
##      Balanced Accuracy : 0.8626
##
##      '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      28      9
## unhealthy     5     19
##
##              Accuracy : 0.7705
##              95% CI : (0.645, 0.8685)
##      No Information Rate : 0.541
##      P-Value [Acc > NIR] : 0.0001784
##
##              Kappa : 0.5328
##
## McNemar's Test P-Value : 0.4226781
##
##      Sensitivity : 0.8485
##      Specificity : 0.6786
##      Pos Pred Value : 0.7568
```



```
##          Neg Pred Value : 0.7917
##          Prevalence : 0.5410
##          Detection Rate : 0.4590
##          Detection Prevalence : 0.6066
##          Balanced Accuracy : 0.7635
##
##          'Positive' Class : healthy
##
```

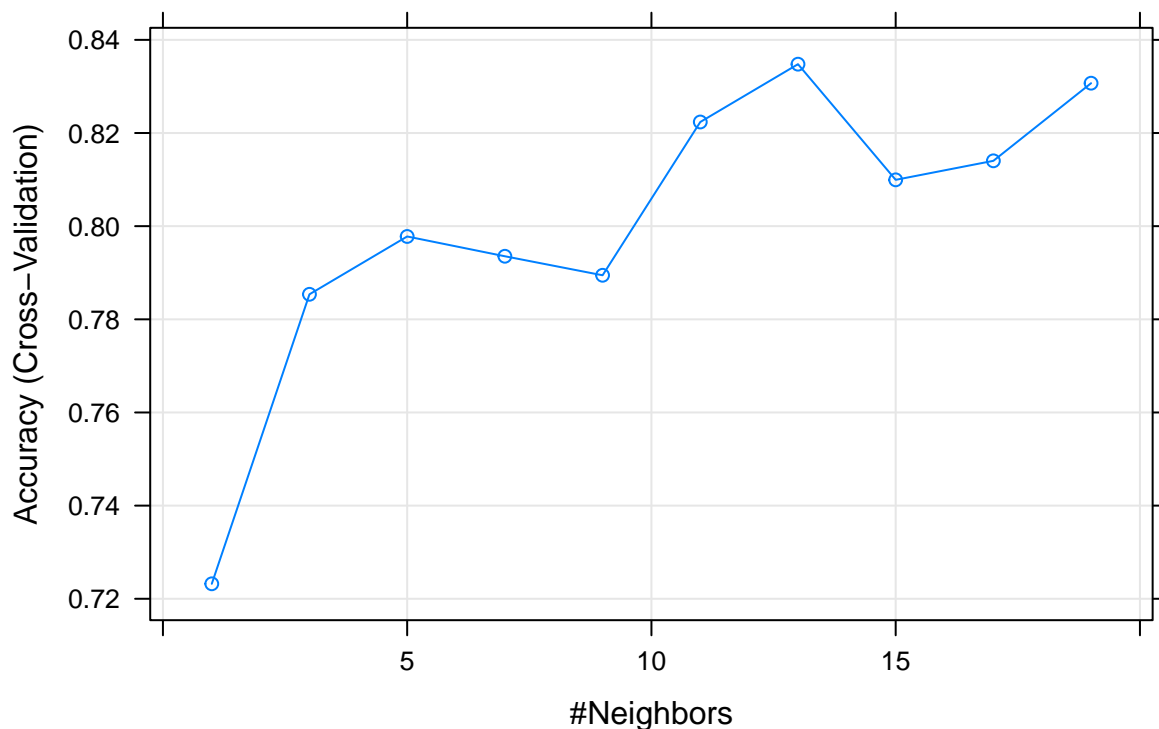
4. KNN -

```
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      30         6
## unhealthy     3        22
```

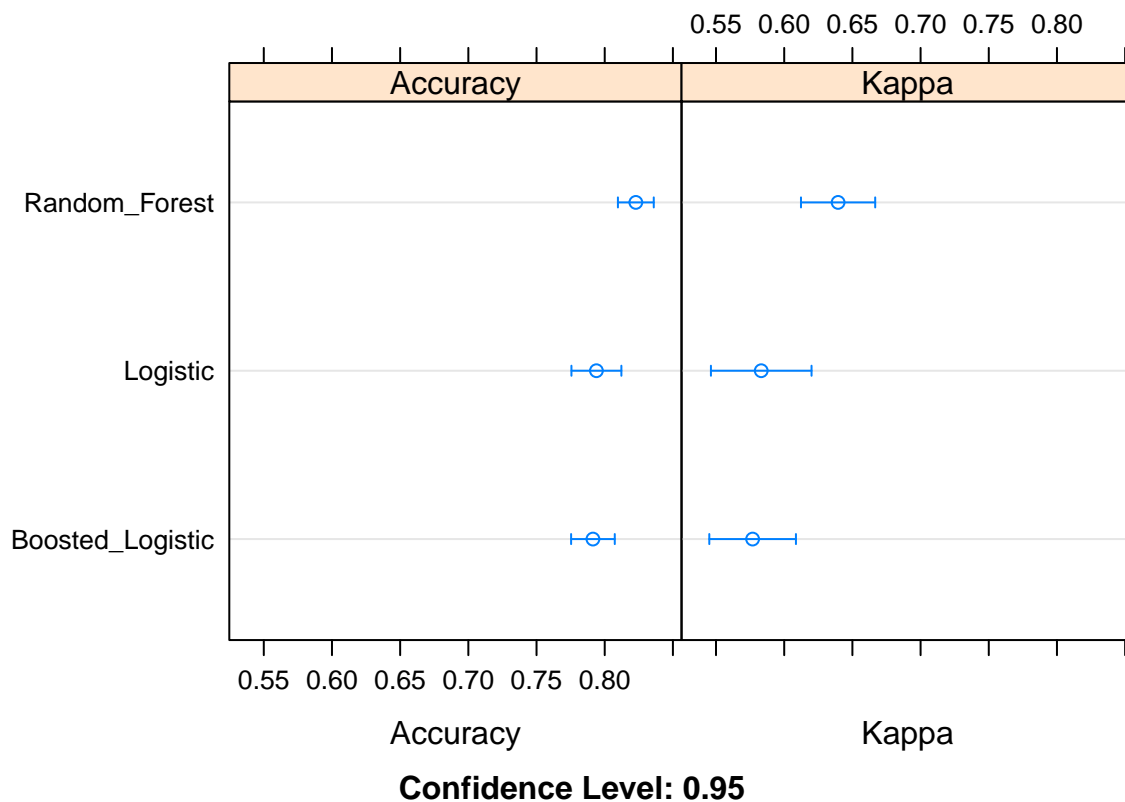
```
##
##           Accuracy : 0.8525
##           95% CI   : (0.7383, 0.9302)
##      No Information Rate : 0.541
##      P-Value [Acc > NIR] : 2.6e-07
##
##           Kappa : 0.7005
##
##  McNemar's Test P-Value : 0.505
##
##           Sensitivity : 0.9091
##           Specificity : 0.7857
##      Pos Pred Value : 0.8333
##      Neg Pred Value : 0.8800
##           Prevalence : 0.5410
##      Detection Rate : 0.4918
##      Detection Prevalence : 0.5902
##      Balanced Accuracy : 0.8474
##
##      '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      0.7045455 0.7634409 0.7956989 0.7939150 0.8260870 0.8588235
## Random_Forest  0.7710843 0.7954545 0.8279570 0.8228161 0.8461538 0.8817204
## Boosted_Logistic 0.7058824 0.7555556 0.8068182 0.7913639 0.8235294 0.8484848
##           NA's
## Logistic      0
## Random_Forest  0
## Boosted_Logistic 0
##
## Kappa
##           Min.   1st Qu.   Median     Mean   3rd Qu.   Max.
## Logistic      0.3982115 0.5270270 0.5845286 0.5831608 0.6454721 0.7171381
## Random_Forest  0.5231327 0.5838150 0.6528231 0.6395206 0.6875000 0.7593507
## Boosted_Logistic 0.4115204 0.5111111 0.6058733 0.5768440 0.6455696 0.6966292
##           NA's
## Logistic      0
## Random_Forest  0
## Boosted_Logistic 0
```

```
dotplot(results)
```



```
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"]])

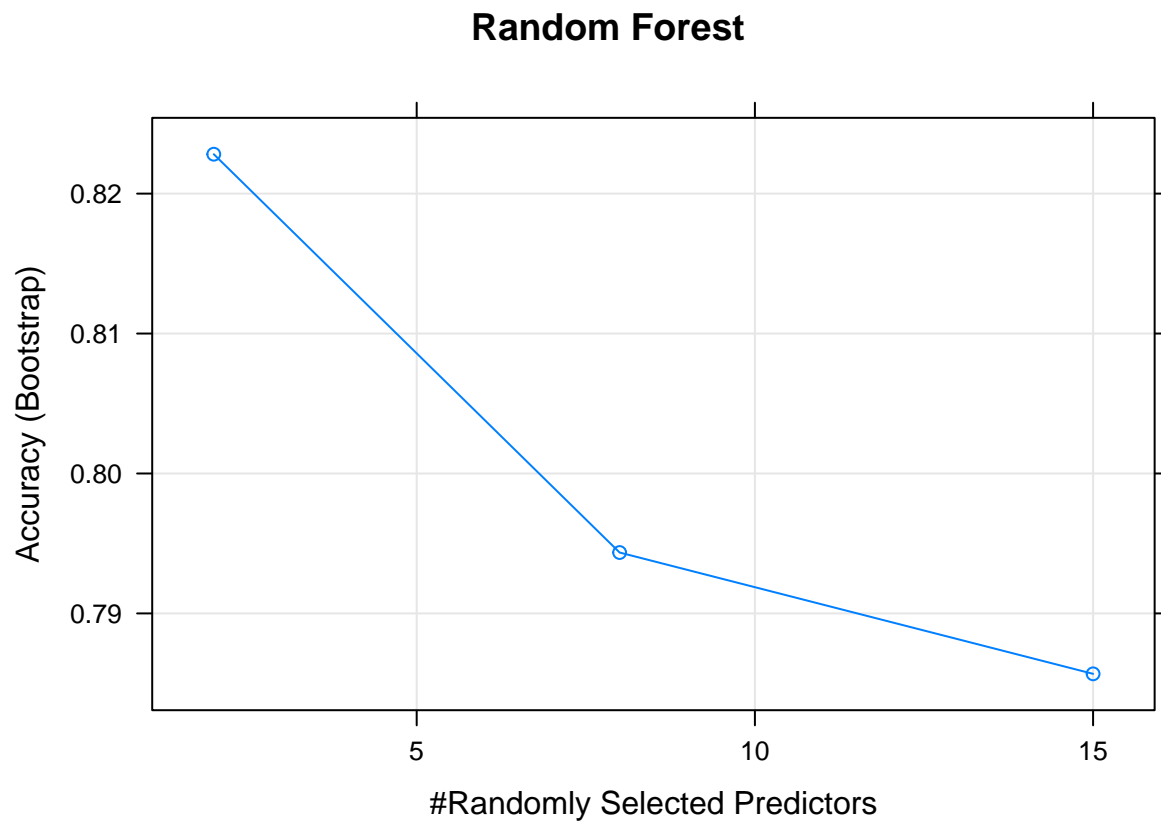
Sensitivity <- 100*rbind(cf_log[["byClass"]][["Sensitivity"]],
  cf_rf[["byClass"]][["Sensitivity"]],
  cf_blog[["byClass"]][["Sensitivity"]],
  cf_knn[["byClass"]][["Sensitivity"]])

pf_result <- t(data.frame(Accuracy, Specificity, Sensitivity))
colnames(pf_result) <- c("Log", "RF", "LogitB", "KNN")
pf_result <- as.matrix(pf_result)
```

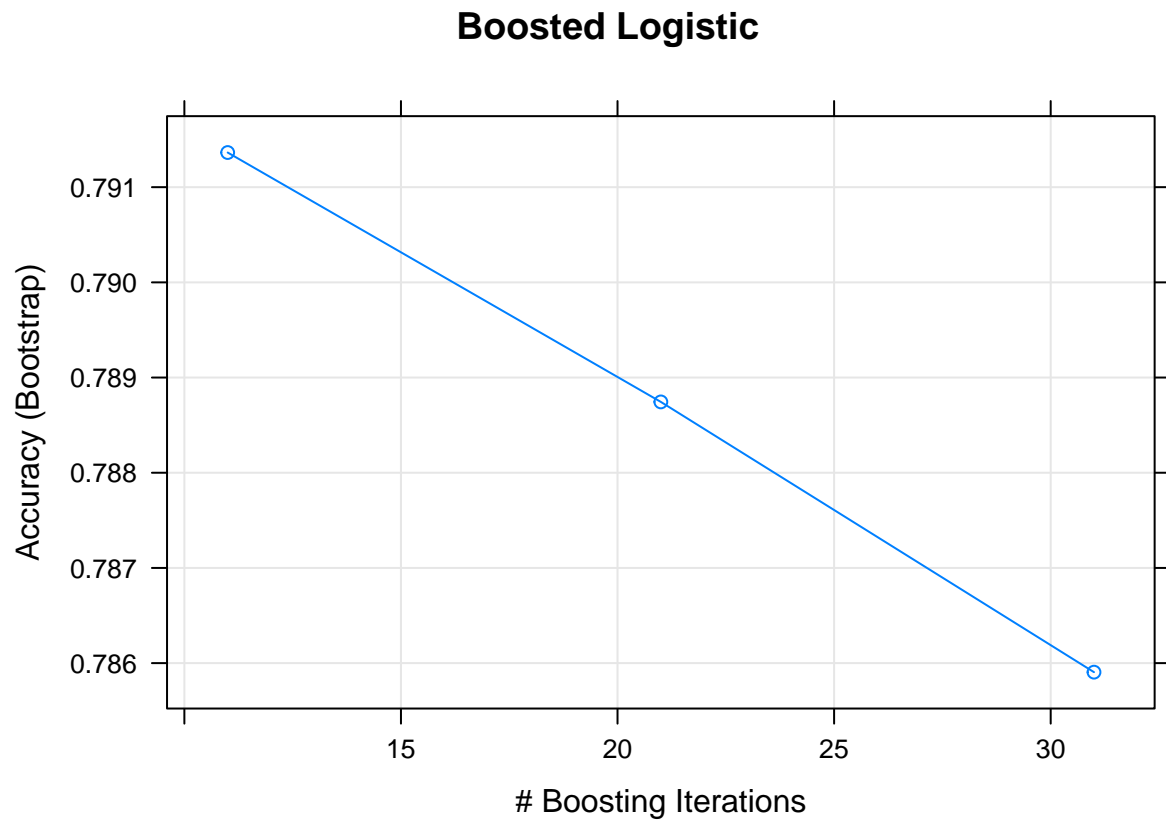
```
pf_result
```

##		Log	RF	LogitB	KNN
## Accuracy		90.16393	86.88525	90.16393	85.24590
## Specificity		85.71429	78.57143	85.71429	78.57143
## Sensitivity		93.93939	93.93939	93.93939	90.90909

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

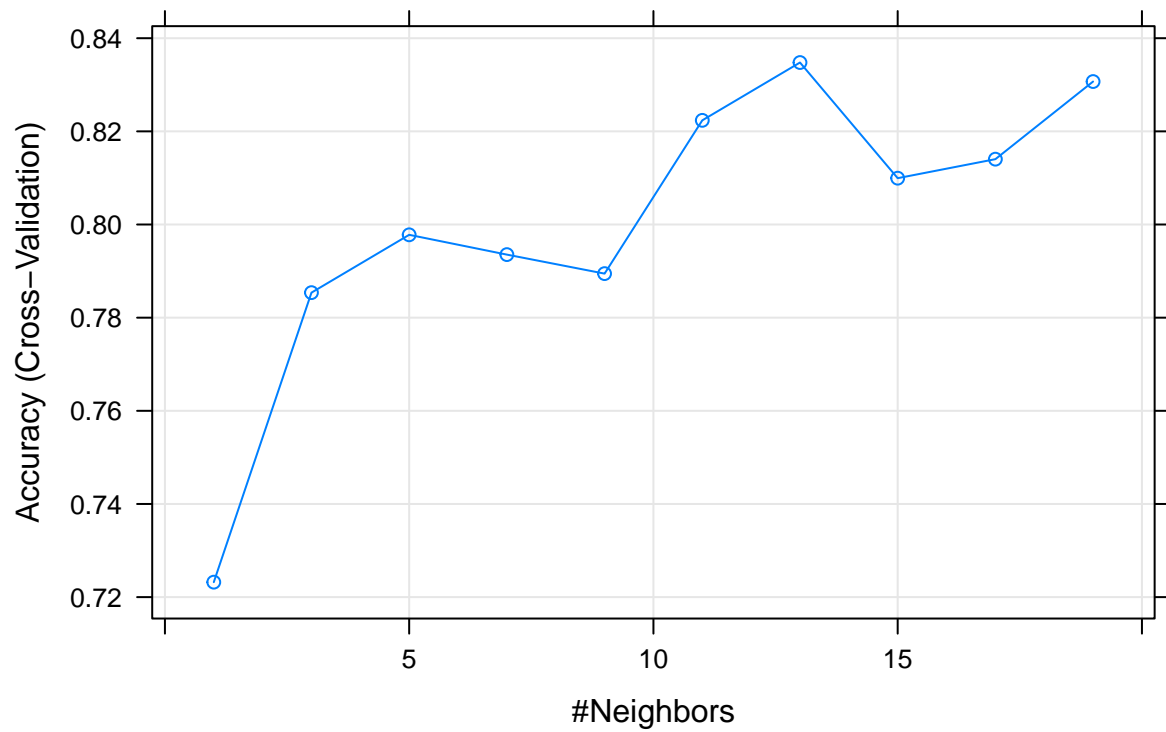


```
plot(blog_fit, main = "Boosted Logistic")
```



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

K-nearest neighbour

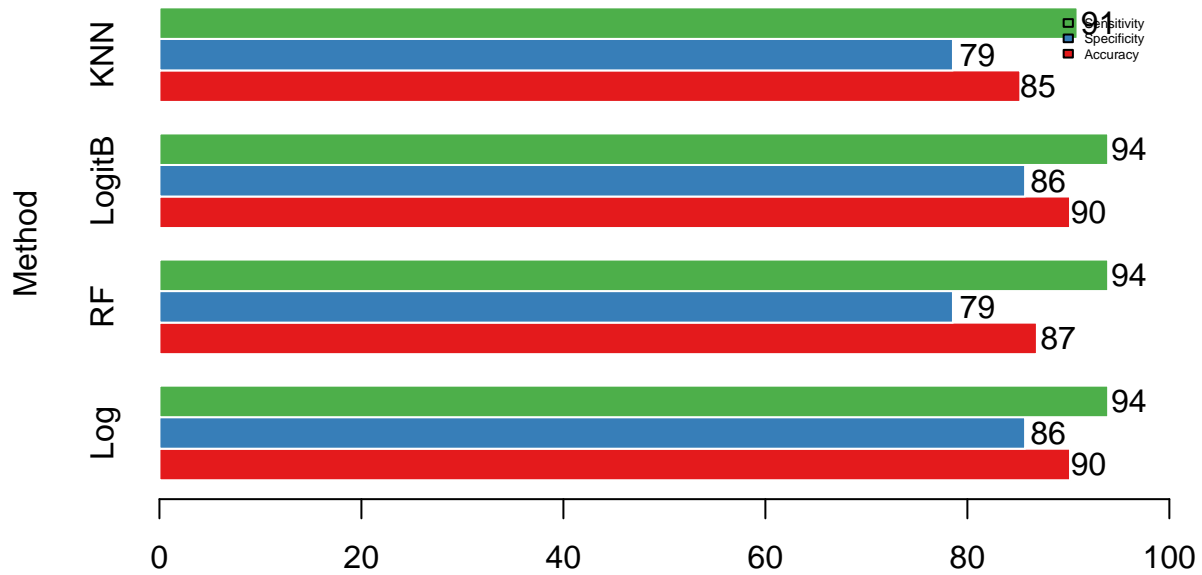


```
y <- barplot(pf_result, beside = TRUE, horiz = TRUE,
             col=brewer.pal(3,"Set1"),border="white",
             legend.text = c("Accuracy", "Specificity", "Sensitivity"),
             args.legend = list(bty = "n", cex = 0.4), xlim=c(0,100),
             main = "Performance Chart", ylab =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

x <- round(pf_result)

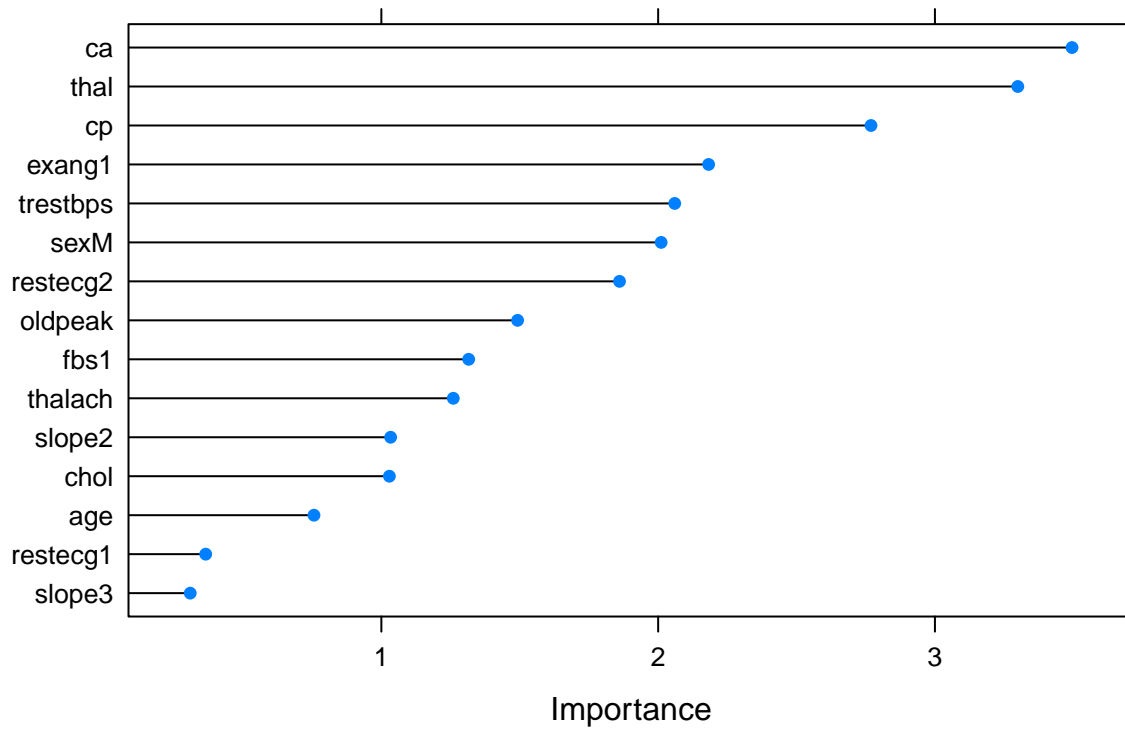
text(x+2,y,labels=as.character(x))
```

Performance Chart



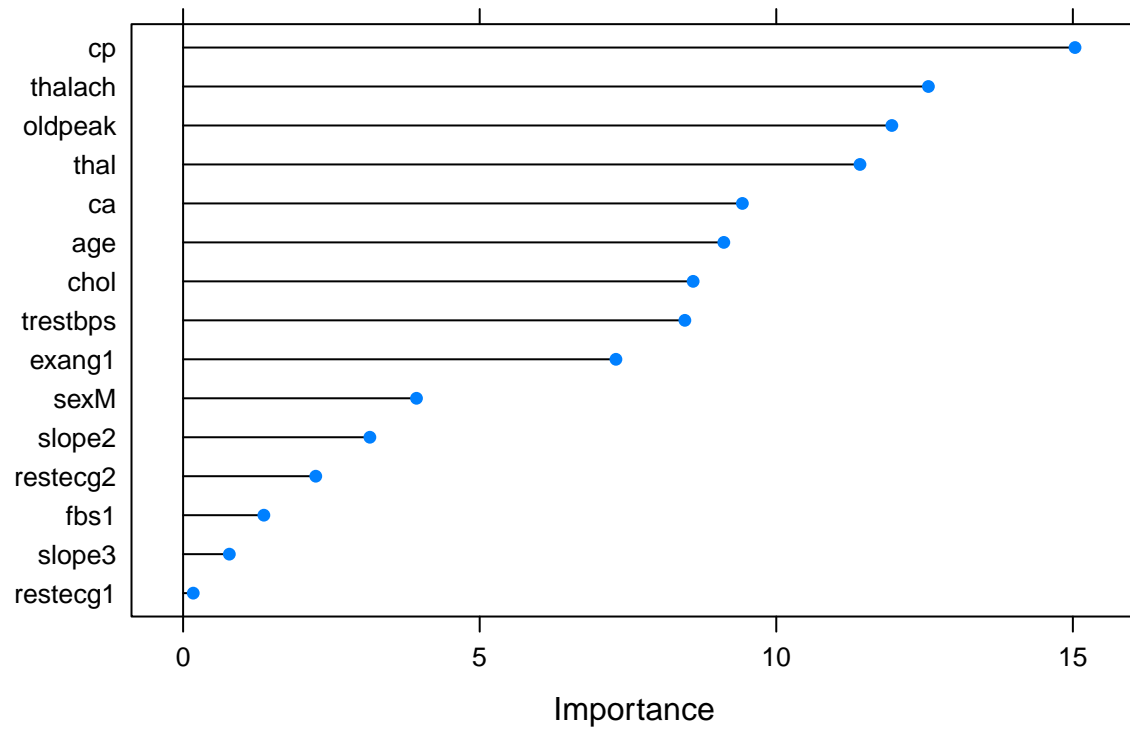
```
## Feature extraction -  
feat_log <- varImp(log_fit, scale = FALSE)  
feat_rf <- varImp(rf_fit, scale = FALSE)  
feat_blog <- varImp(blog_fit, scale = FALSE)  
feat_knn <- varImp(knn_fit, scale = FALSE)  
  
plot(feat_log, main = "Logistic regression: features")
```

Logistic regression: features



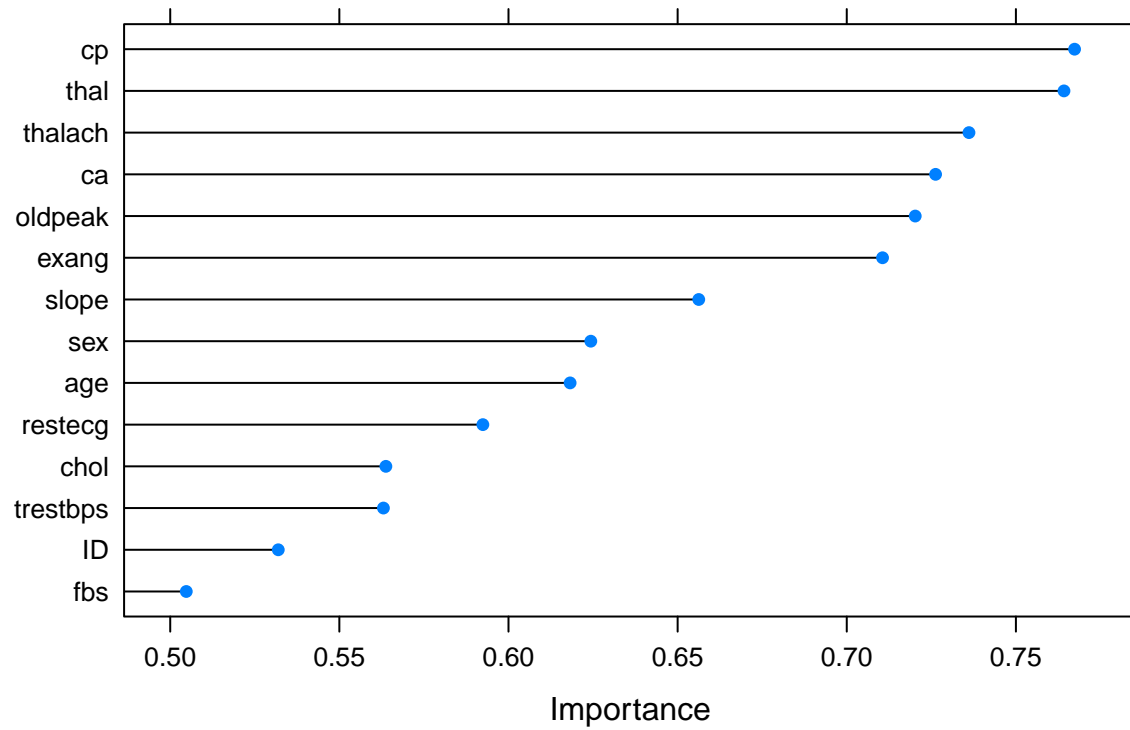
```
plot(feats_rf, main = "Random forest: features")
```


Random forest: features



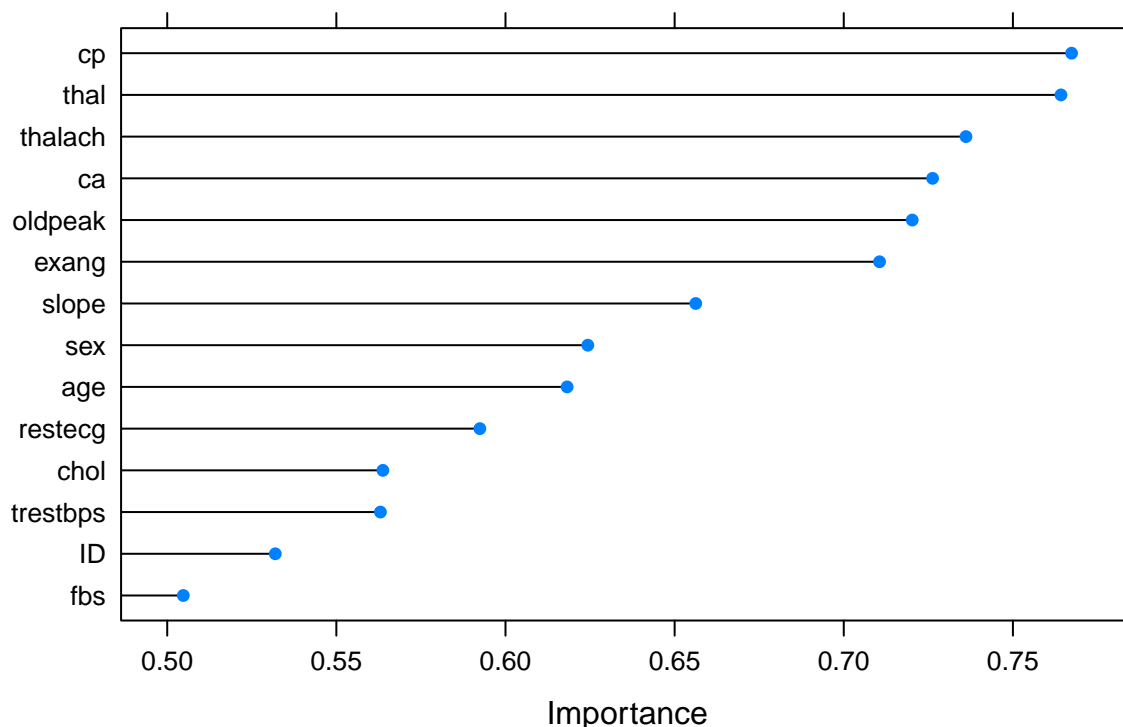
```
plot(feats_blog, main = "Boosted Logistic regression: features")
```

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)
```

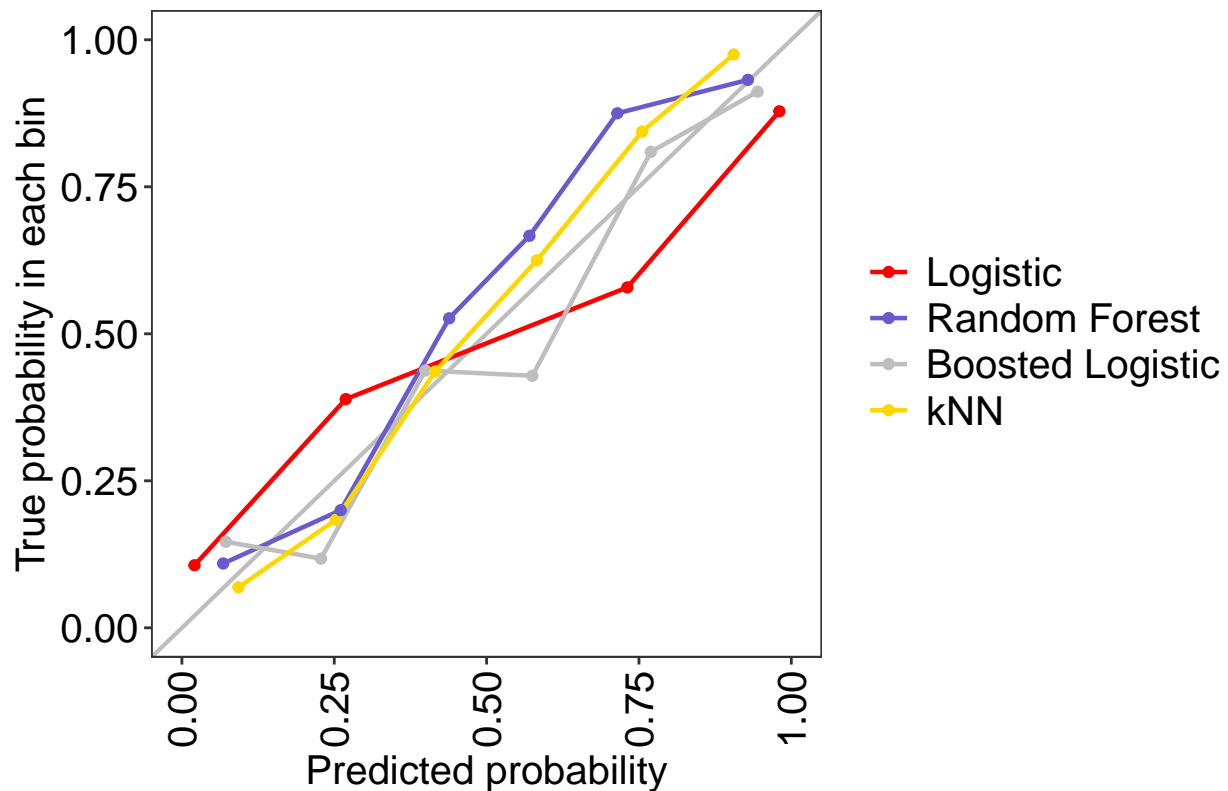
```
## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" was not
```

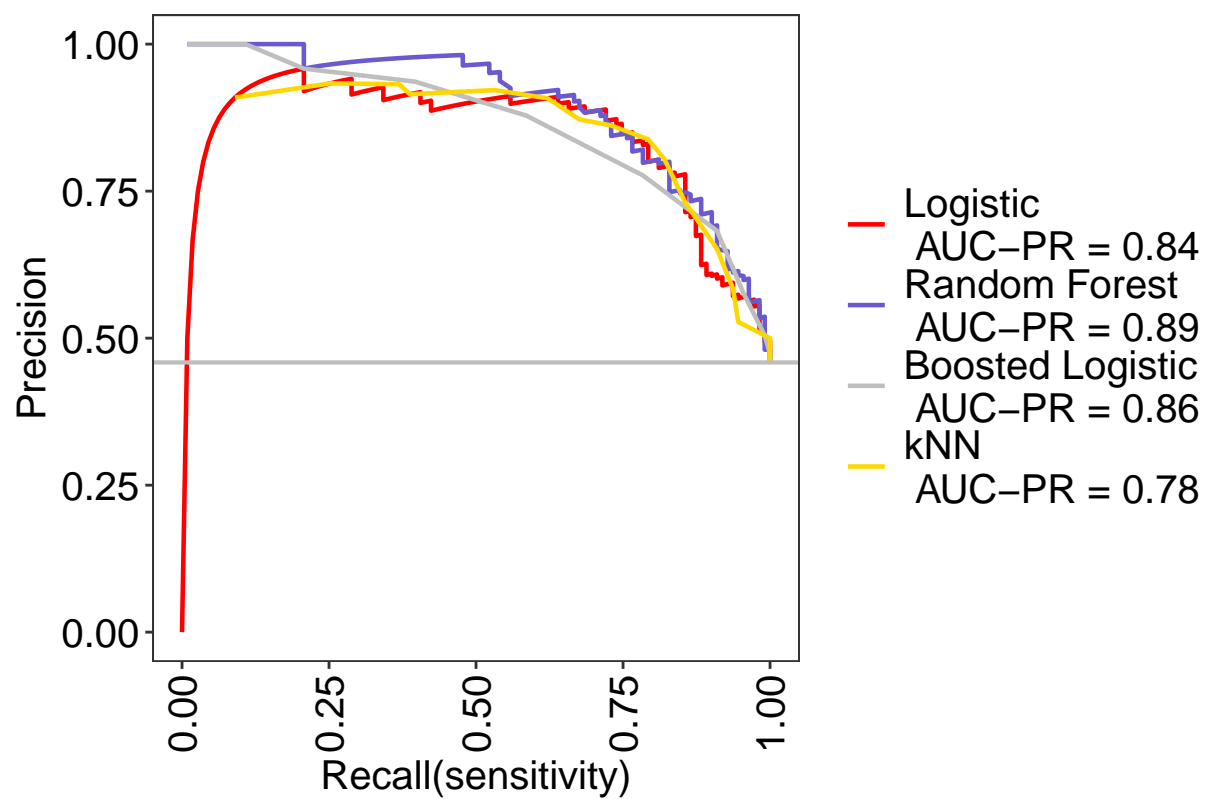
```
## in the result set. ROC will be used instead.
```

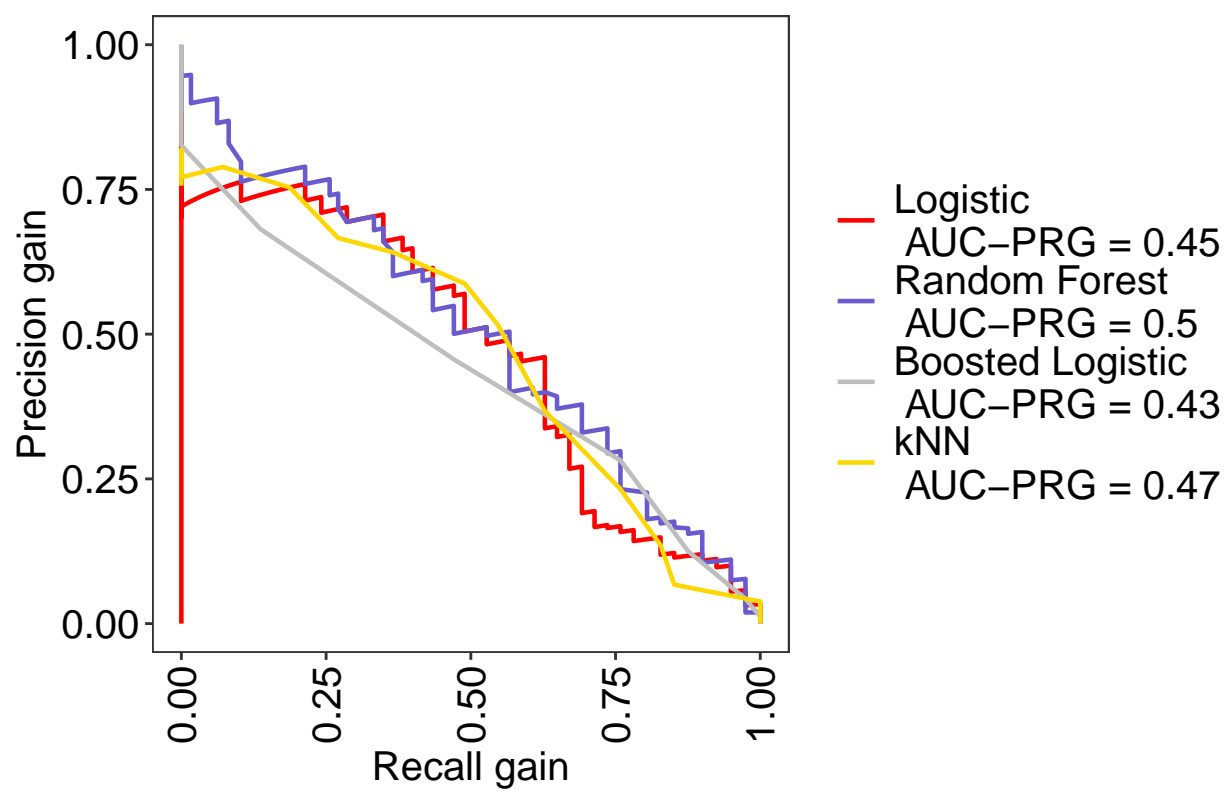
```
knn <- train(goal ~. -ID , data = train_data,
  method = "knn", preProcess = c("center","scale"),
  trControl = cont , tuneGrid = expand.grid(k = seq(1, 20, 2)))
```

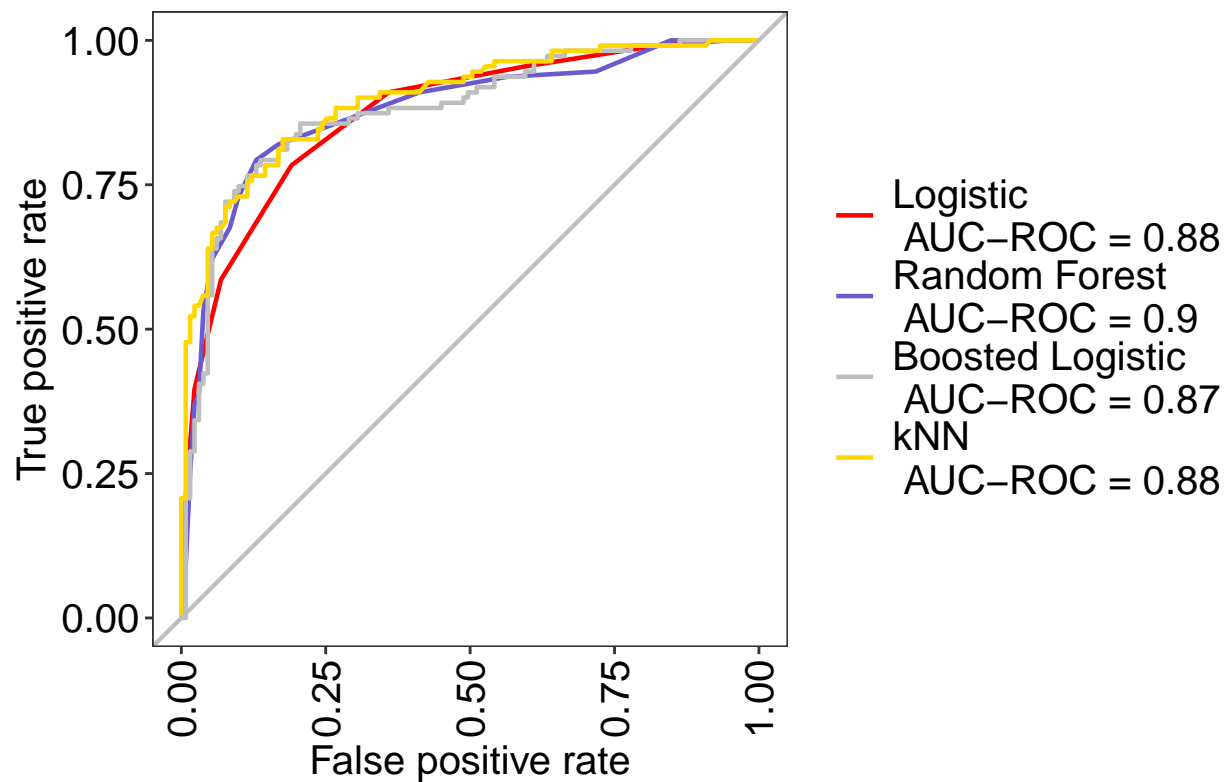
```
## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" was not
## in the result set. ROC will be used instead.
```

```
#####
##                               AUC-ROC                               ##
#####
metric <- evalm(list(log,rf,blog,knn),
  gnames=c('Logistic','Random Forest',
    'Boosted Logistic', 'kNN'),
  rlinethick=0.8,fsiz=15, silent = TRUE)
```









```
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      0.87
## Random Forest 0.90
## Boosted Logit 0.88
## kNN           0.89
```

```
##                               EOL                               -
```