Project DS 1: Exploratory data analysis, and classifier training to predict condition using Cleveland Heart disease dataset.

Abhinav Mishra

2022-11-16

```
##            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              -
```

```
if (!require("pacman", quietly = TRUE))
  install.packages("pacman")

pacman::p_load(tidyverse,
               skimr,
               ggvis,
               caret,
               MLeval,
               mice,
               RColorBrewer,
               ggplot2,
               ggpubr,
               GGally)
```

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

```
processedWithHeader_cleveland <-
  read_csv(
    "~/Documents/Freie/IFA/WiSe 22-23/Data Science/Week 2/processedWithHeader.cleveland.data",
    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)
```

```r
heart_data$slope <- as.factor(heart_data$slope)
#heart_data$cp <- as.factor(heart_data$cp)
```

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

```r
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
```

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

```r
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" ...
```

```r
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
```

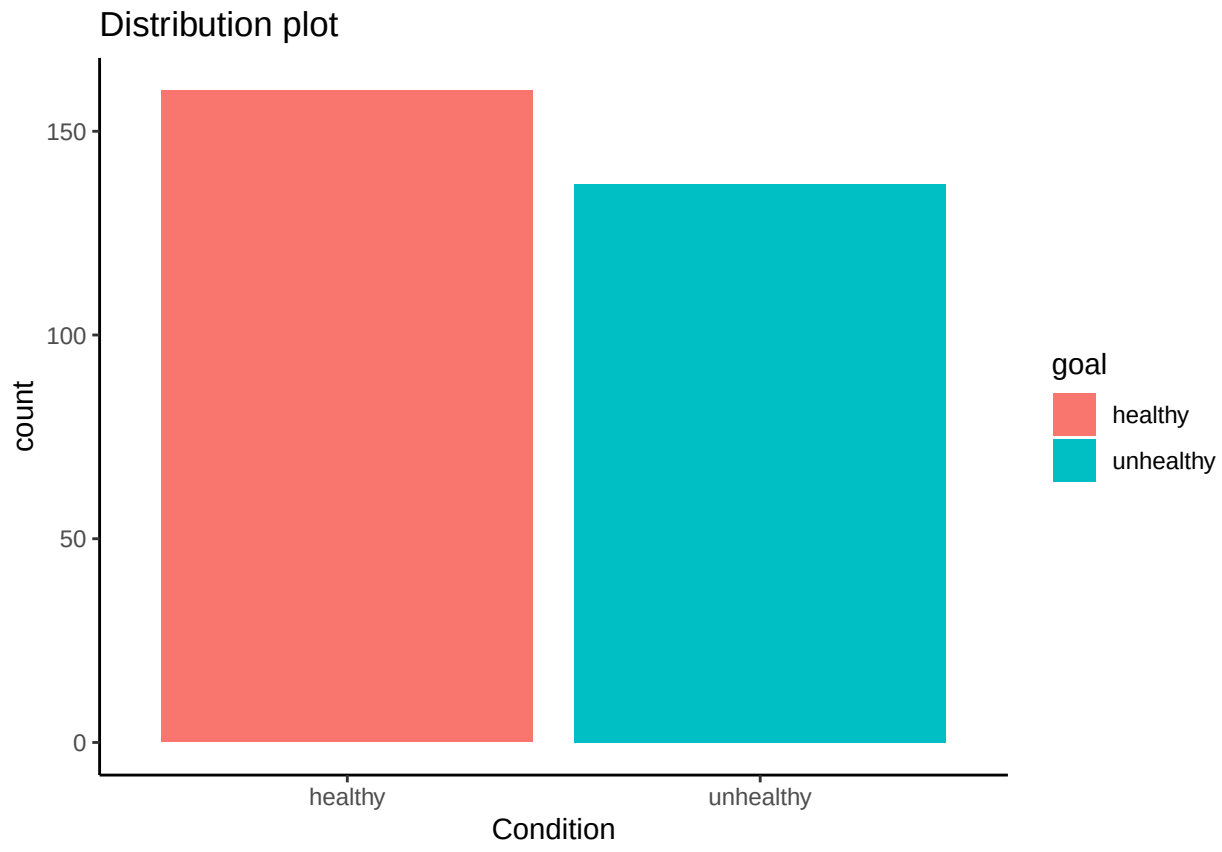
```
##  fbs     restecg    thalach      exang      oldpeak      slope
##  0:254   0:147   Min.   : 71.0   0:200   Min.   :0.000   1:139
##  1: 43   1:  4   1st Qu.:133.0   1: 97   1st Qu.:0.000   2:137
##          2:146   Median :153.0           Median :0.800   3: 21
##                  Mean   :149.6           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")
```




```
##########################################################################
##                 Run this chunk after decommenting it                 ##
```

```
##               to get pair plots, grouped by sex and goal.              ##
##   It was creating problems in markdown generation so I commented it.  ##
################################################################################

# ggscatmat(heart_data,
#           columns =
#             c('cp', 'ca', 'thal', 'thalach'),
#           color = "sex") + theme_classic() + labs(
#             x = "",
#             y = "",
#             title = "Pairplot",
#             subtitle = "Group: Sex"
#           )

# ggscatmat(heart_data,
#           columns =
#             c('cp', 'ca', 'thal', 'thalach'),
#           color = "goal") + theme_classic() + labs(
#             x = "",
#             y = "",
#             title = "Pairplot",
#             subtitle = "Group: Disease 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
```

```
##                          EDA: Heatmap                          -
```

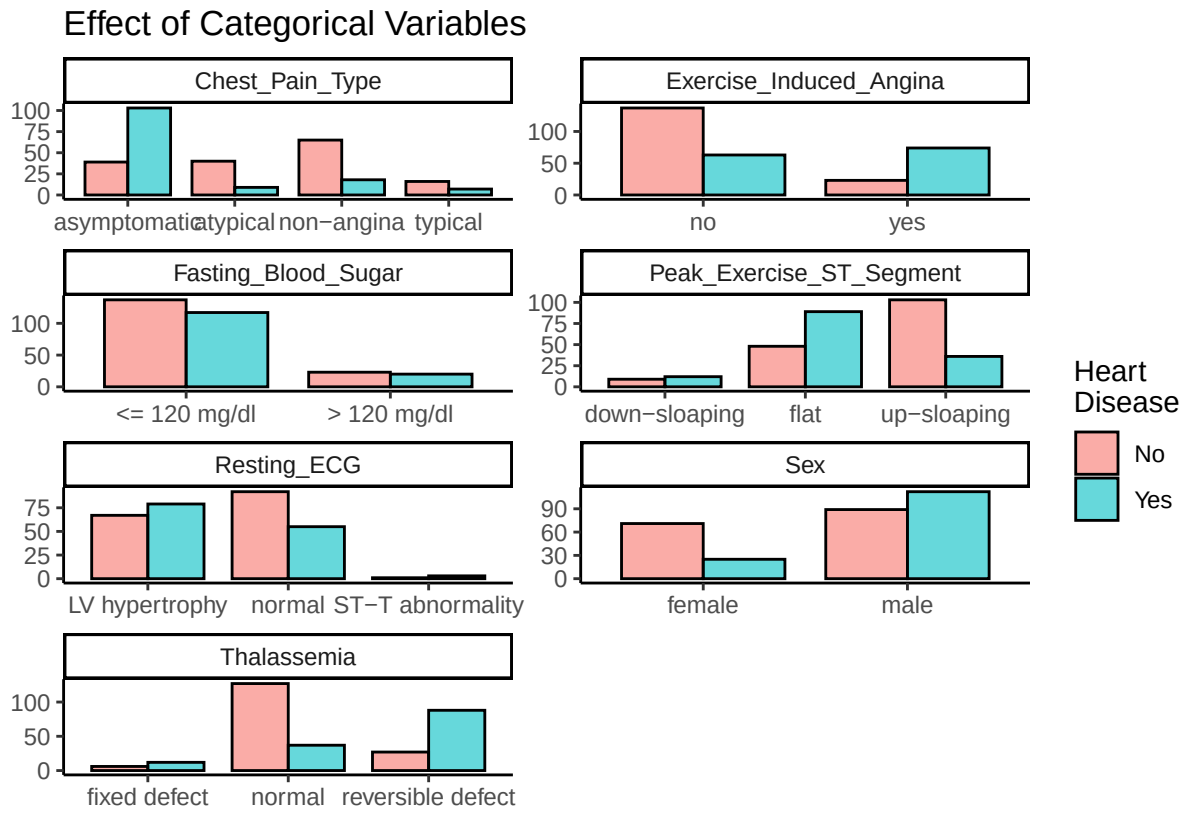
```
names <- c(
  "Age",
  "Sex",
  "Chest_Pain_Type",
  "Resting_Blood_Pressure",
  "Serum_Cholesterol",
  "Fasting_Blood_Sugar",
  "Resting_ECG",
  "Max_Heart_Rate_Achieved",
  "Exercise_Induced_Angina",
  "ST_Depression_Exercise",
  "Peak_Exercise_ST_Segment",
  "Num_Major_Vessels_Flouro",
  "Thalassemia",
  "Diagnosis_Heart_Disease"
)
```

```r
colnames(data) <- names

data <- data %>% drop_na() %>%
  mutate_at(
    c(
      "Resting_ECG",
      "Fasting_Blood_Sugar",
      "Sex",
      "Diagnosis_Heart_Disease",
      "Exercise_Induced_Angina",
      "Peak_Exercise_ST_Segment",
      "Chest_Pain_Type"
    ),
    as_factor
  ) %>%
  mutate(Num_Major_Vessels_Flouro = as.numeric(Num_Major_Vessels_Flouro)) %>%
  mutate(Diagnosis_Heart_Disease = fct_lump(Diagnosis_Heart_Disease, other_level = "1")) %>%
  filter(Thalassemia != "?") %>%
  select(
    Age,
    Resting_Blood_Pressure,
    Serum_Cholesterol,
    Max_Heart_Rate_Achieved,
    ST_Depression_Exercise,
    Num_Major_Vessels_Flouro,
    everything()
  )

data.long <- data  %>%
  select(
    Sex,
    Chest_Pain_Type,
    Fasting_Blood_Sugar,
    Resting_ECG,
    Exercise_Induced_Angina,
    Peak_Exercise_ST_Segment,
    Thalassemia,
    Diagnosis_Heart_Disease
  ) %>%
  mutate(
    Sex = recode_factor(Sex, `0` = "female",
                        `1` = "male"),
    Chest_Pain_Type = recode_factor(
      Chest_Pain_Type,
      `1` = "typical",
      `2` = "atypical",
      `3` = "non-angina",
      `4` = "asymptomatic"
    ),
    Fasting_Blood_Sugar = recode_factor(Fasting_Blood_Sugar, `0` = "<= 120 mg/dl",
                                        `1` = "> 120 mg/dl"),
    Resting_ECG = recode_factor(
      Resting_ECG,
```

```
      `0` = "normal",
      `1` = "ST-T abnormality",
      `2` = "LV hypertrophy"
    ),
    Exercise_Induced_Angina = recode_factor(Exercise_Induced_Angina, `0` = "no",
                                            `1` = "yes"),
    Peak_Exercise_ST_Segment = recode_factor(
      Peak_Exercise_ST_Segment,
      `1` = "up-sloaping",
      `2` = "flat",
      `3` = "down-sloaping"
    ),
    Thalassemia = recode_factor(
      Thalassemia,
      `3` = "normal",
      `6` = "fixed defect",
      `7` = "reversible defect"
    )
  ) %>%
  gather(key = "key", value = "value",-Diagnosis_Heart_Disease)
```

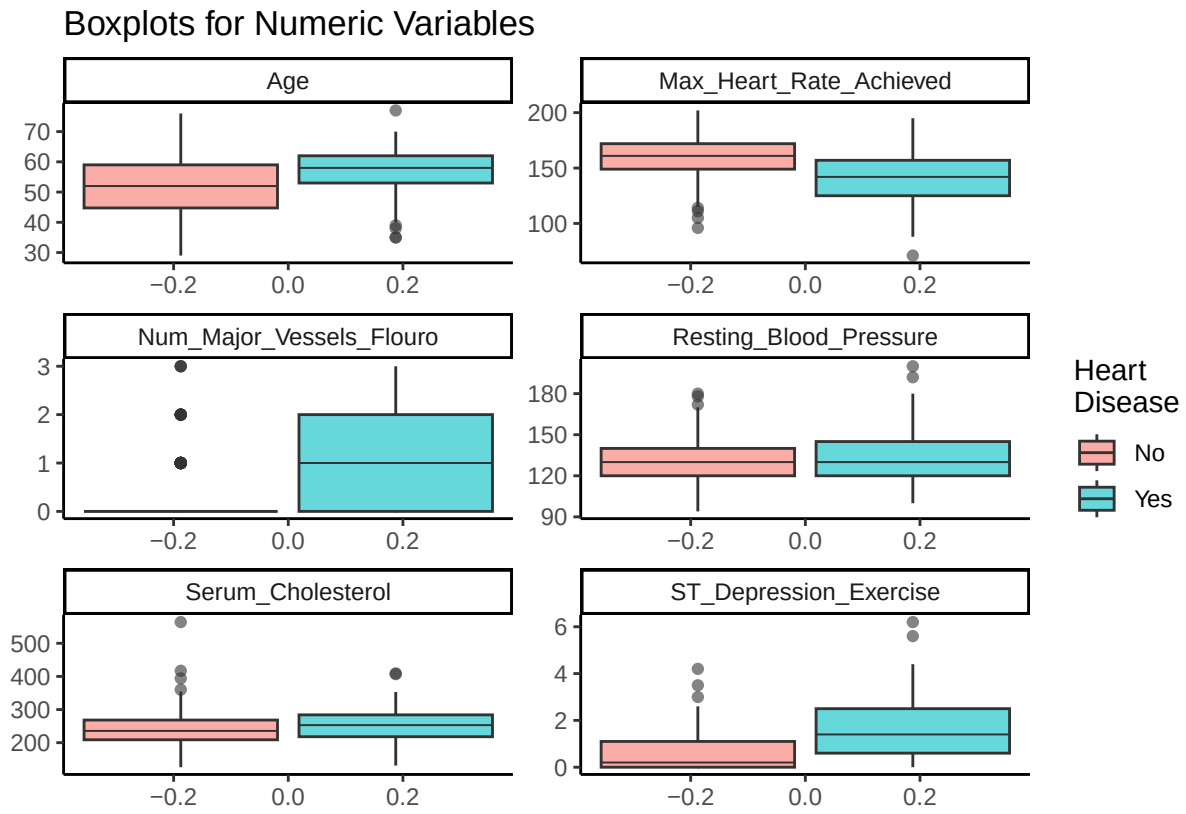
```
## Warning: attributes are not identical across measure variables;
## they will be dropped
```

```
data.long %>%
  ggplot(aes(value)) +
  geom_bar(
    aes(x        = value,
        fill     = Diagnosis_Heart_Disease),
    alpha    = .6,
    position = "dodge",
    color    = "black",
    width    = .8
  ) +
  labs(x = "",
       y = "",
       title = "Effect of Categorical Variables") +
  theme(axis.text.y  = element_blank(),
        axis.ticks.y = element_blank()) +
  facet_wrap( ~ key, scales = "free", nrow = 4) +
  scale_fill_manual(
    values = c("#F8766D", "#00BFC4"),
    name   = "Heart\nDisease",
    labels = c("No", "Yes")
  ) + theme_classic()
```



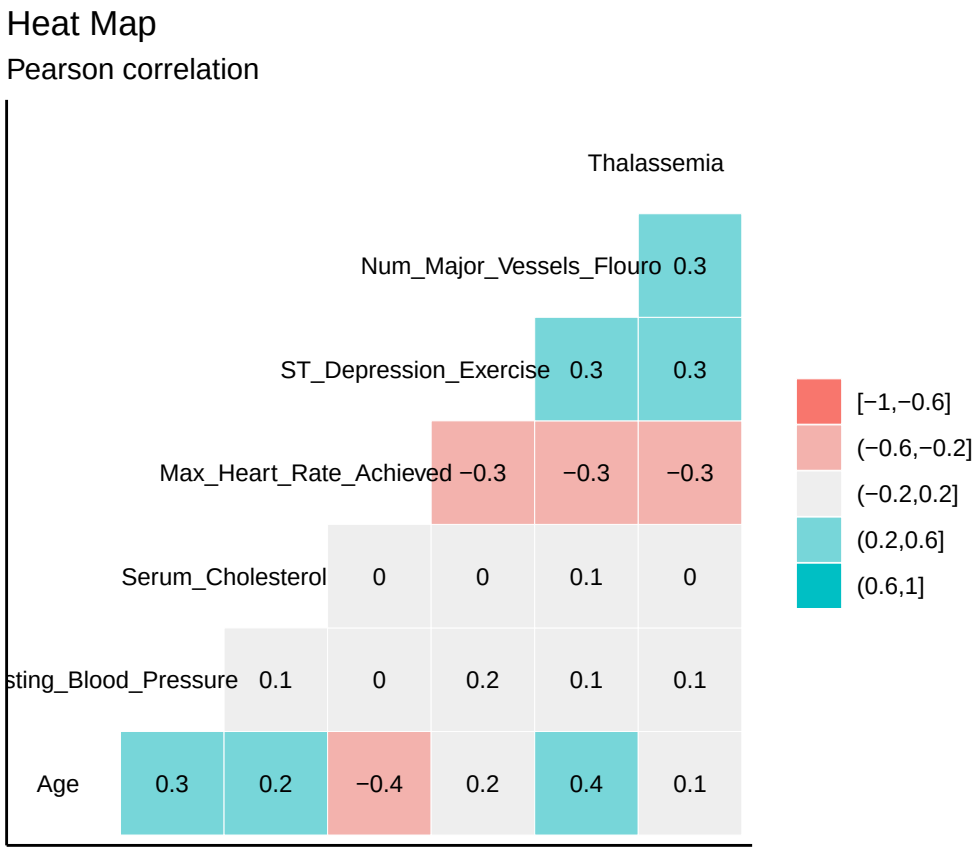

```
data.long.cont <- data  %>%
  select(
    Age,
    Resting_Blood_Pressure,
    Serum_Cholesterol,
    Max_Heart_Rate_Achieved,
    ST_Depression_Exercise,
    Num_Major_Vessels_Flouro,
    Diagnosis_Heart_Disease
  ) %>%
  gather(key   = "key",
         value = "value",-Diagnosis_Heart_Disease)

data.long.cont %>%
  ggplot(aes(y = value)) +
  geom_boxplot(aes(fill = Diagnosis_Heart_Disease),
               alpha  = .6,
               fatten = .7) +
  labs(x = "",
       y = "",
       title = "Boxplots for Numeric Variables") +
  scale_fill_manual(
    values = c("#F8766D", "#00BFC4"),
    name   = "Heart\nDisease",
    labels = c("No", "Yes")
  ) +
```

```
  theme(axis.text.x  = element_blank(),
        axis.ticks.x = element_blank()) +
  facet_wrap( ~ key,
              scales = "free",
              ncol   = 2) + theme_classic()
```



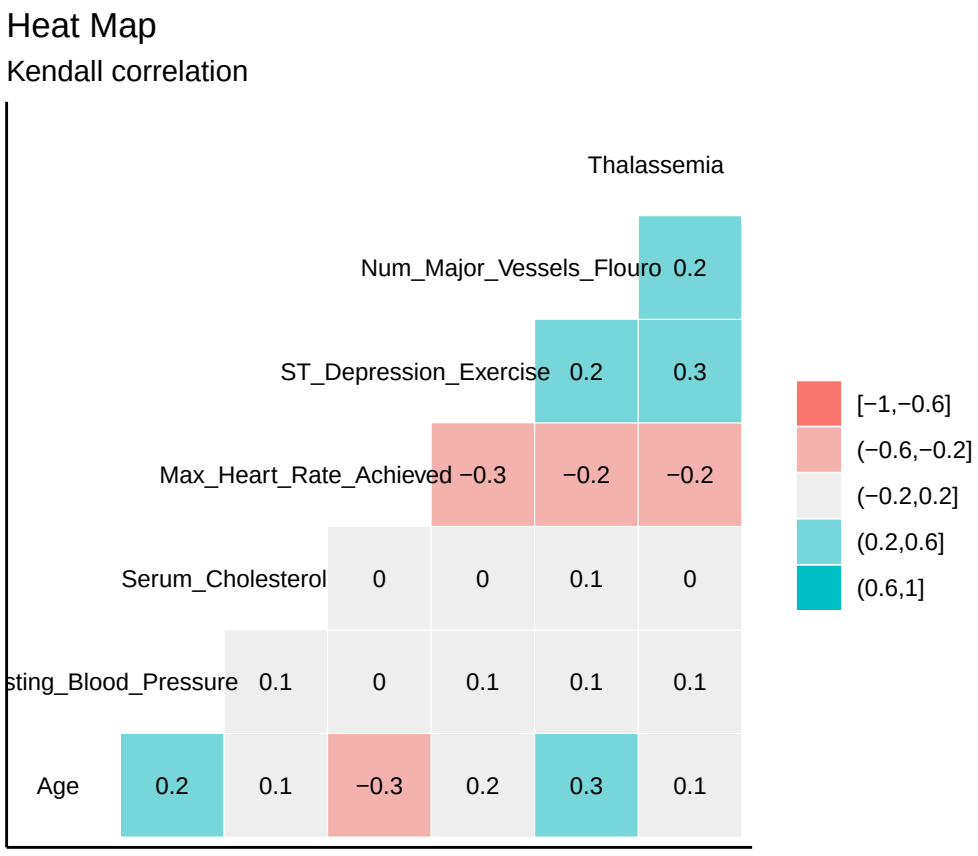

```
data %>% ggcorr(
  high       = "#00BFC4",
  low        = "#F8766D",
  label      = TRUE,
  hjust      = .75,
  size       = 3,
  label_size = 3,
  nbreaks    = 5
) +
  labs(title = "Heat Map", subtitle = "Pearson correlation") +
  theme_classic()
```

```
## Warning in ggcorr(., high = "#00BFC4", low = "#F8766D", label = TRUE, hjust
## = 0.75, : data in column(s) 'Sex', 'Chest_Pain_Type', 'Fasting_Blood_Sugar',
## 'Resting_ECG', 'Exercise_Induced_Angina', 'Peak_Exercise_ST_Segment',
## 'Diagnosis_Heart_Disease' are not numeric and were ignored
```




```
data %>% ggcorr(
  method     = c("pairwise", "kendall"),
  high       = "#00BFC4",
  low        = "#F8766D",
  label      = TRUE,
  hjust      = .75,
  size       = 3,
  label_size = 3,
  nbreaks    = 5
) +
  labs(title = "Heat Map", subtitle = "Kendall correlation") +
  theme_classic()
```

```
## Warning in ggcorr(., method = c("pairwise", "kendall"), high = "#00BFC4", : data
## in column(s) 'Sex', 'Chest_Pain_Type', 'Fasting_Blood_Sugar', 'Resting_ECG',
## 'Exercise_Induced_Angina', 'Peak_Exercise_ST_Segment', 'Diagnosis_Heart_Disease'
## are not numeric and were ignored
```




```
##                          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        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
##
```

```
## 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        26         5
##   unhealthy       6        23
##
##                Accuracy : 0.8167
##                  95% CI : (0.6956, 0.9048)
##     No Information Rate : 0.5333
##     P-Value [Acc > NIR] : 4.344e-06
##
##                   Kappa : 0.6325
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8125
```

```
##             Specificity : 0.8214
##          Pos Pred Value : 0.8387
##          Neg Pred Value : 0.7931
##              Prevalence : 0.5333
##          Detection Rate : 0.4333
##    Detection Prevalence : 0.5167
##       Balanced Accuracy : 0.8170
##
##        '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.              -
```

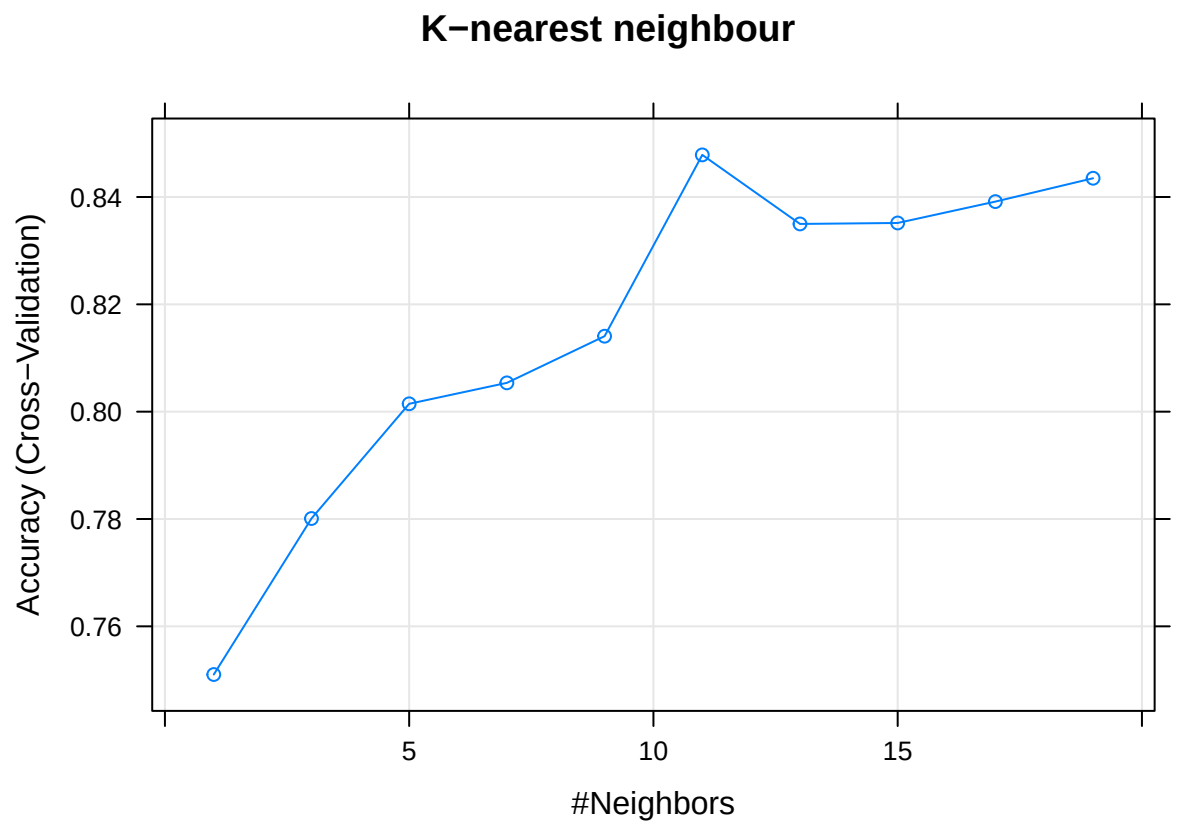
```
set.seed(112)
blog_fit <- train(goal ~ . - ID,
                  data = train_data,
                  method = "LogitBoost")
blog_pred <- predict(blog_fit, test_data)
confusionMatrix(blog_pred, test_data$goal)
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  healthy unhealthy
##   healthy        24         4
##   unhealthy       8        24
##
##                Accuracy : 0.8
##                  95% CI : (0.6767, 0.8922)
##     No Information Rate : 0.5333
##     P-Value [Acc > NIR] : 1.609e-05
##
##                   Kappa : 0.6018
##
##  Mcnemar's Test P-Value : 0.3865
##
##             Sensitivity : 0.7500
##             Specificity : 0.8571
##          Pos Pred Value : 0.8571
##          Neg Pred Value : 0.7500
##              Prevalence : 0.5333
##          Detection Rate : 0.4000
##    Detection Prevalence : 0.4667
##       Balanced Accuracy : 0.8036
##
##        '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)
```

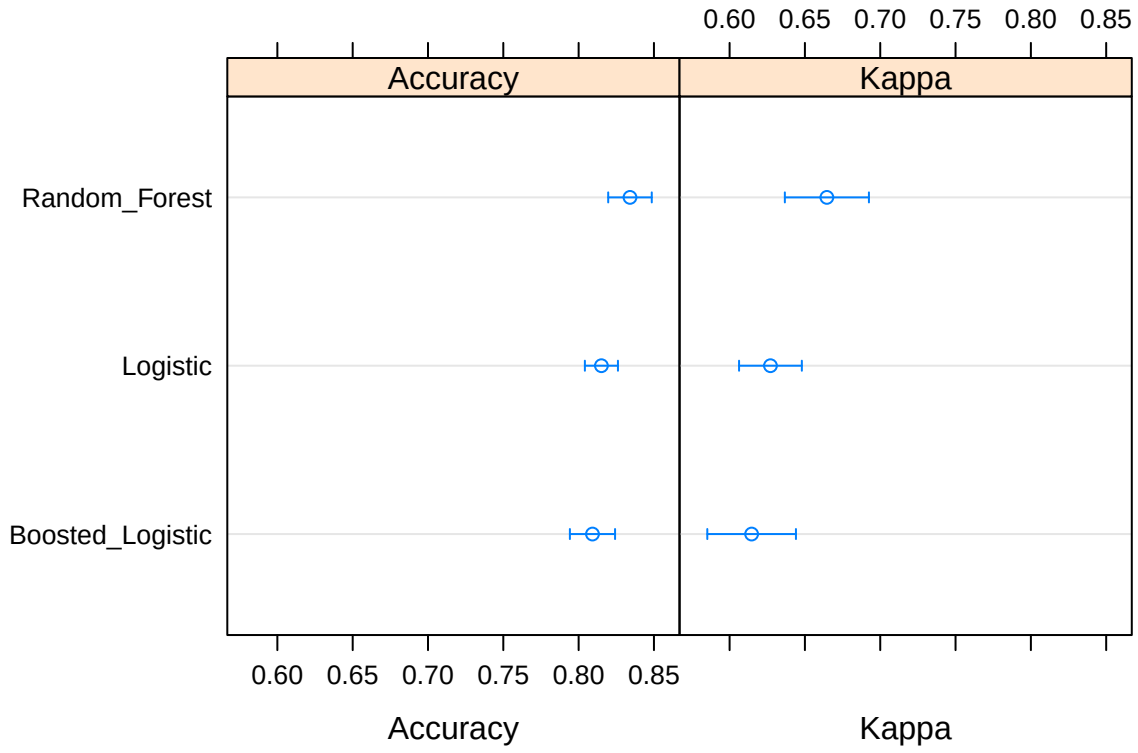
```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  healthy unhealthy
##   healthy        27         5
##   unhealthy       5        23
##
##                Accuracy : 0.8333
##                  95% CI : (0.7148, 0.9171)
```

```
##     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
##
```

```
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         0.7555556 0.7951807 0.8148148 0.8151206 0.8333333 0.8777778
## Random_Forest    0.7619048 0.8181818 0.8390805 0.8340918 0.8555556 0.8924731
## Boosted_Logistic 0.7530864 0.7804878 0.8076923 0.8091958 0.8292683 0.8817204
##                  NA's
## Logistic            0
## Random_Forest       0
## Boosted_Logistic    0
##
## Kappa
##                      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## Logistic         0.5224313 0.5870647 0.6293436 0.6271195 0.6532206 0.7538538
## Random_Forest    0.5350993 0.6253444 0.6705710 0.6645850 0.7096774 0.7765497
## Boosted_Logistic 0.5035734 0.5591398 0.6128392 0.6146170 0.6517214 0.7579844
##                  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)
```

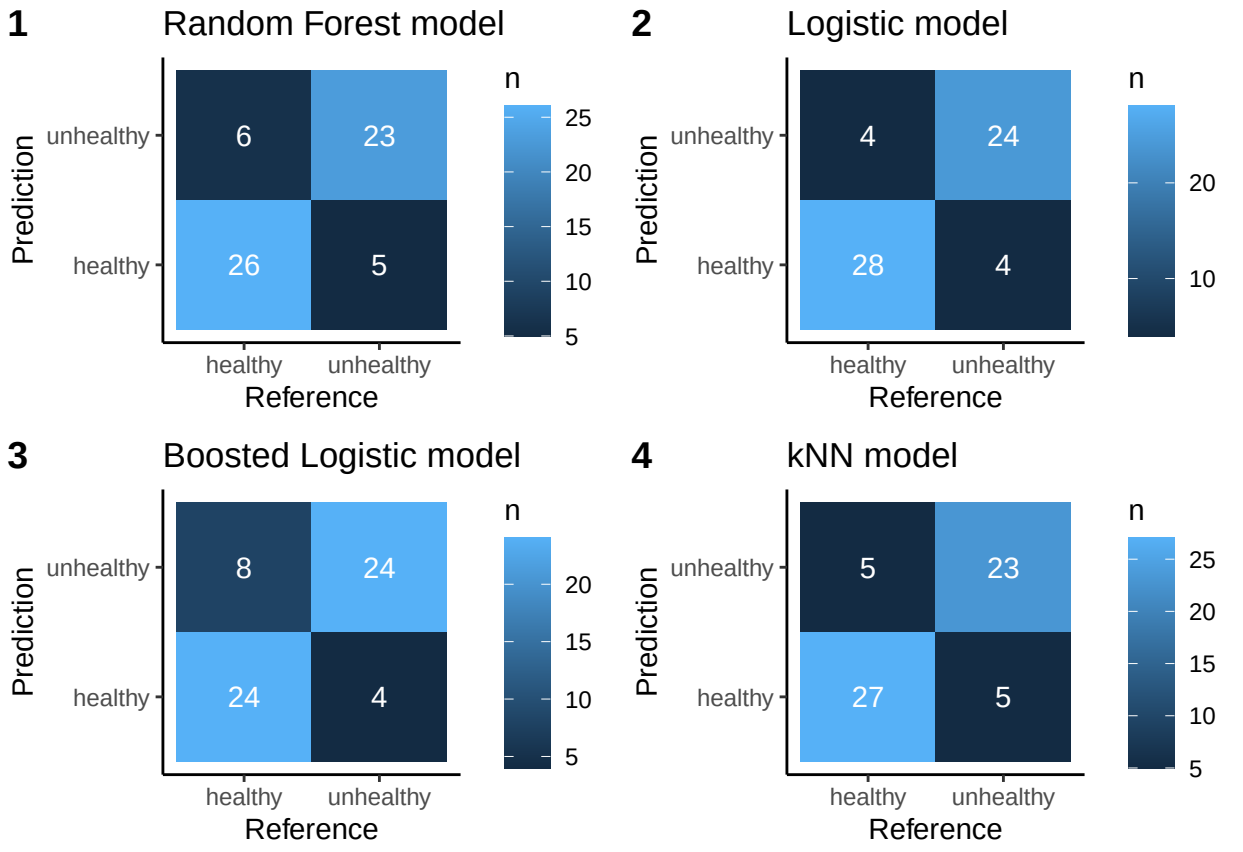
```
##                  Confusion Matrix as Heatmaps                  -
```

```
A <-
  ggplot(as.tibble(as.table((cf_rf))), aes(x = Reference, y = Prediction, fill =
                                             n)) +
  geom_tile() + geom_text(aes(label = n), color = "white") +
  labs(title = "Random Forest model") + theme_classic()
```

```
## Warning: `as.tibble()` was deprecated in tibble 2.0.0.
## i Please use `as_tibble()` instead.
## i The signature and semantics have changed, see `?as_tibble`.
```

```
B <-
  ggplot(as.tibble(as.table((cf_log))), aes(x = Reference, y = Prediction, fill =
                                              n)) +
  geom_tile() + geom_text(aes(label = n), color = "white") +
  labs(title = "Logistic model") + theme_classic()
C <-
  ggplot(as.tibble(as.table((cf_blog))), aes(x = Reference, y = Prediction, fill =
                                               n)) +
```

```
  geom_tile() + geom_text(aes(label = n), color = "white") +
  labs(title = "Boosted Logistic model") + theme_classic()
D <-
  ggplot(as.tibble(as.table((cf_knn))), aes(x = Reference, y = Prediction, fill =
                                              n)) +
  geom_tile() + geom_text(aes(label = n), color = "white") +
  labs(title = "kNN model") + theme_classic()

ggarrange(
  A,
  B,
  C,
  D,
  labels =
    c("1", "2", "3", "4"),
  ncol = 2,
  nrow = 2
)
```



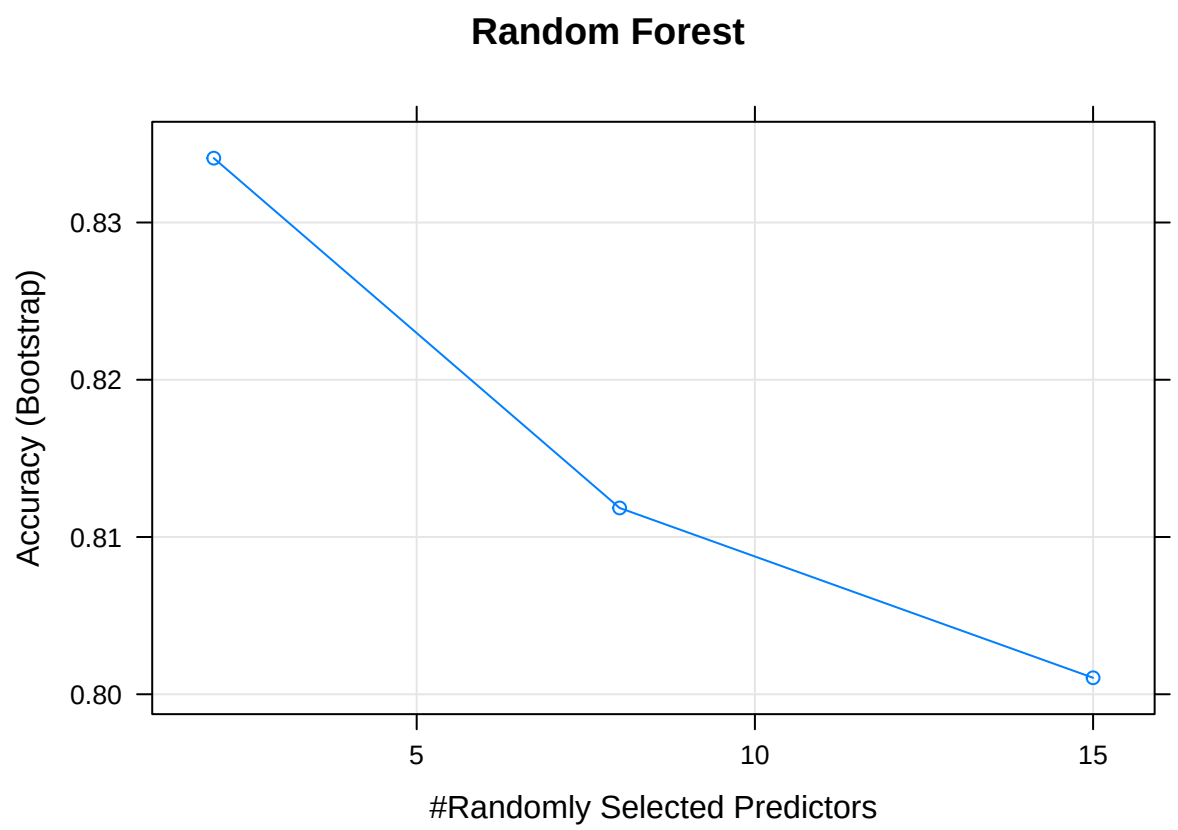

```
##                    Performance Comparison                    -
```

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

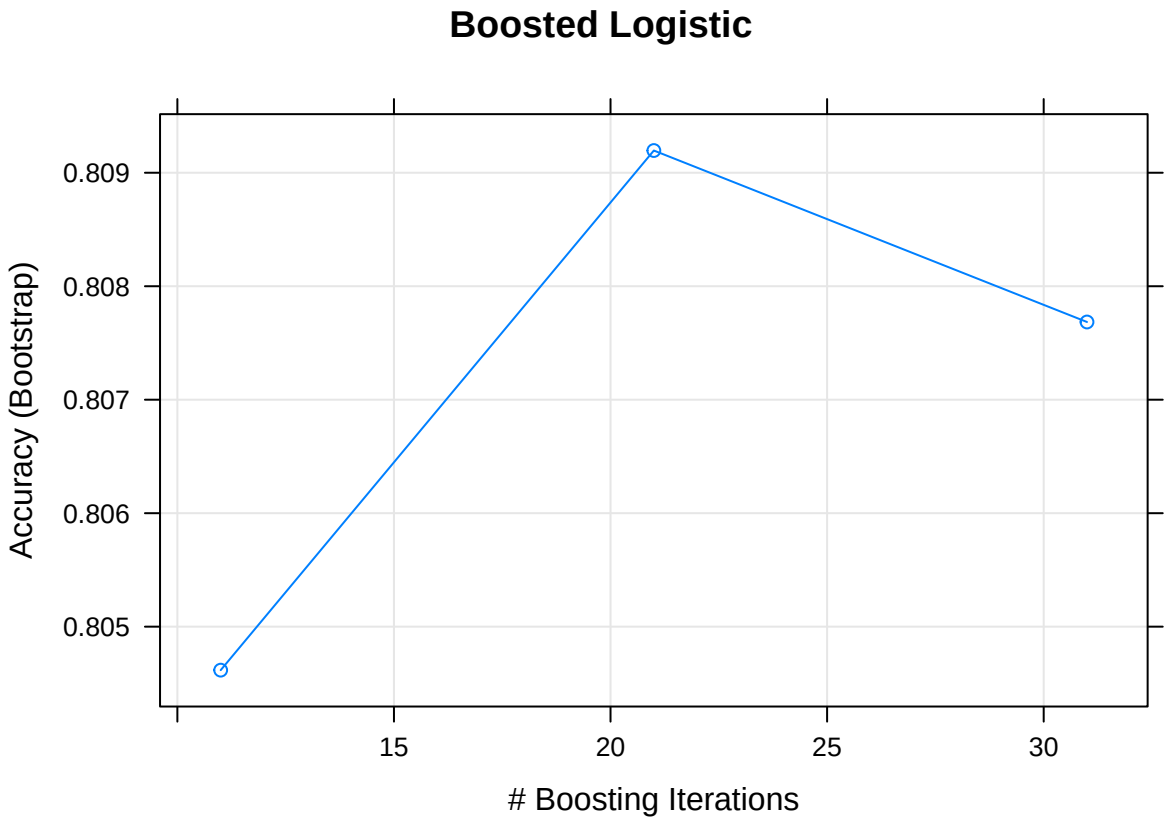
```
Specificity <- 100 * rbind(cf_log[["byClass"]][["Specificity"]],
                           cf_rf[["byClass"]][["Specificity"]],
                           cf_blog[["byClass"]][["Specificity"]],
                           cf_knn[["byClass"]][["Specificity"]])

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 81.66667 80.00000 83.33333
## Specificity 85.71429 82.14286 85.71429 82.14286
## Sensitivity 87.50000 81.25000 75.00000 84.37500
```

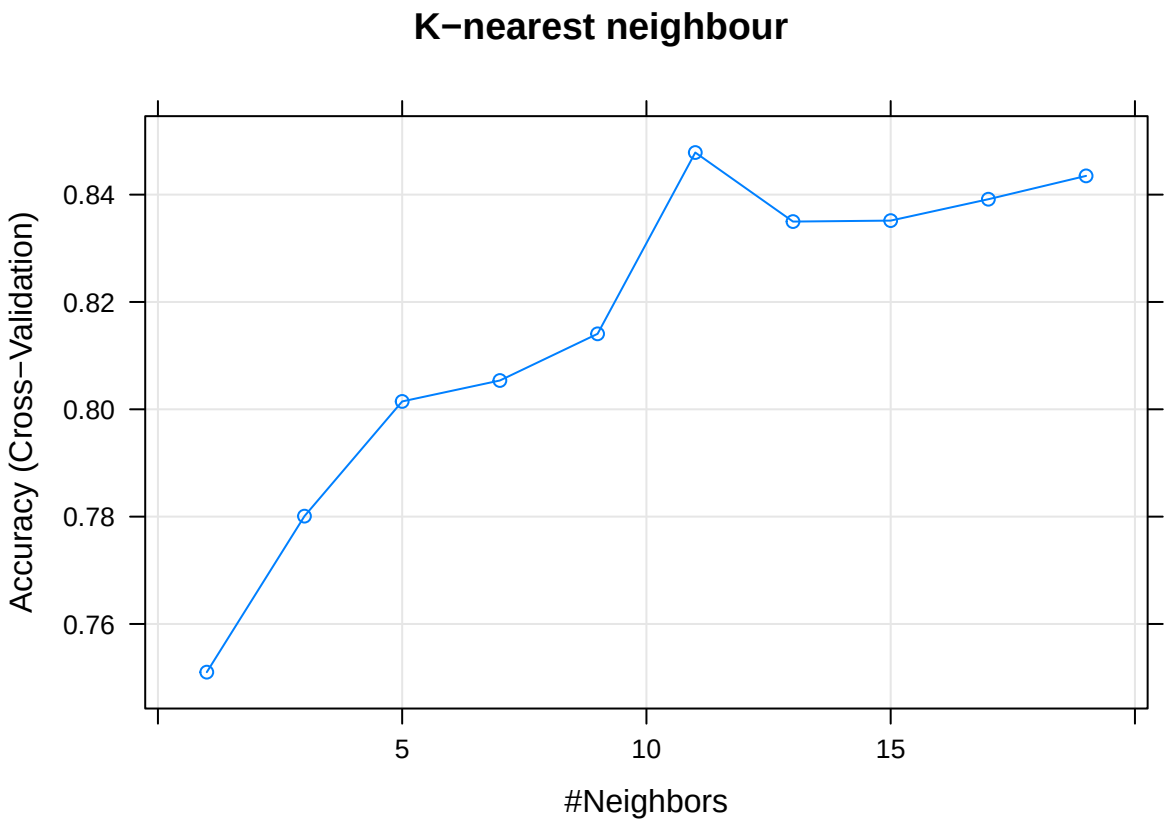
```
plot(rf_fit, main = "Random Forest")
```



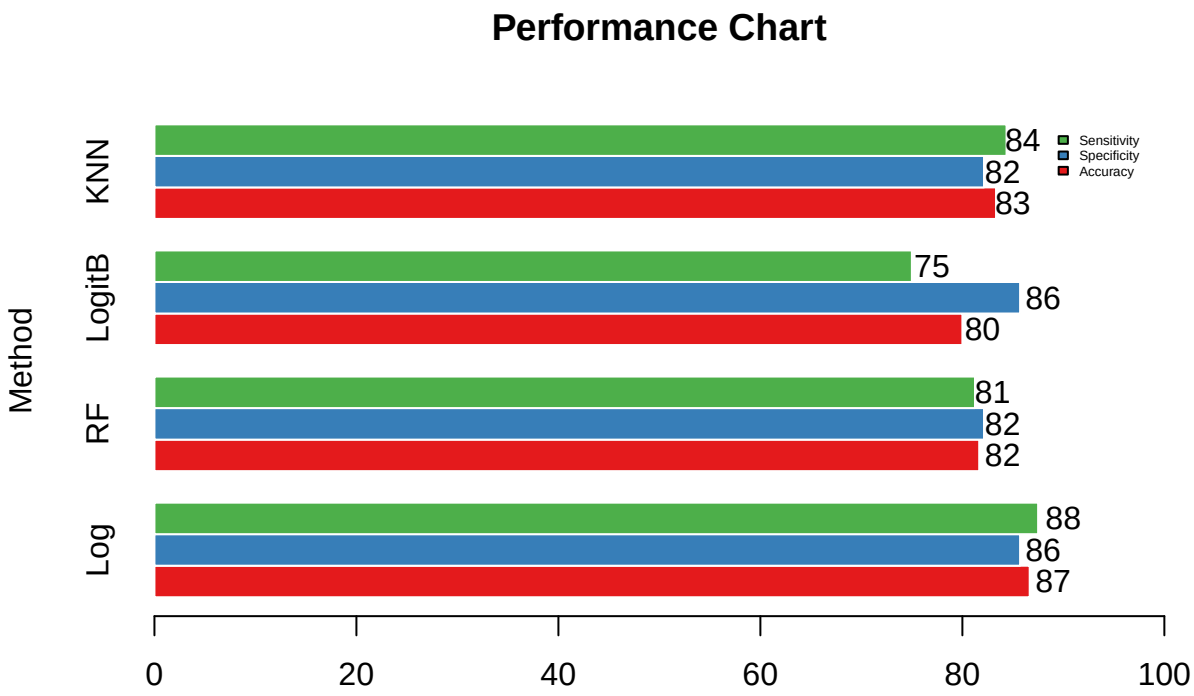

```
plot(blog_fit, main = "Boosted Logistic")
```




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

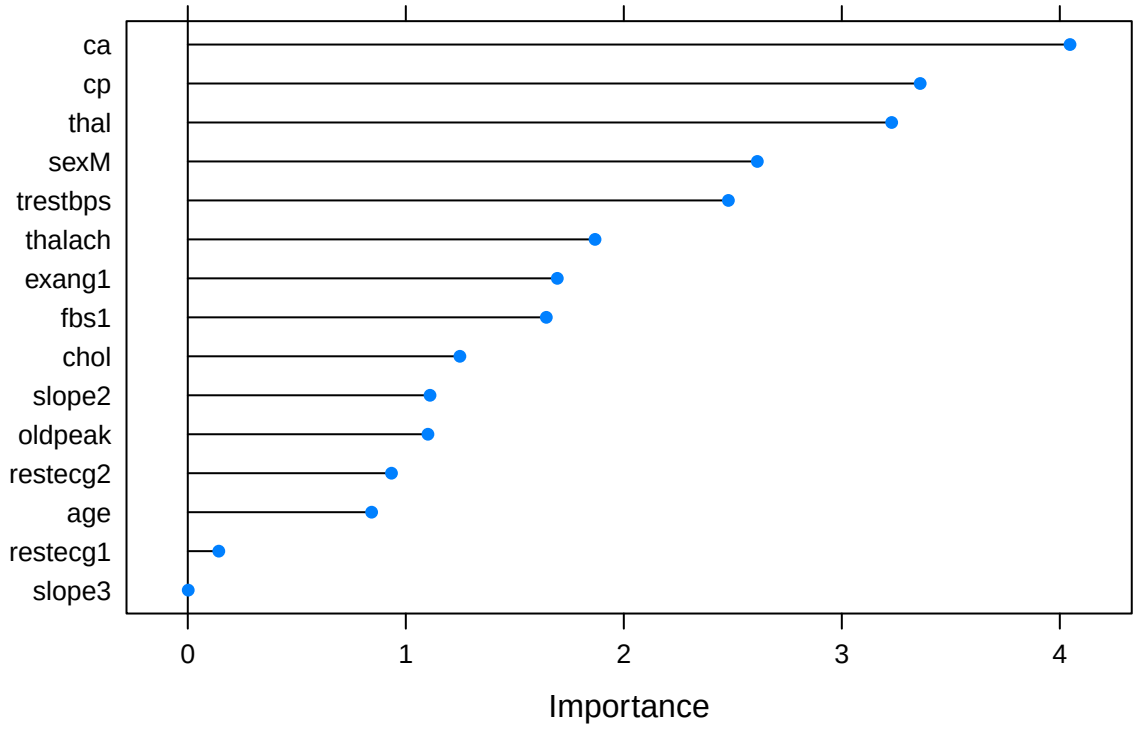



```
y <- barplot(
  pf_result,
  beside = TRUE,
  horiz = TRUE,
  col = brewer.pal(3, "Set1"),
  border = "white",
  legend.text = c("Accuracy", "Specificity", "Sensitivity"),
  args.legend = list(bty = "n", cex = 0.4),
  xlim = c(0, 100),
  main = "Performance Chart",
  ylab = "Method"
)

x <- round(pf_result)

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

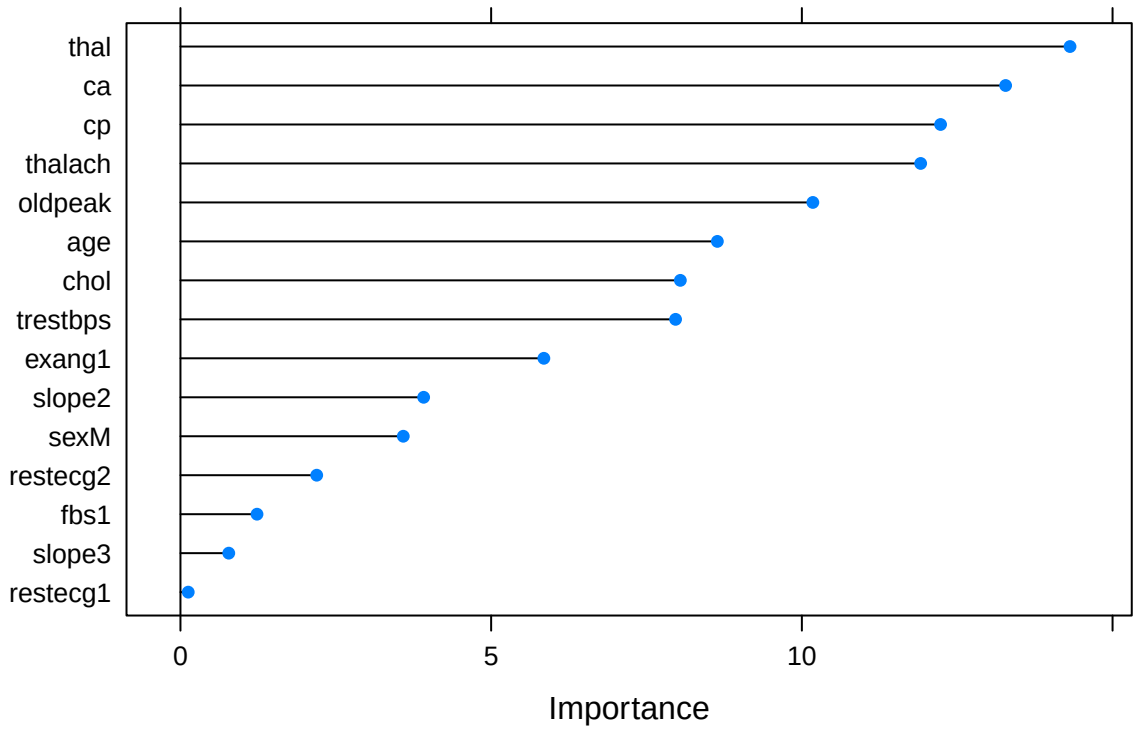



```
##                        Feature extraction                          -
```

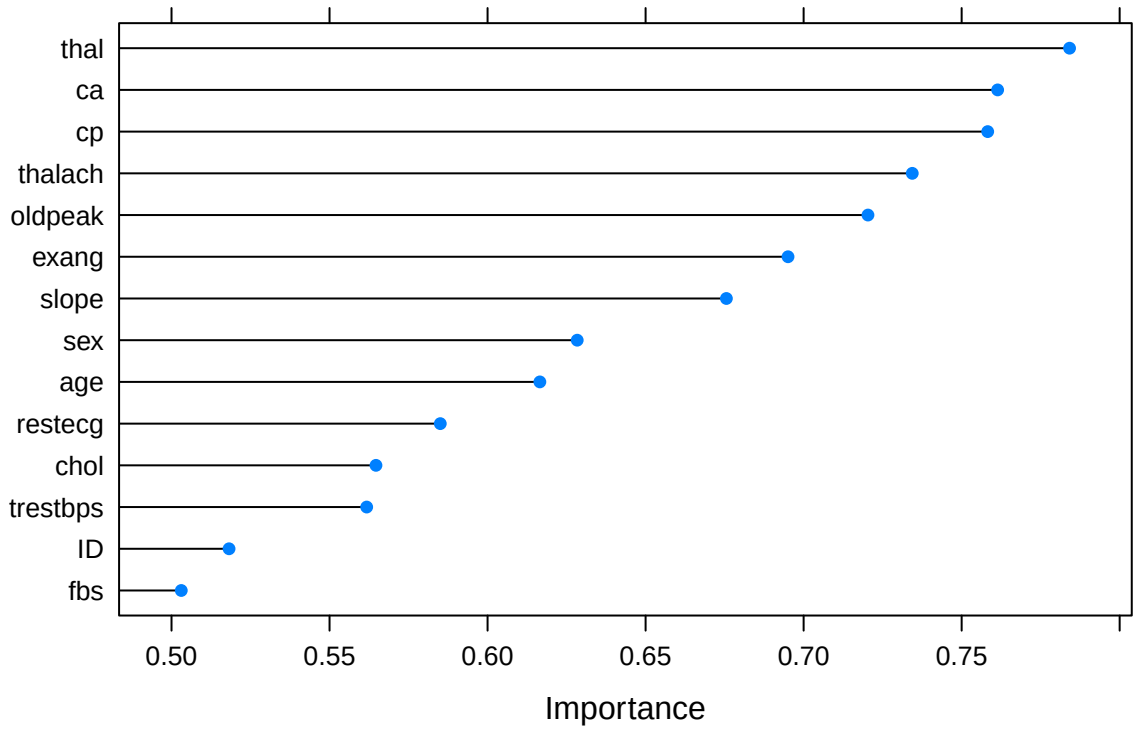
```
feat_log <- varImp(log_fit, scale = FALSE)
feat_rf <- varImp(rf_fit, scale = FALSE)
feat_blog <- varImp(blog_fit, scale = FALSE)
feat_knn <- varImp(knn_fit, scale = FALSE)

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



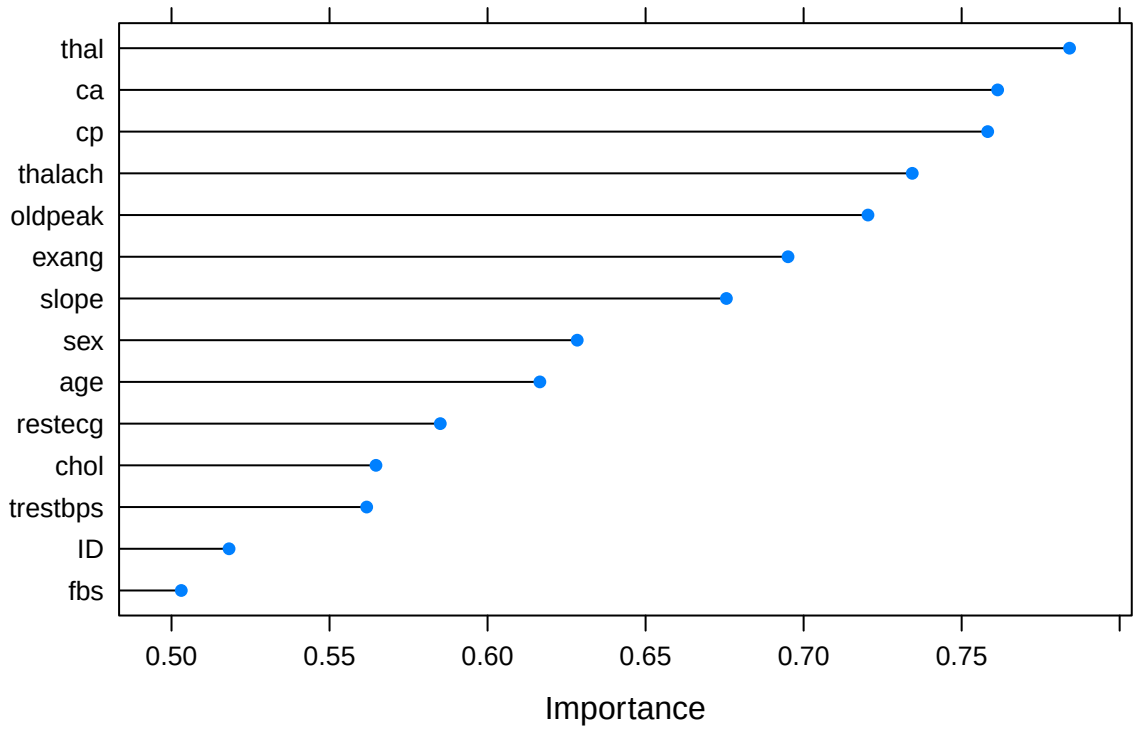

```
plot(feat_rf, main = "Random forest: features")
```




```
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.
```

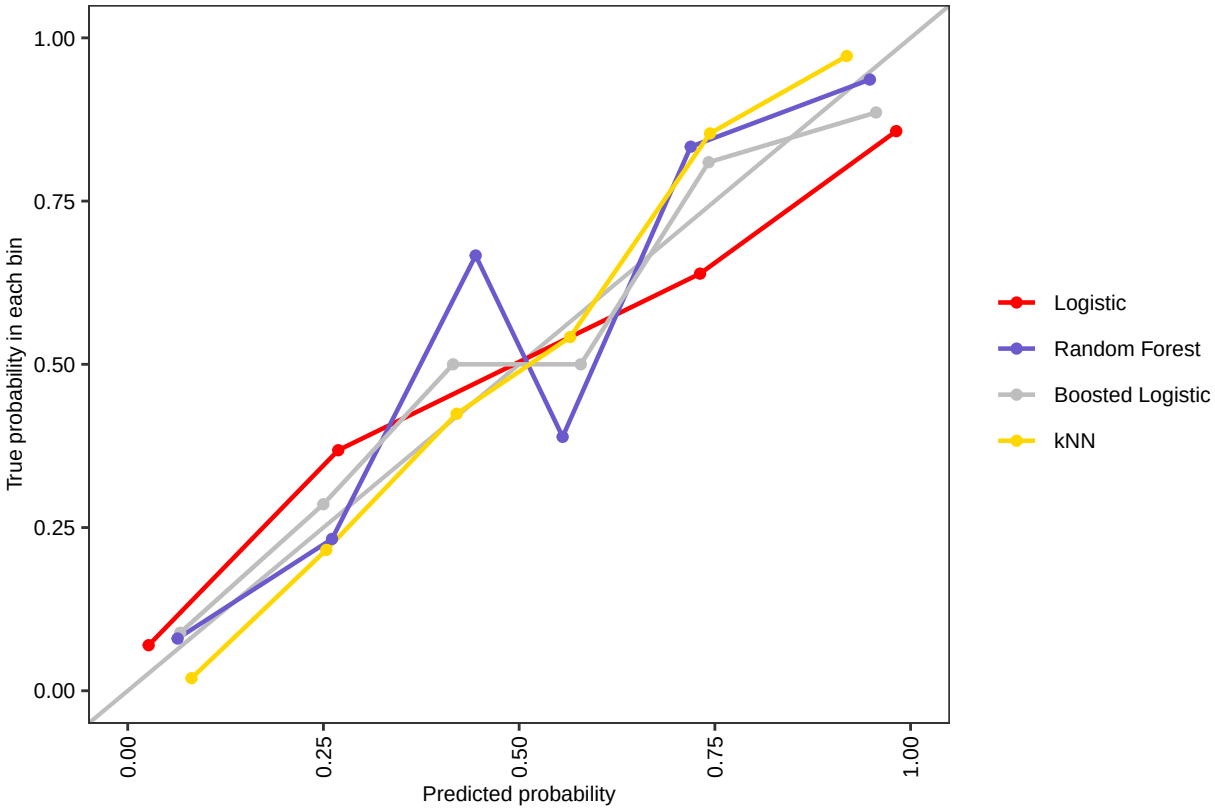
```
blog <- train(
  goal ~ . - ID,
  data = train_data,
  preProc = c("center", "scale"),
  method = "LogitBoost",
  trControl = cont
)
```

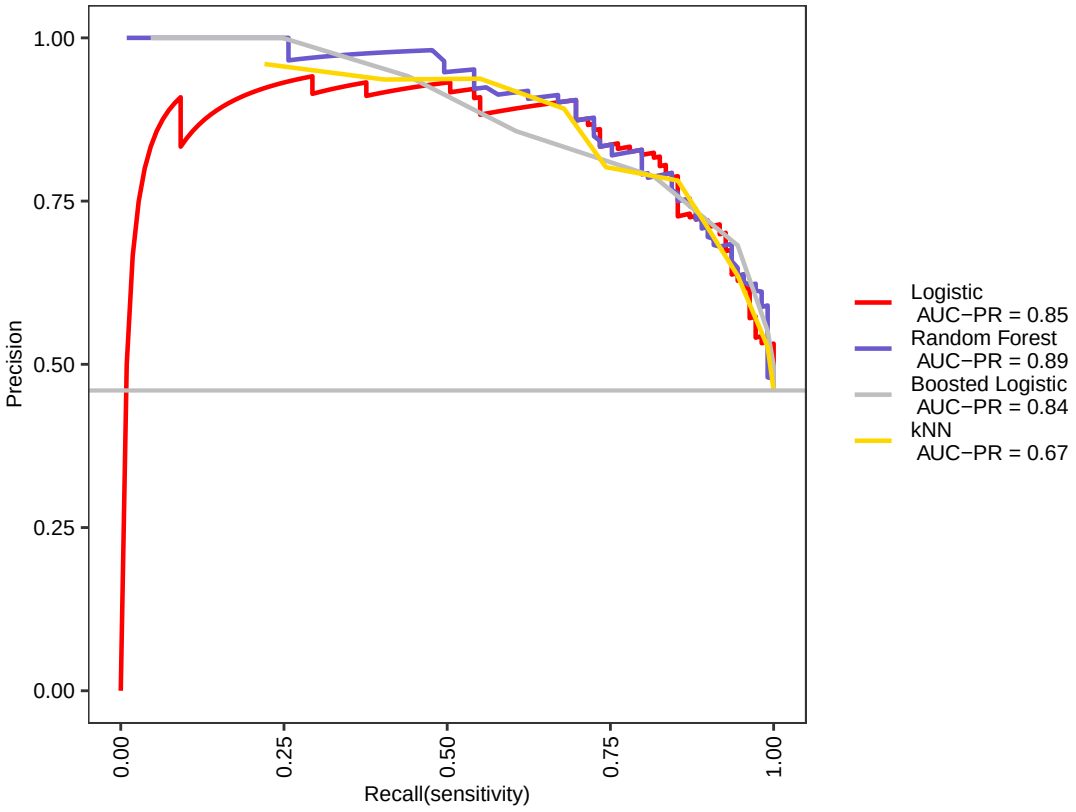
```
## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" was not
## in the result set. ROC will be used instead.
```

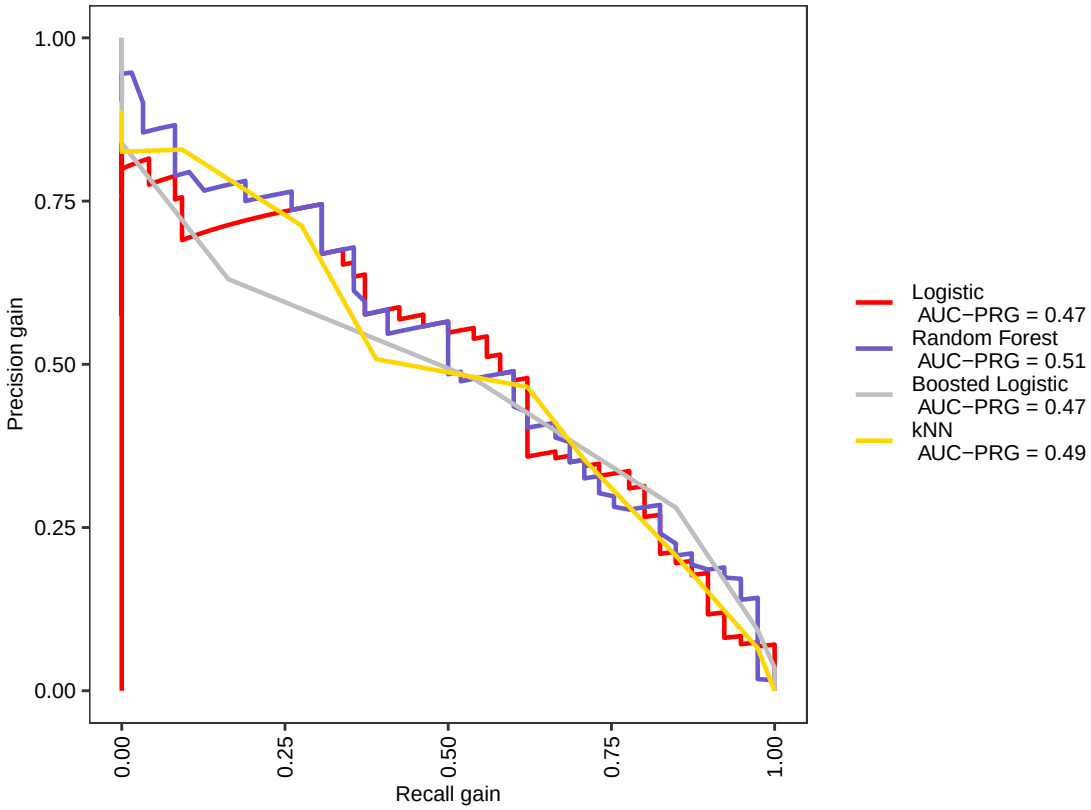
```
knn <- train(
  goal ~ . - ID ,
  data = train_data,
  method = "knn",
  preProcess = c("center", "scale"),
  trControl = cont ,
  tuneGrid = expand.grid(k = seq(1, 20, 2))
)
```

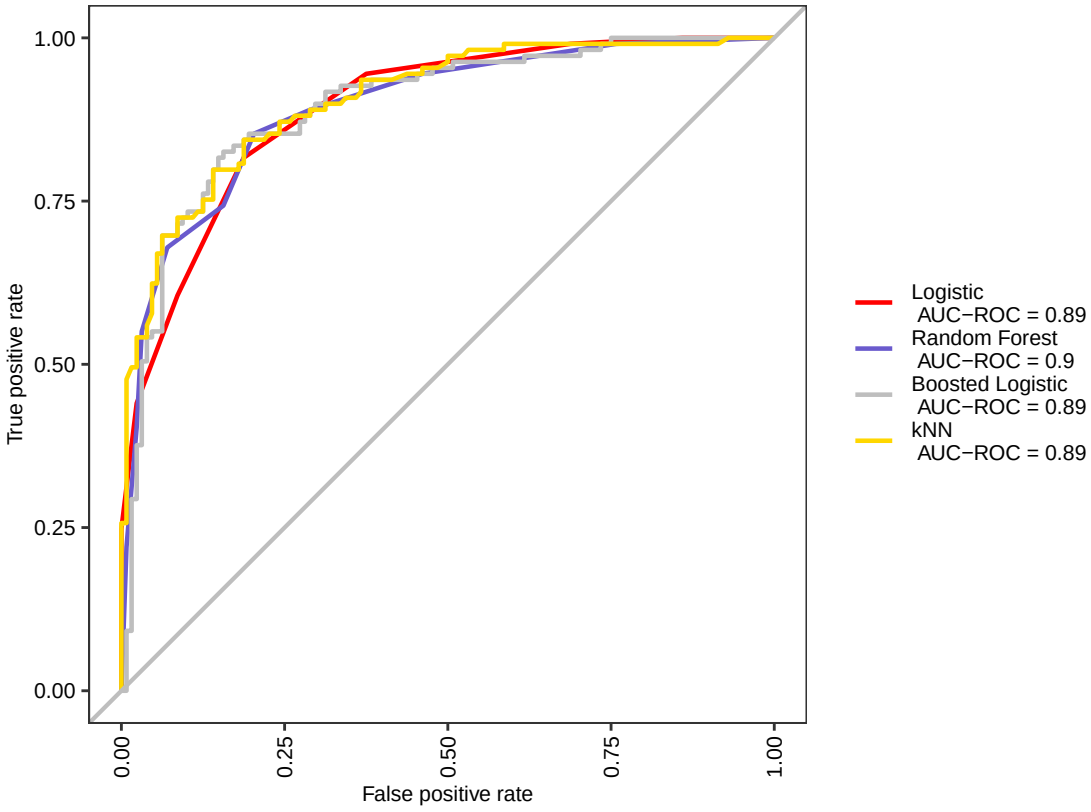
```
## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" was not
## in the result set. ROC will be used instead.
```

```
######################################################################
##                              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
True probability in each bin
0.00
0.25
0.50
0.75
1.00
Predicted probability
Logistic
Random Forest
Boosted Logistic
kNN


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 = 0.85
Random Forest
AUC−PR = 0.89
Boosted Logistic
AUC−PR = 0.84
kNN
AUC−PR = 0.67


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 = 0.47
Random Forest
AUC−PRG = 0.51
Boosted Logistic
AUC−PRG = 0.47
kNN
AUC−PRG = 0.49




```
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    0.89
## kNN              0.90
```

```
##                              EOL                              -
```