Project DS 3: Exploratory data analysis, and classifier training to predict condition using Cleveland Heart disease dataset with imputation of missing values.

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 4/processedWithHeader.cleveland.data",
    show_col_types = FALSE,
    na = "?"
  )
heart_data <- data.frame(processedWithHeader_cleveland)
```

```
##                    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
```

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

###  5 imputed dataset values
impute$imp$ca
```

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

```
impute$imp$thal
```

```
##     1 2 3 4 5
## 88  3 3 3 3 3
## 267 7 7 6 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
```

```
data <- heart_data

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
##       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   :0.6634   Mean   :4.736                      Mean   :152.0
##  3rd Qu.:1.0000   3rd Qu.:7.000                      3rd Qu.:227.5
##  Max.   :3.0000   Max.   :7.000                      Max.   :303.0
```

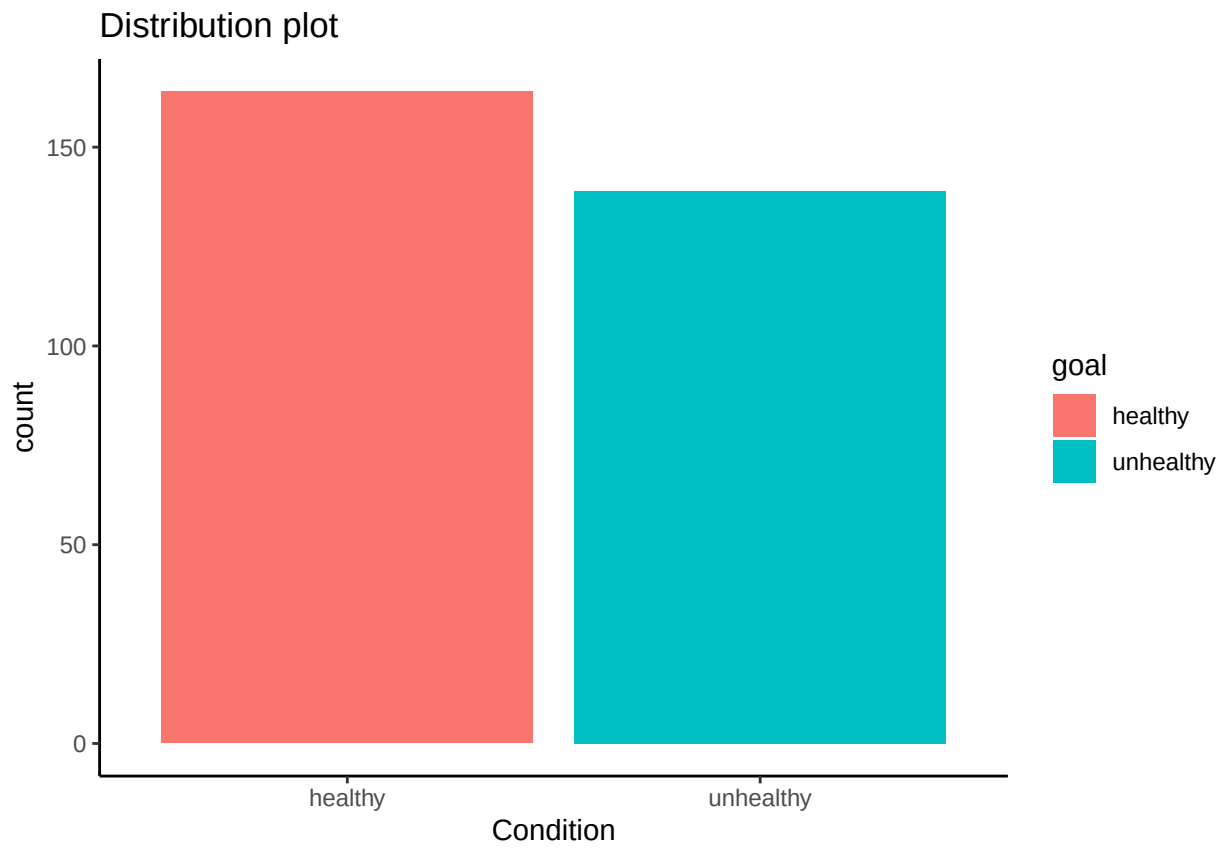
```
sum(is.na(heart_data))
```

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

```
###  missing block
# md.pattern(heart_data, plot = TRUE)

#####################################################################
##      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")
```



```
##############################################################################
##                  Run this chunk after decommenting it                    ##
##                to 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 = "Attributes",
#             y = "Attributes",
#             title = "Pairplot",
#             subtitle = "Group: Sex"
#           )

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

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

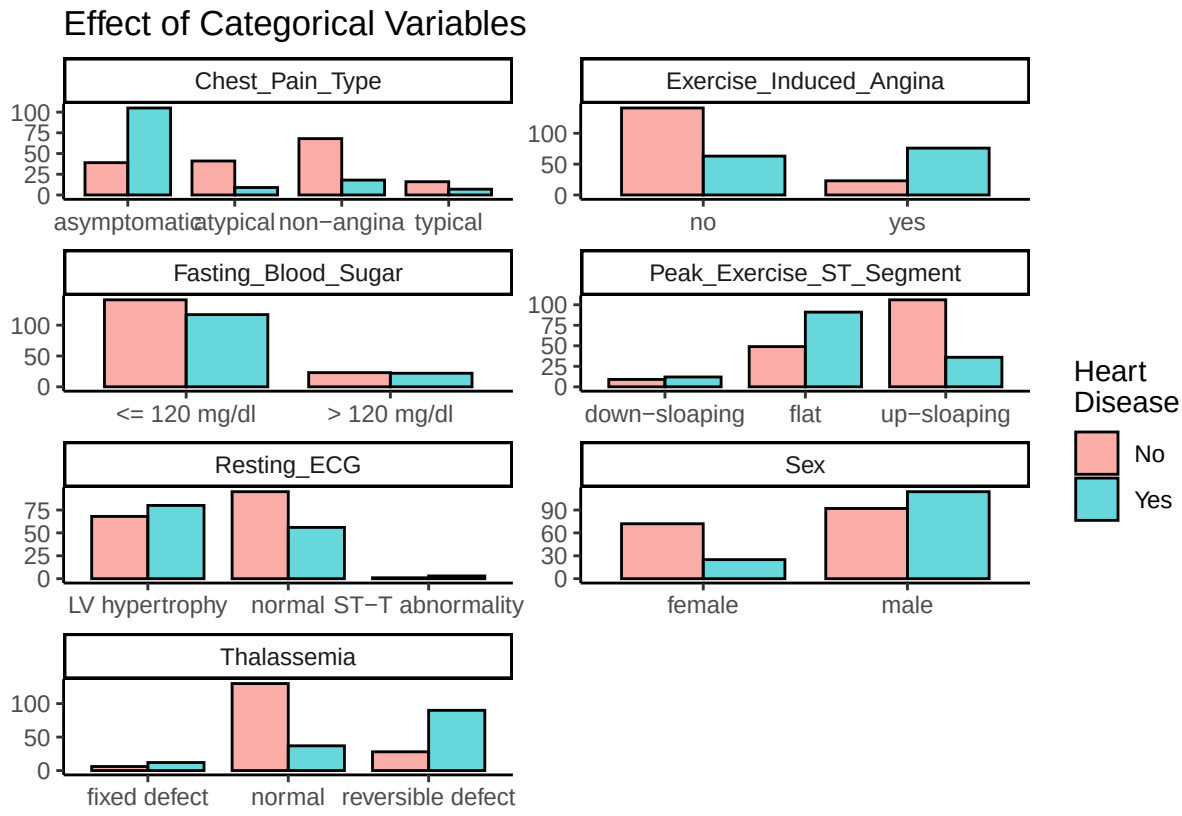
```
names <- c(
  "Age",
  "Sex",
  "Chest_Pain_Type",
  "Resting_Blood_Pressure",
  "Serum_Cholesterol",
  "Fasting_Blood_Sugar",
  "Resting_ECG",
  "Max_Heart_Rate_Achieved",
  "Exercise_Induced_Angina",
  "ST_Depression_Exercise",
  "Peak_Exercise_ST_Segment",
  "Num_Major_Vessels_Flouro",
  "Thalassemia",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

```
)

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)
```

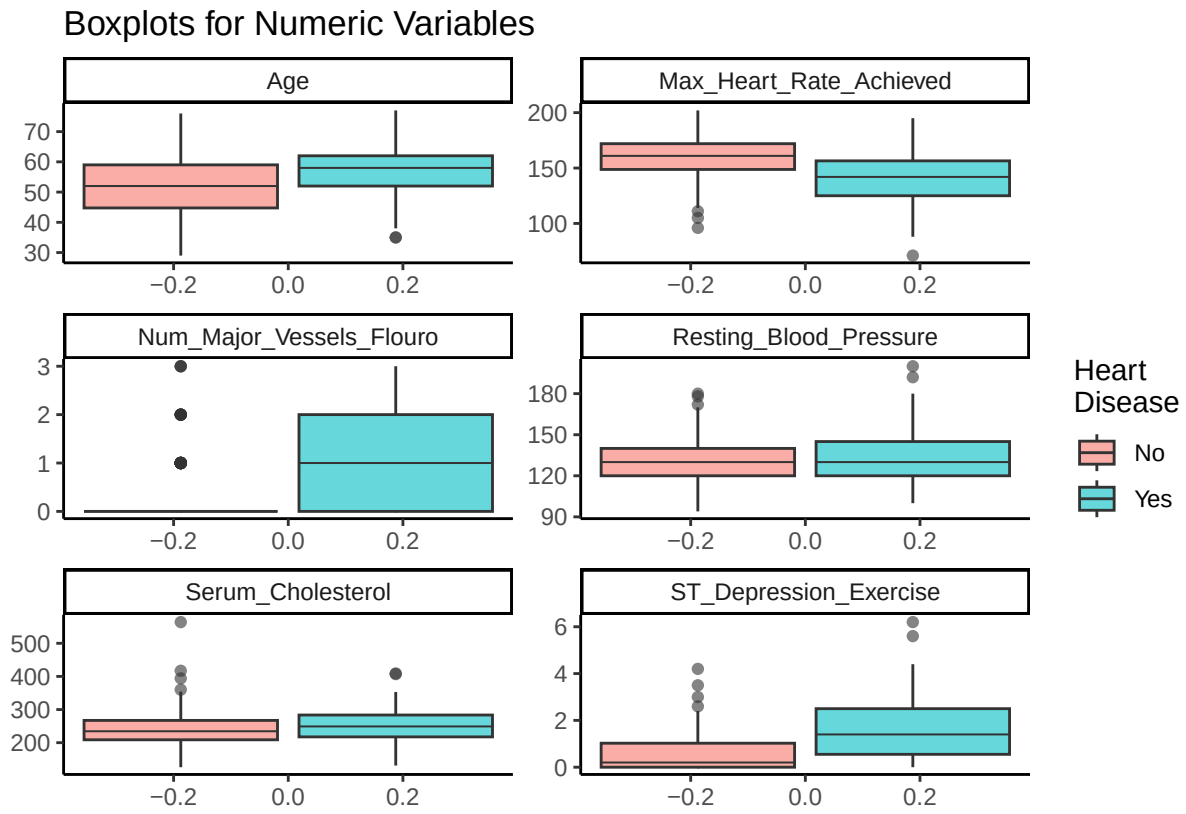
```
## Warning: attributes are not identical across measure variables;
## they will be dropped
```

```
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()
```



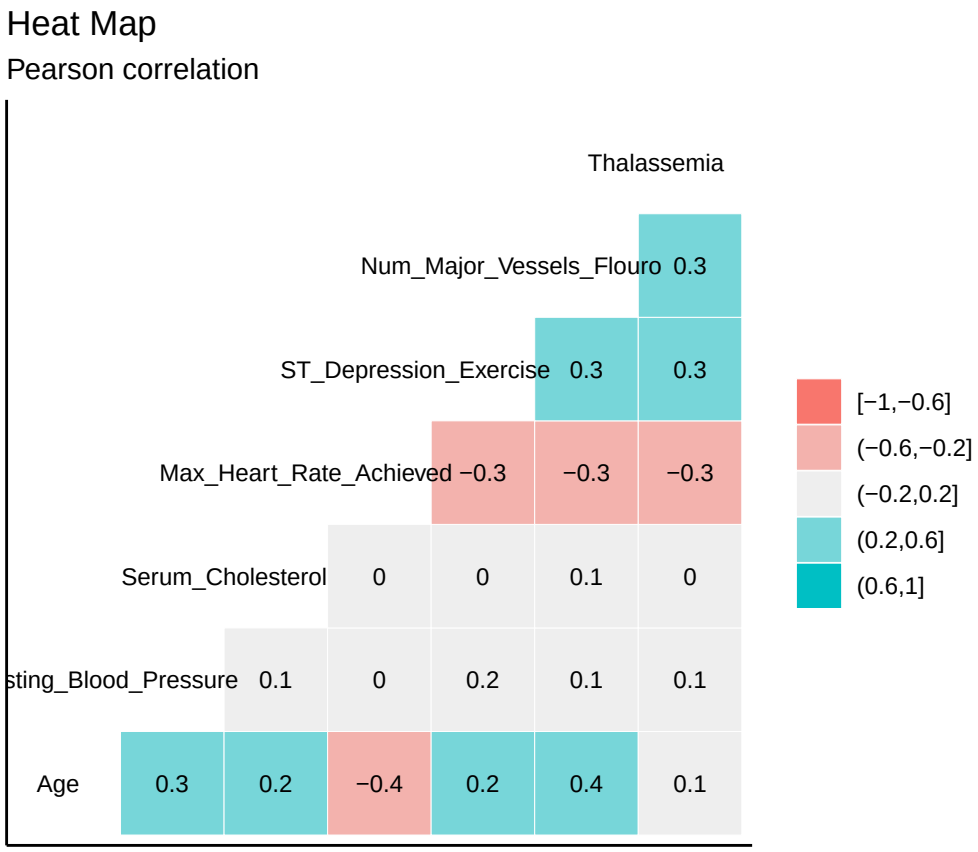

```
data.long.cont <- data  %>%
  select(
    Age,
    Resting_Blood_Pressure,
    Serum_Cholesterol,
    Max_Heart_Rate_Achieved,
    ST_Depression_Exercise,
    Num_Major_Vessels_Flouro,
    Diagnosis_Heart_Disease
  ) %>%
  gather(key   = "key",
         value = "value",-Diagnosis_Heart_Disease)

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()
```



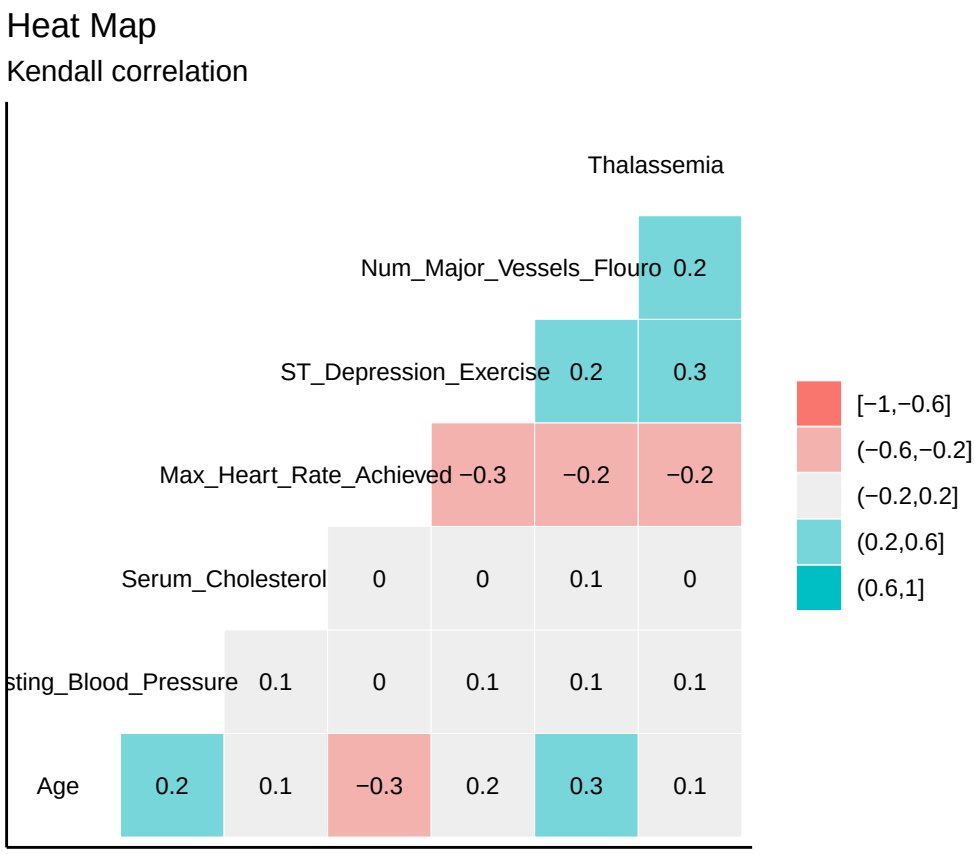

```
data %>% ggcorr(
  high       = "#00BFC4",
  low        = "#F8766D",
  label      = TRUE,
  hjust      = .75,
  size       = 3,
  label_size = 3,
  nbreaks    = 5
) +
  labs(title = "Heat Map", subtitle = "Pearson correlation") +
  theme_classic()
```

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

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

```
## Prediction  healthy unhealthy
##   healthy        28         1
##   unhealthy       5        27
##
##                Accuracy : 0.9016
##                  95% CI : (0.7981, 0.963)
##     No Information Rate : 0.541
##     P-Value [Acc > NIR] : 1.252e-09
##
##                   Kappa : 0.8041
##
##  Mcnemar's Test P-Value : 0.2207
##
##             Sensitivity : 0.8485
##             Specificity : 0.9643
##          Pos Pred Value : 0.9655
##          Neg Pred Value : 0.8438
##              Prevalence : 0.5410
##          Detection Rate : 0.4590
##    Detection Prevalence : 0.4754
##       Balanced Accuracy : 0.9064
##
##        '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         3
##   unhealthy       2        25
##
##                Accuracy : 0.918
##                  95% CI : (0.819, 0.9728)
##     No Information Rate : 0.541
##     P-Value [Acc > NIR] : 1.542e-10
##
##                   Kappa : 0.8345
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9394
##             Specificity : 0.8929
##          Pos Pred Value : 0.9118
##          Neg Pred Value : 0.9259
```

```
##              Prevalence : 0.5410
##          Detection Rate : 0.5082
##    Detection Prevalence : 0.5574
##       Balanced Accuracy : 0.9161
##
##        '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.              -
```

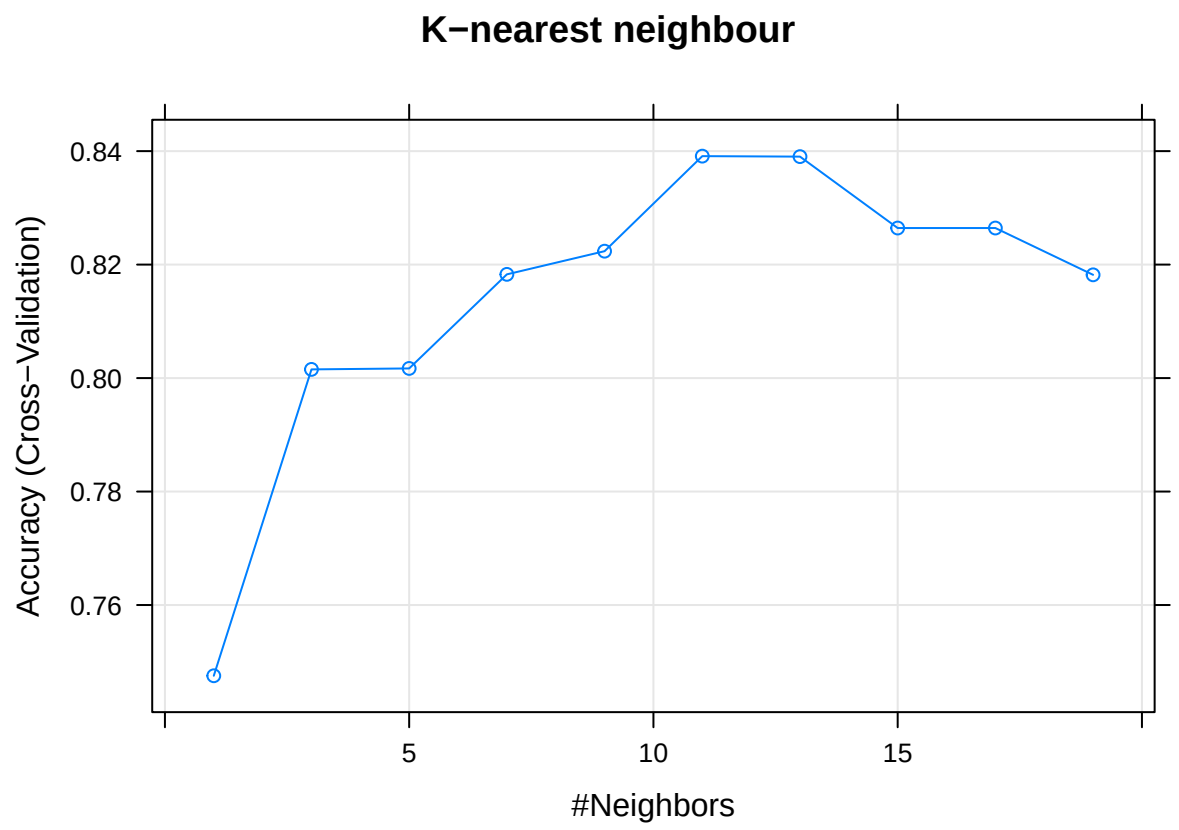
```
set.seed(112)
blog_fit <- train(goal ~ . - ID,
                  data = train_data,
                  method = "LogitBoost")
blog_pred <- predict(blog_fit, test_data)
confusionMatrix(blog_pred, test_data$goal)
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  healthy unhealthy
##   healthy        29         4
##   unhealthy       4        24
##
##                Accuracy : 0.8689
##                  95% CI : (0.7578, 0.9416)
##     No Information Rate : 0.541
##     P-Value [Acc > NIR] : 5.049e-08
##
##                   Kappa : 0.7359
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8788
##             Specificity : 0.8571
##          Pos Pred Value : 0.8788
##          Neg Pred Value : 0.8571
##              Prevalence : 0.5410
##          Detection Rate : 0.4754
##    Detection Prevalence : 0.5410
##       Balanced Accuracy : 0.8680
##
##        'Positive' Class : healthy
##
```

```
##                             4. KNN                                -
```

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

```r
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")
```




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

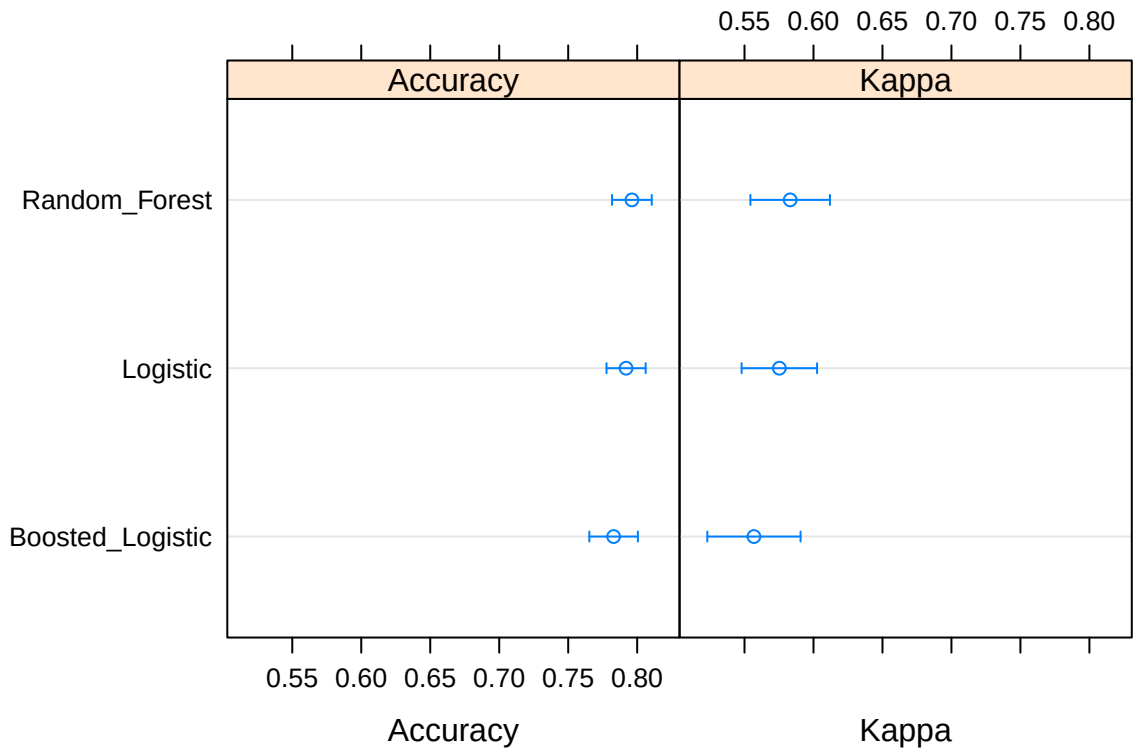
```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  healthy unhealthy
##   healthy        28         1
##   unhealthy       5        27
##
##                Accuracy : 0.9016
##                  95% CI : (0.7981, 0.963)
##     No Information Rate : 0.541
##     P-Value [Acc > NIR] : 1.252e-09
##
```

```
##                  Kappa : 0.8041
##
##  Mcnemar's Test P-Value : 0.2207
##
##             Sensitivity : 0.8485
##             Specificity : 0.9643
##          Pos Pred Value : 0.9655
##          Neg Pred Value : 0.8438
##              Prevalence : 0.5410
##          Detection Rate : 0.4590
##    Detection Prevalence : 0.4754
##       Balanced Accuracy : 0.9064
##
##        '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         0.7317073 0.7717391 0.8000000 0.7919459 0.8131868 0.8494624
## Random_Forest    0.7333333 0.7727273 0.7951807 0.7961772 0.8111111 0.8681319
## Boosted_Logistic 0.6923077 0.7555556 0.7872340 0.7829209 0.8131868 0.8709677
##                  NA's
## Logistic            0
## Random_Forest       0
## Boosted_Logistic    0
##
## Kappa
##                      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## Logistic         0.4602699 0.5333333 0.5859334 0.5752630 0.6196498 0.6859624
## Random_Forest    0.4559194 0.5396419 0.5594755 0.5831105 0.6102088 0.7294351
## Boosted_Logistic 0.3872054 0.5113230 0.5672878 0.5568318 0.6220376 0.7282026
##                  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)
```

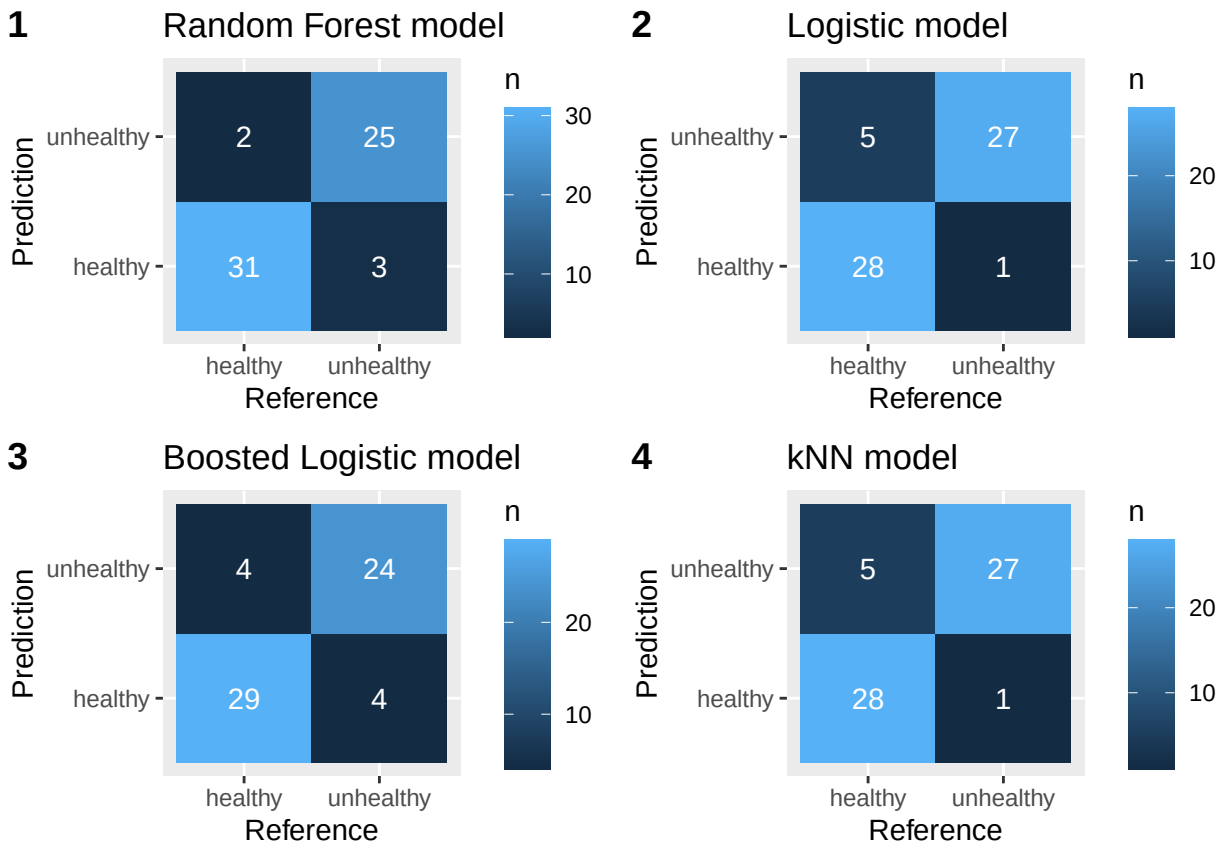
```
##                  Confusion Matrix as Heatmaps                  -
```

```
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")
```

```
## 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")
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")
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")

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



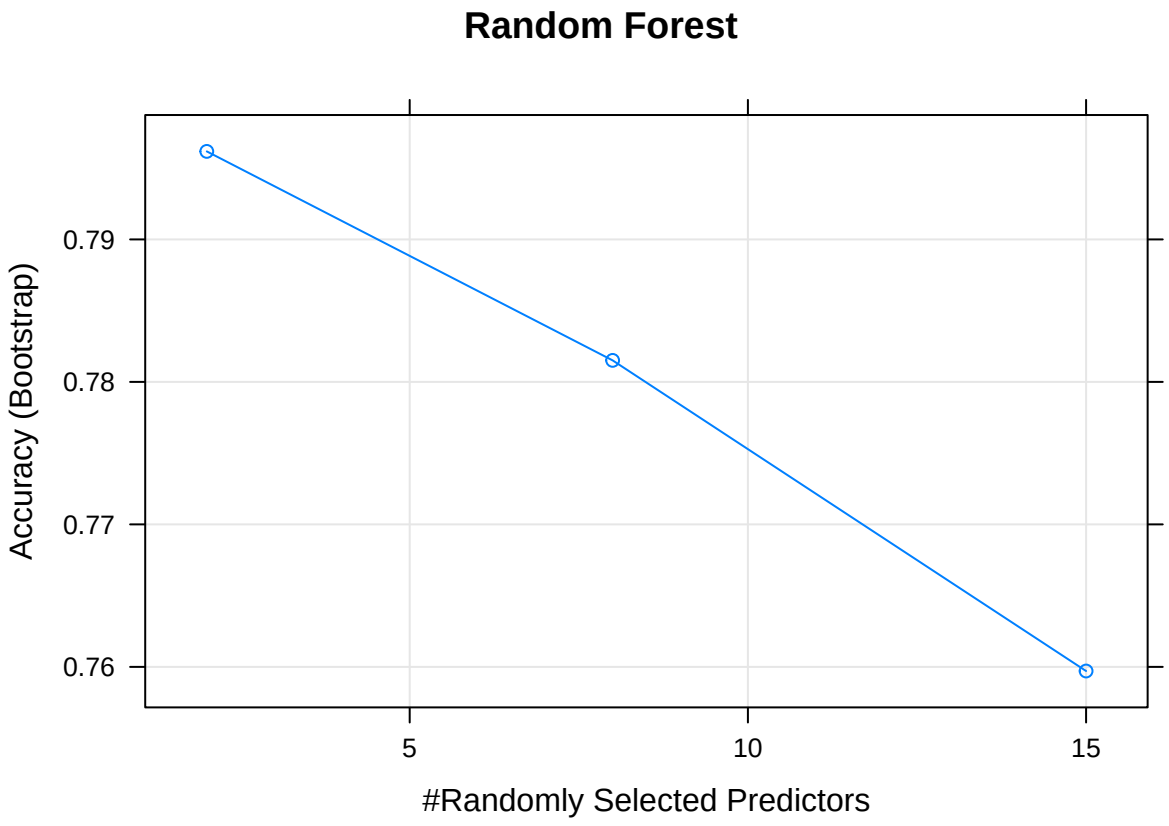

```
##                    Performance Comparison                    -
```

```
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"]],
```

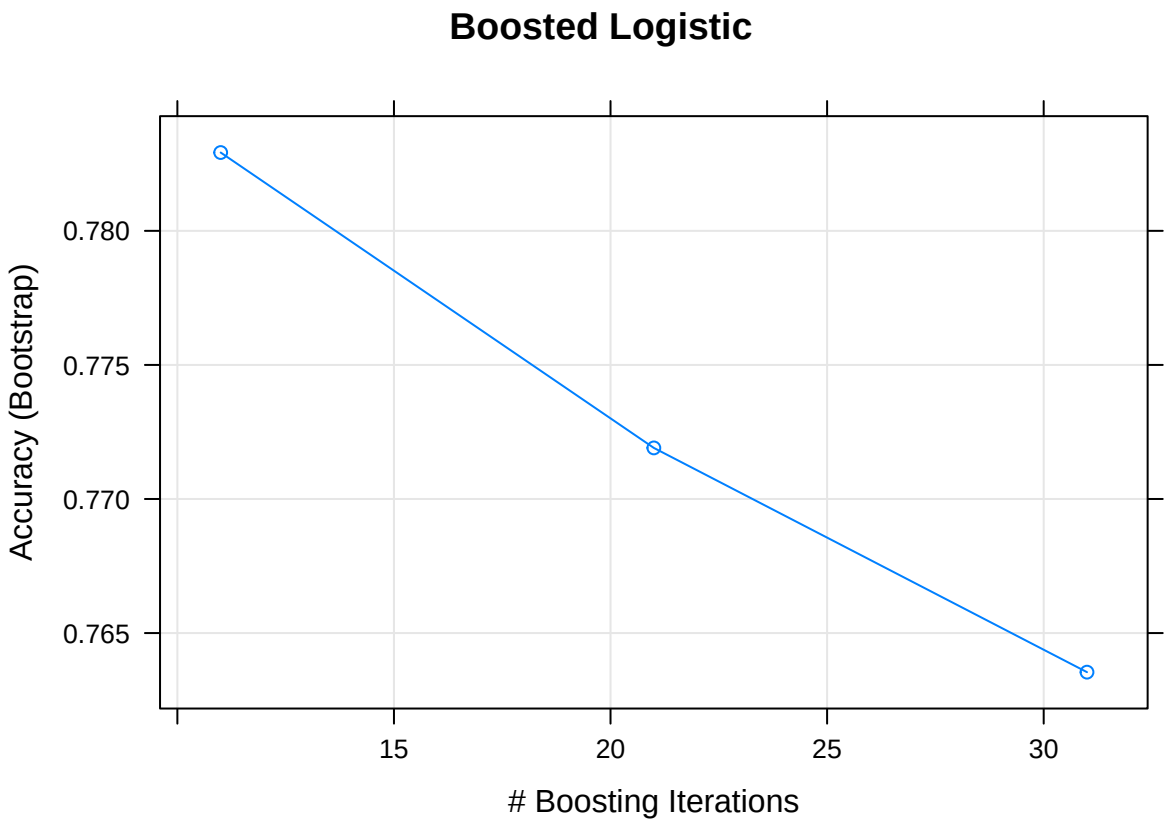
```
                           cf_knn[["byClass"]][["Specificity"]])

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    90.16393 91.80328 86.88525 90.16393
## Specificity 96.42857 89.28571 85.71429 96.42857
## Sensitivity 84.84848 93.93939 87.87879 84.84848
```

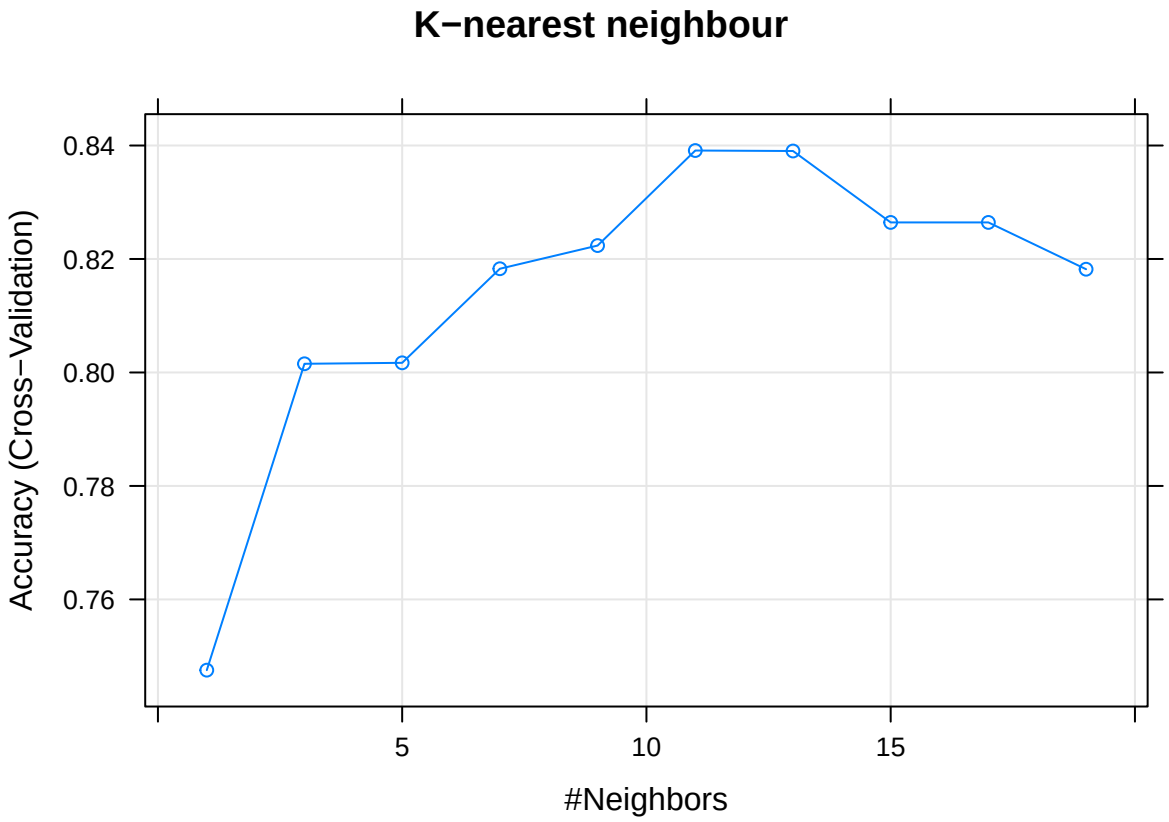
```
plot(rf_fit, main = "Random Forest")
```



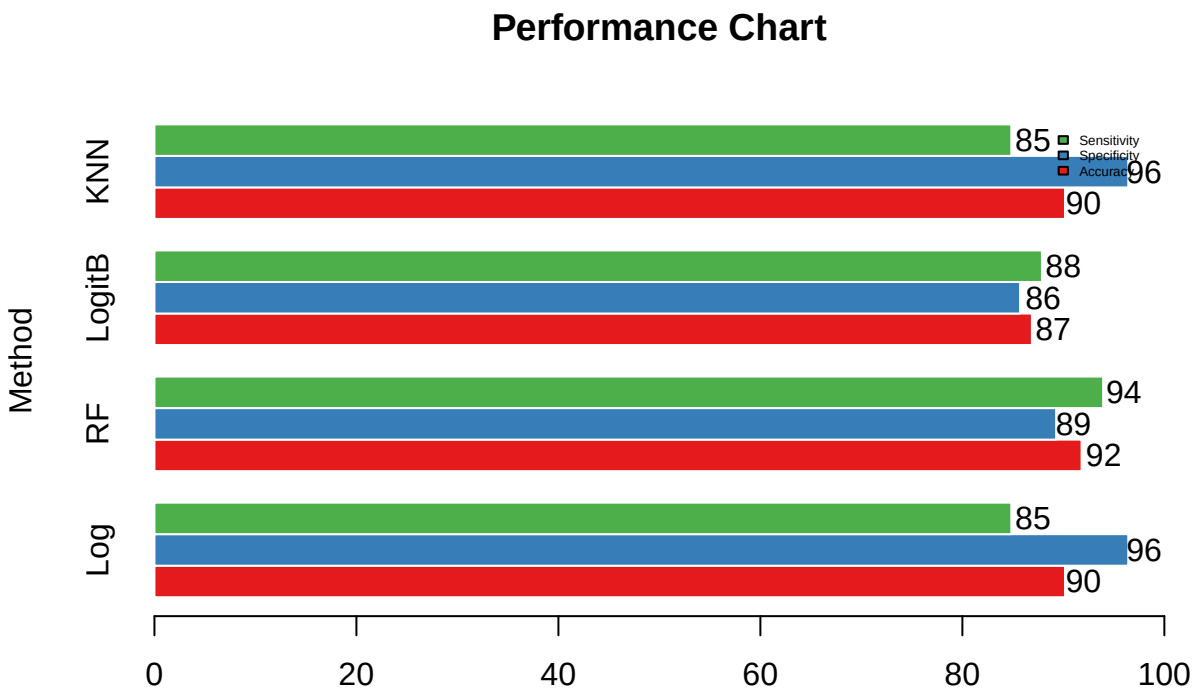

```
plot(blog_fit, main = "Boosted Logistic")
```




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

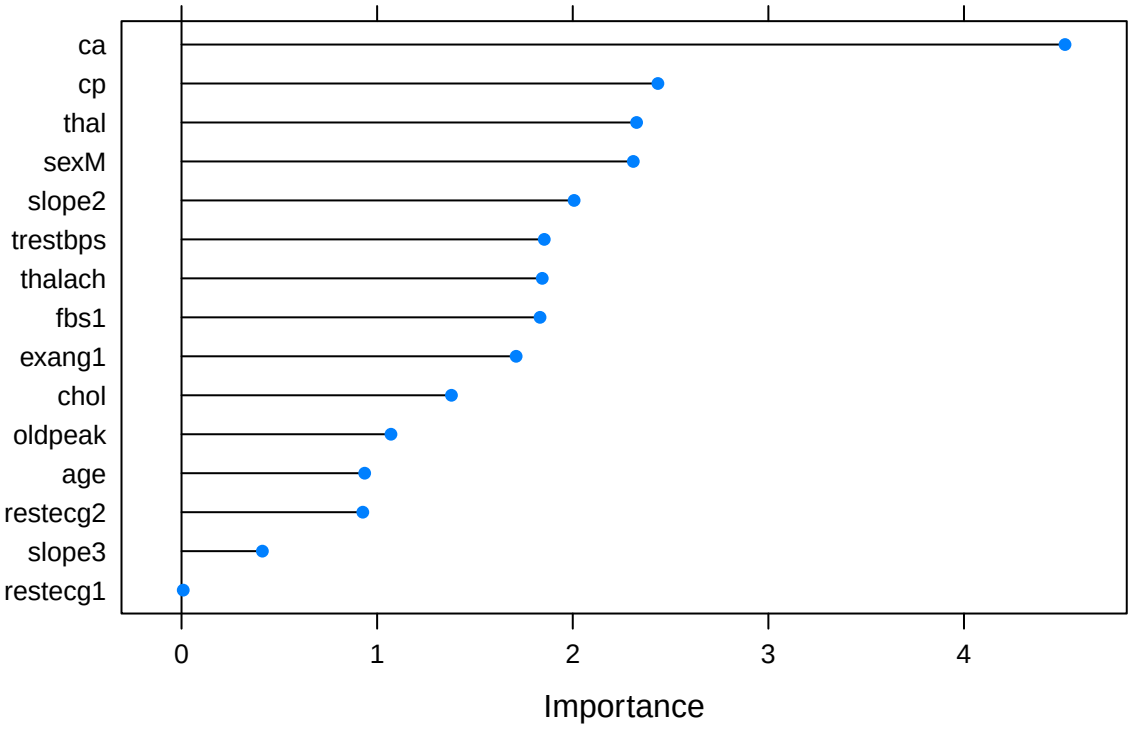



```
y <- barplot(
  pf_result,
  beside = TRUE,
  horiz = TRUE,
  col = brewer.pal(3, "Set1"),
  border = "white",
  legend.text = c("Accuracy", "Specificity", "Sensitivity"),
  args.legend = list(bty = "n", cex = 0.4),
  xlim = c(0, 100),
  main = "Performance Chart",
  ylab = "Method"
)

x <- round(pf_result)

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

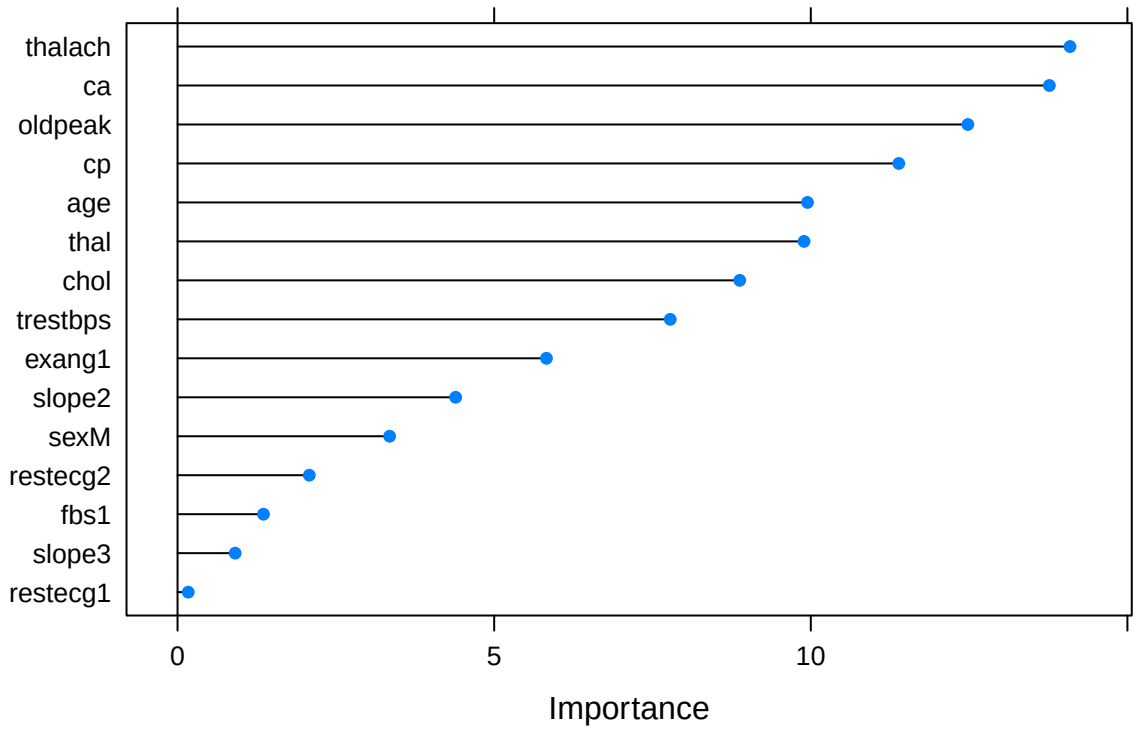



```
##                        Feature extraction                          -
```

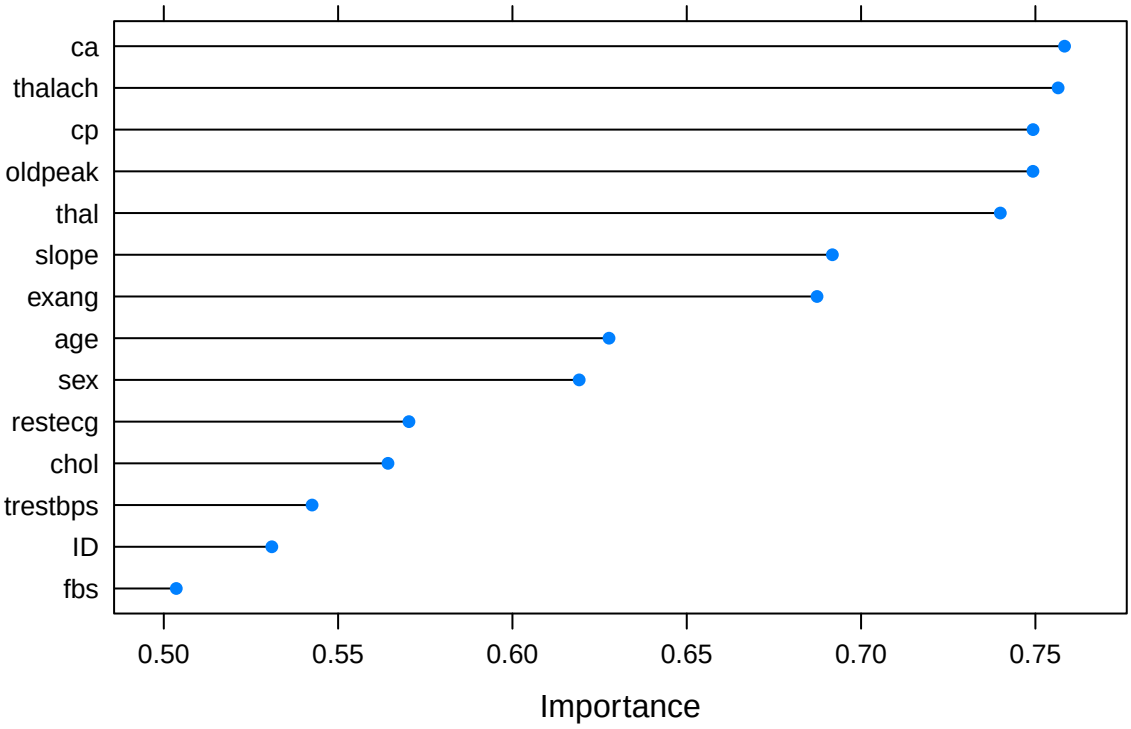
```
feat_log <- varImp(log_fit, scale = FALSE)
feat_rf <- varImp(rf_fit, scale = FALSE)
feat_blog <- varImp(blog_fit, scale = FALSE)
feat_knn <- varImp(knn_fit, scale = FALSE)

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



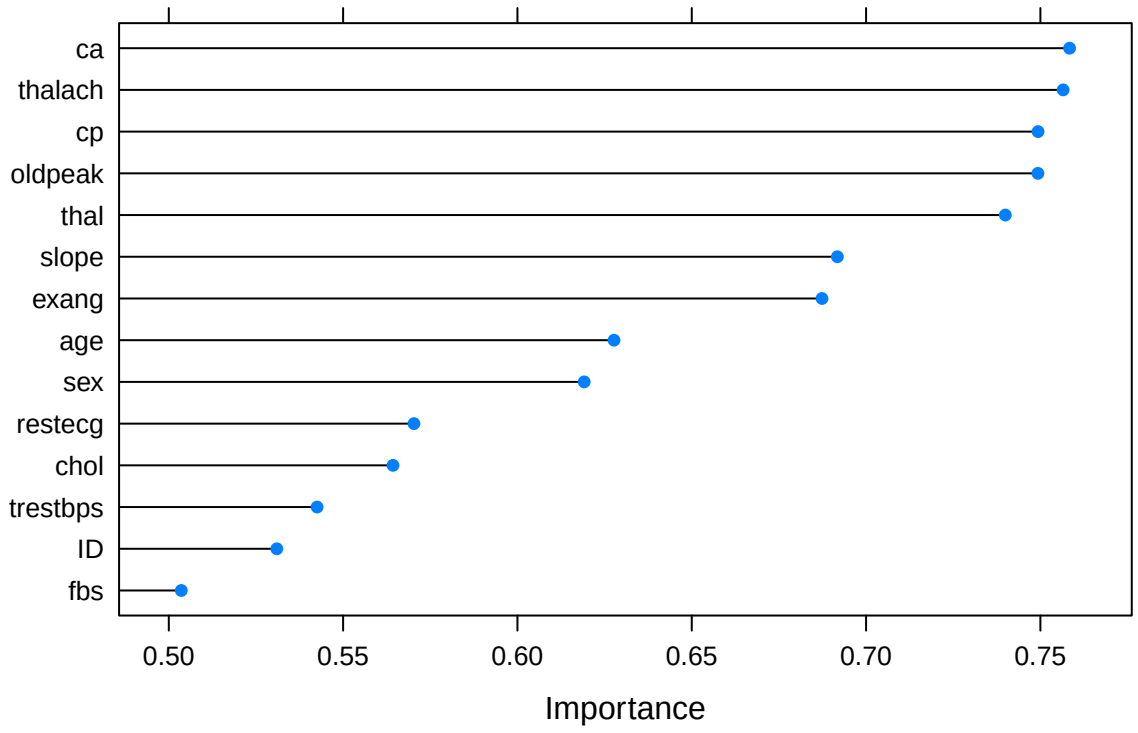

```
plot(feat_rf, main = "Random forest: features")
```




```
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.
```

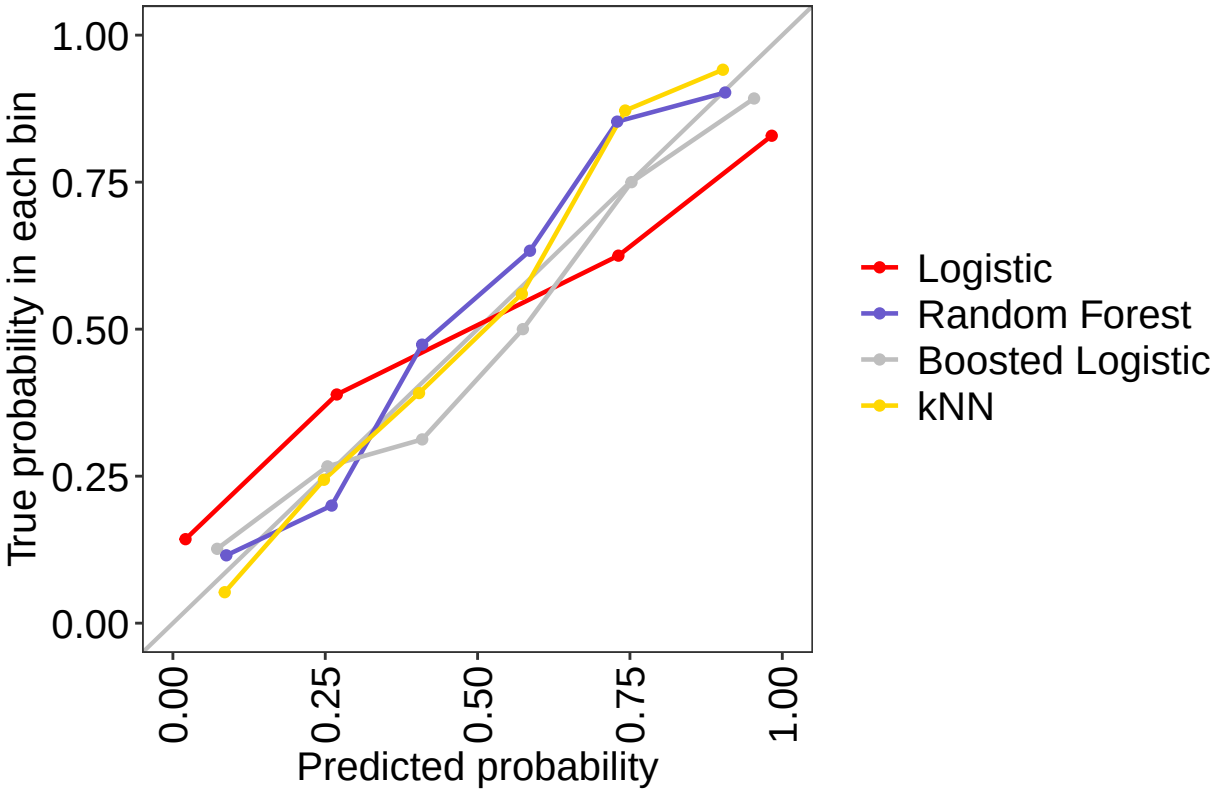
```
blog <- train(
  goal ~ . - ID,
  data = train_data,
  preProc = c("center", "scale"),
  method = "LogitBoost",
  trControl = cont
)
```

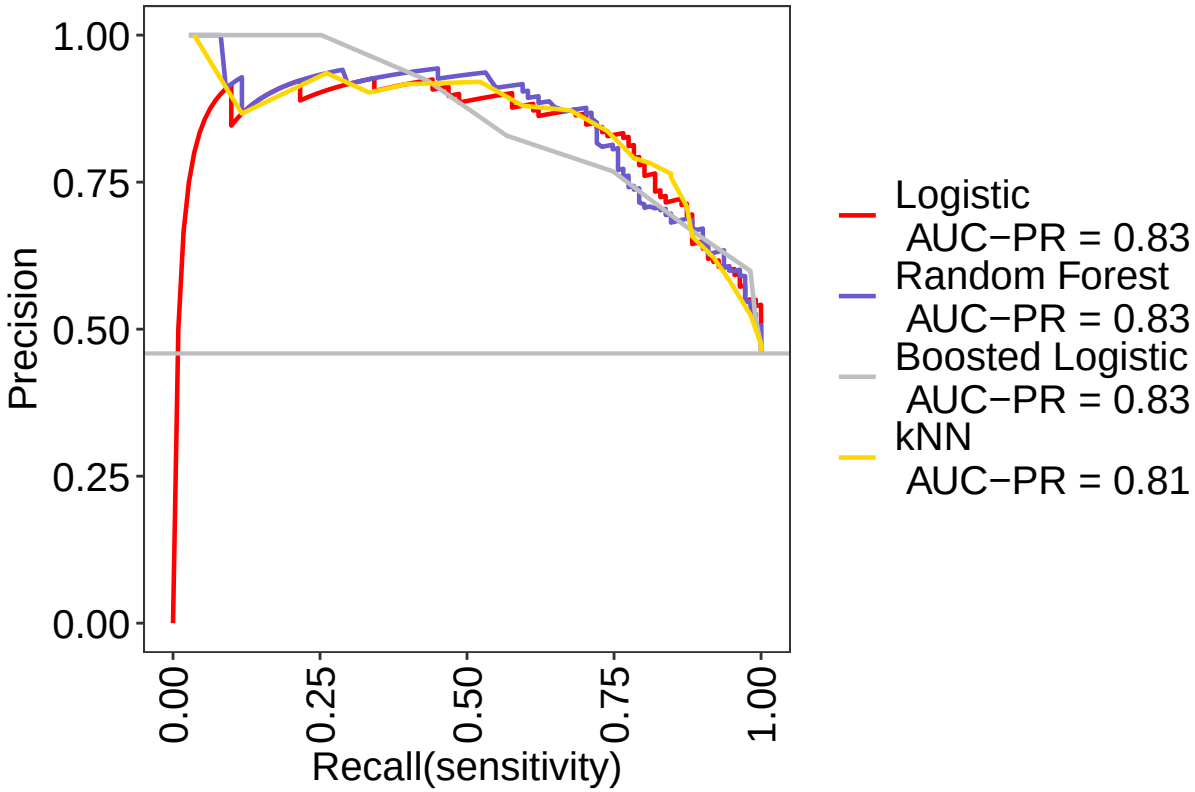
```
## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" was not
## in the result set. ROC will be used instead.
```

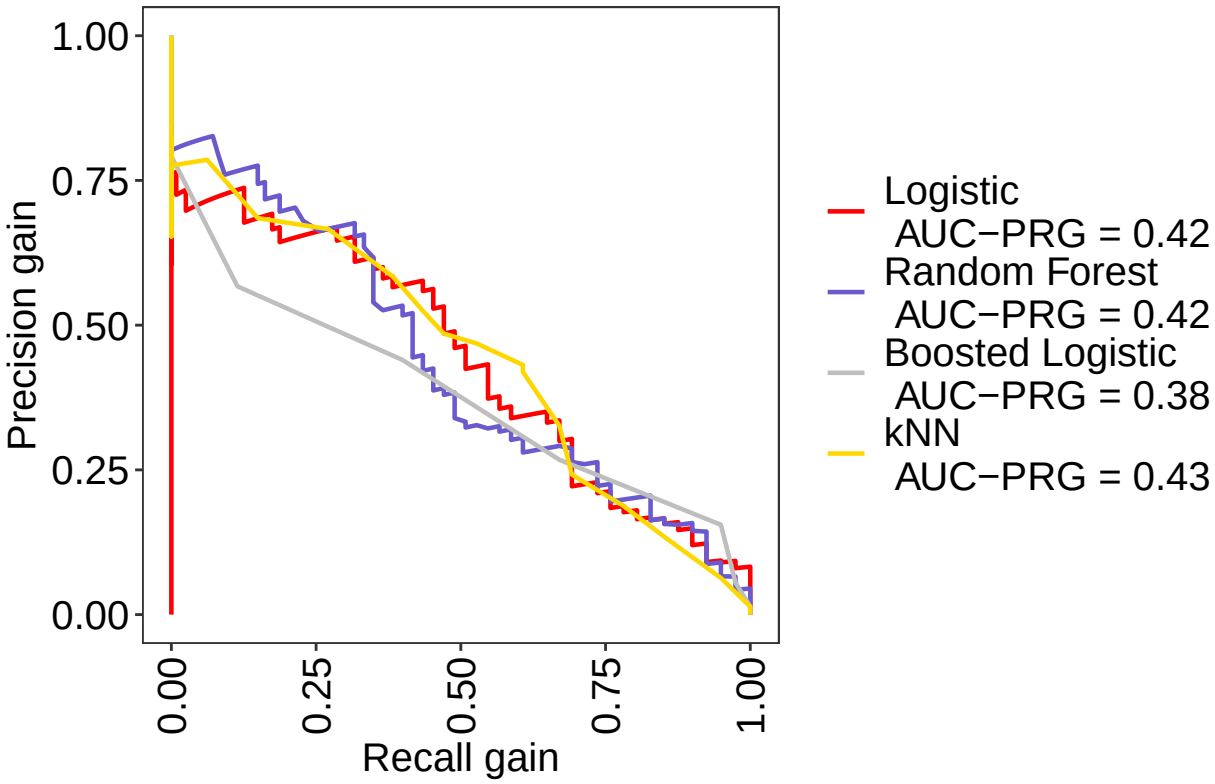
```
knn <- train(
  goal ~ . - ID ,
  data = train_data,
  method = "knn",
  preProcess = c("center", "scale"),
  trControl = cont ,
  tuneGrid = expand.grid(k = seq(1, 20, 2))
)
```

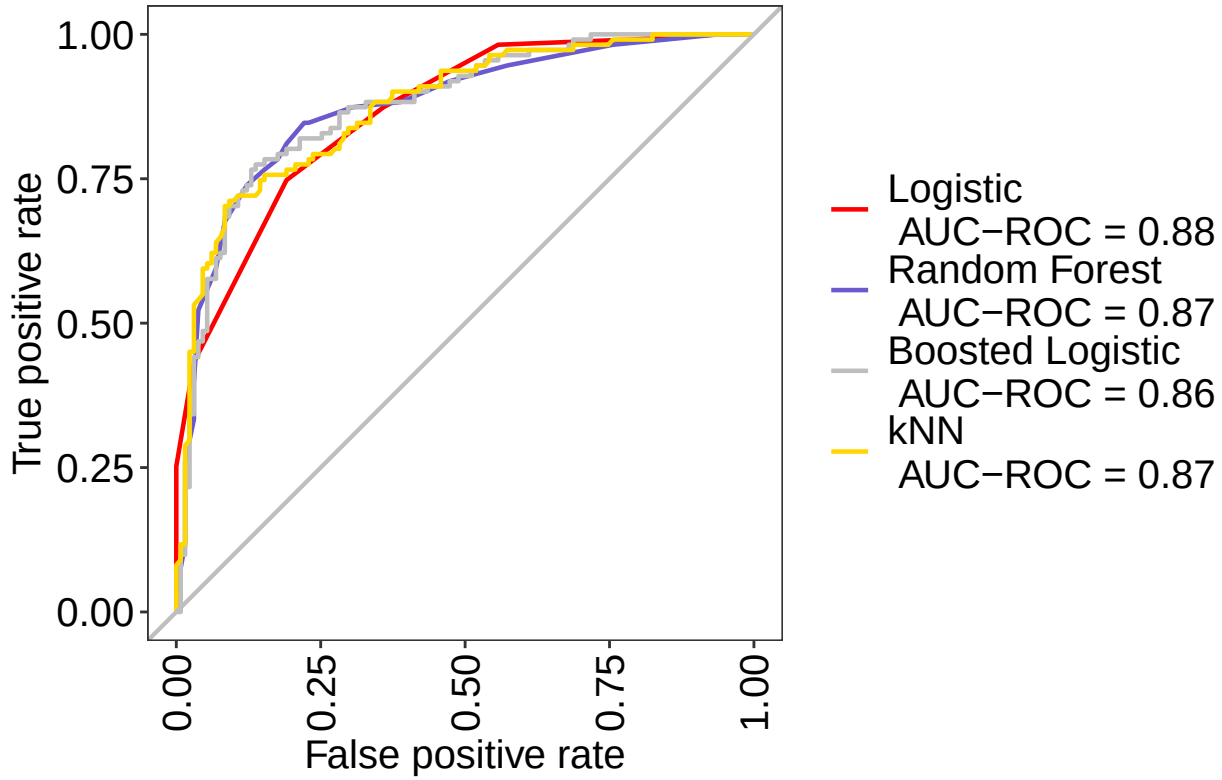
```
## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" was not
## in the result set. ROC will be used instead.
```

```
######################################################################
##                              AUC-ROC                             ##
######################################################################
metric <- evalm(
  list(log, rf, blog, knn),
  gnames = c('Logistic', 'Random Forest',
             'Boosted Logistic', 'kNN'),
  rlinethick = 0.8,
  fsize = 15,
  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 = 0.83
Random Forest
AUC−PR = 0.83
Boosted Logistic
AUC−PR = 0.83
kNN
AUC−PR = 0.81


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 = 0.42
Random Forest
AUC−PRG = 0.42
Boosted Logistic
AUC−PRG = 0.38
kNN
AUC−PRG = 0.43




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

```
##                              EOL                              -
```