p8106 Final Project

Qixiang Chen

5/5/2022

TO-DO List: 1. explanatory analysis (Is there any interesting structure present in the data? What were your findings?) 2. visualization work (feature plot done, plots for factors) 3. build 6 classification models 4. build a frame for the final model selection (ROC, AUC, ConfusionMatrix, error rate) 5. Try to list out the important predictors 6. tuning parameters 7. interpretation of each model

In this dataset, `age` refers to age in days. For variable `gender`, 1 represents women, 2 represents men. Thus, we need to do the corresponding adjustments to make it look formal.

1. Since variable `id` does not contribute to the following analysis, we exclude `id` from the dataset.
2. For variable `gender`, 1 represents women, 2 represents men. To make it serve as a dummy variable, we convert it into factor.

```
df = read.csv("./cardio_train.csv", header = TRUE, stringsAsFactors = FALSE, sep = ";") %>%
  janitor::clean_names() %>%
  dplyr::select(-id) %>%
  rename(age_day = age) %>%
  mutate(gender = gender - 1,
         cholesterol = case_when(cholesterol == 1 ~ "normal",
                                 cholesterol == 2 ~ "above normal",
                                 cholesterol == 3 ~ "well above normal"
                                 ),
         gluc = case_when(gluc == 1 ~ "normal",
                          gluc == 2 ~ "above normal",
                          gluc == 3 ~ "well above normal"
                          ),
         gender = as.factor(gender),
         smoke = as.factor(smoke),
         alco = as.factor(alco),
         active = as.factor(active),
         cardio = case_when(cardio == 0 ~ "nondiseased",
                            cardio == 1 ~ "diseased"
                            ),
         cardio = as.factor(cardio)
         ) %>%
  mutate(cholesterol = factor(cholesterol, levels = c("normal", "above normal", "well above normal")),
         gluc = factor(gluc, levels = c("normal", "above normal", "well above normal")),
         cardio = factor(cardio, levels = c("nondiseased", "diseased"))
         ) %>%
  dplyr::select(age_day, height, weight, ap_hi, ap_lo, everything())

#df
```

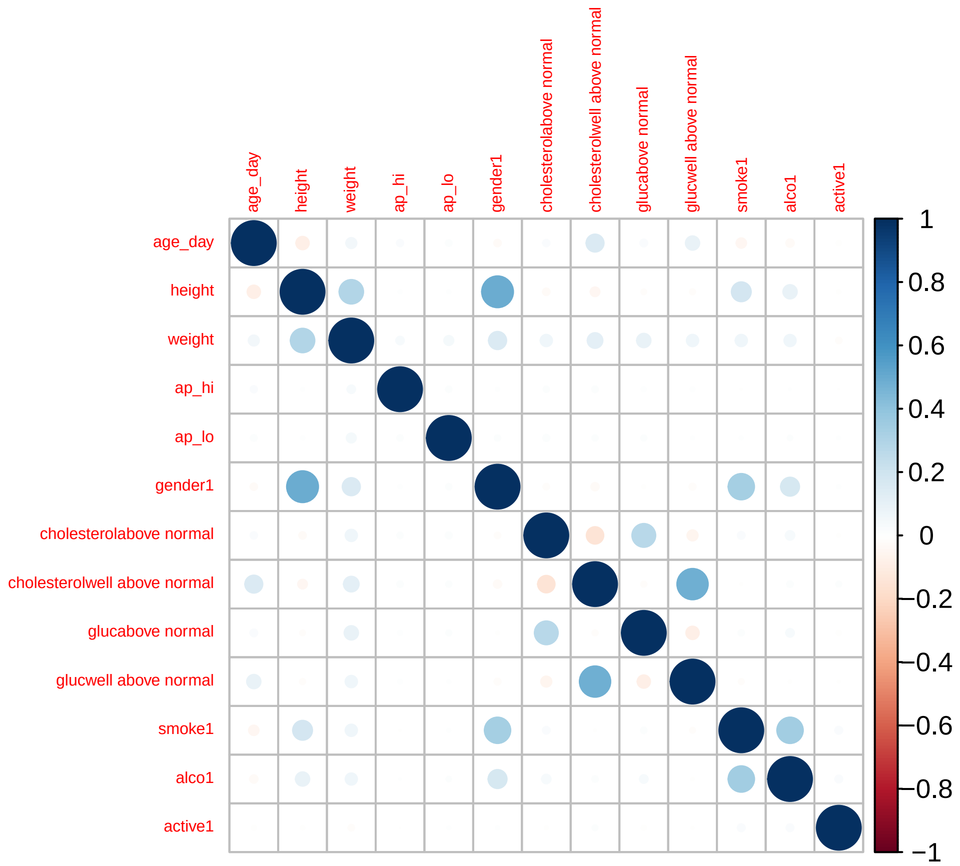
```r
#Check whether there is any missing value.
missing_train = sapply(df, function(x) sum(is.na(x)))
print(missing_train[missing_train > 0])
```

```
## named integer(0)
```

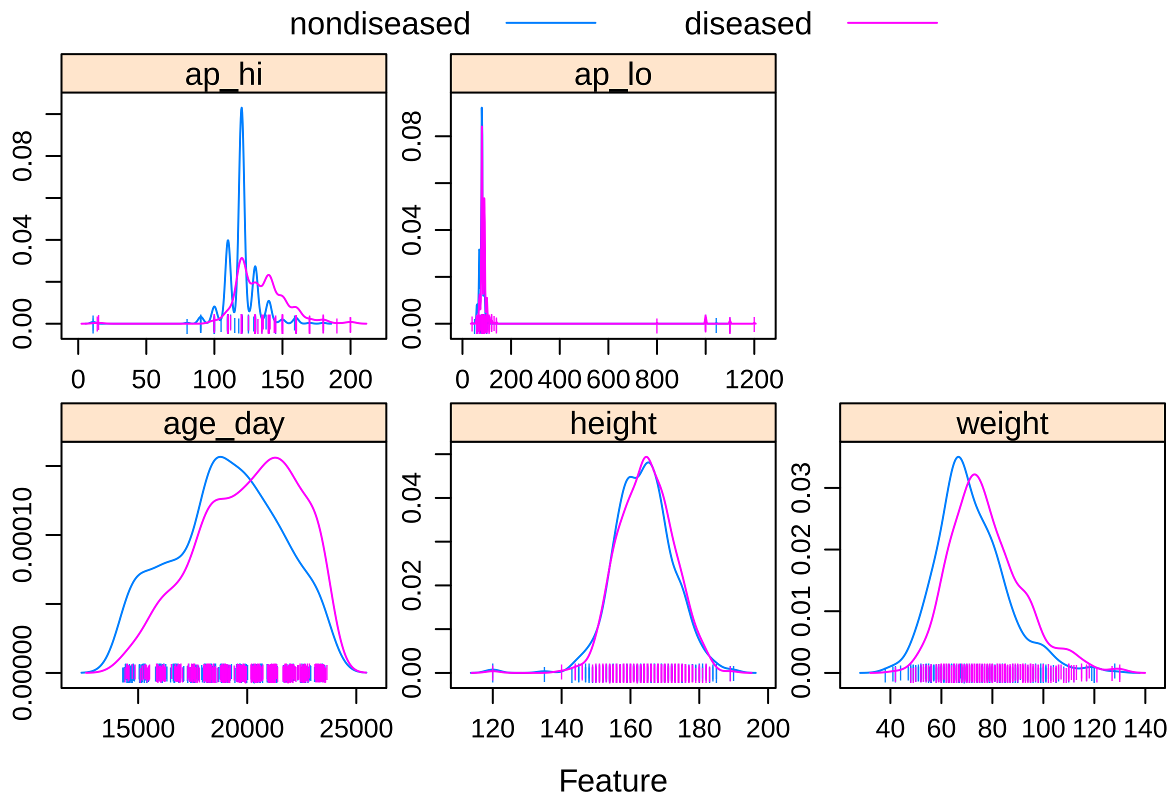
```r
#Summary
summary(df)
```

```
##     age_day          height          weight            ap_hi        
##  Min.   :10798   Min.   : 55.0   Min.   : 10.00   Min.   : -150.0  
##  1st Qu.:17664   1st Qu.:159.0   1st Qu.: 65.00   1st Qu.:  120.0  
##  Median :19703   Median :165.0   Median : 72.00   Median :  120.0  
##  Mean   :19469   Mean   :164.4   Mean   : 74.21   Mean   :  128.8  
##  3rd Qu.:21327   3rd Qu.:170.0   3rd Qu.: 82.00   3rd Qu.:  140.0  
##  Max.   :23713   Max.   :250.0   Max.   :200.00   Max.   :16020.0  
##      ap_lo          gender              cholesterol   
##  Min.   :  -70.00   0:45530   normal           :52385  
##  1st Qu.:   80.00   1:24470   above normal     : 9549  
##  Median :   80.00             well above normal: 8066  
##  Mean   :   96.63                                      
##  3rd Qu.:   90.00                                      
##  Max.   :11000.00                                      
##                 gluc       smoke     alco      active           cardio     
##  normal           :59479   0:63831   0:66236   0:13739   nondiseased:35021  
##  above normal     : 5190   1: 6169   1: 3764   1:56261   diseased   :34979  
##  well above normal: 5331                                                    
##                                                                             
##                                                                             
## 
```

```r
#Corr Plot
x_df = model.matrix(cardio ~ ., df)[,-1]
corrplot::corrplot(cor(x_df),
                   method = "circle",
                   type = "full",
                   tl.cex = 0.5)
```



```r
set.seed(2022)
# Randomly sample 3500 data points without replacement from the data set.
df_sample = sample_n(df, 1000) %>%
  janitor::clean_names()

#colnames(df_sample)[0] <- "id

#save R data
save(df_sample, file = "df_sample.RData")

#summary(df_sample)

load("df_sample.RData")

#Feature Plot
featurePlot(x = df_sample[, 1:5],
            y = df_sample$cardio,
            scales = list(x = list(relation="free"),
                          y = list(relation="free")),
            plot = "density",
            pch = "|",
            auto.key = list(columns = 2)
            )
```



```r
set.seed(2022)
training_tag = createDataPartition(y = df_sample$cardio,
                                   p = 0.7,
                                   list = FALSE)

# For training dataset
training_data = df_sample[training_tag, ]%>%janitor::clean_names()
training_predictors_x = model.matrix(cardio ~ ., training_data)[, -1]
training_outcome_y = training_data$cardio


# For test dataset
test_data = df_sample[-training_tag, ]%>%janitor::clean_names()
test_predictors_x = model.matrix(cardio ~ ., test_data)[, -1]
test_outcome_y = test_data$cardio

# Control
control = trainControl(method = "repeatedcv",
                       summaryFunction = twoClassSummary,
                       repeats = 5,
                       classProbs = TRUE)
```

1.logistic regression

```r
#undiseased: 0
#diseased: 1
contrasts(df_sample$cardio)
```

```
##             diseased
## nondiseased        0
## diseased           1
```

```r
set.seed(2022)
glm_fit = glm(cardio ~ .,
              data = df_sample,
              subset = training_tag,
              family = binomial(link = "logit")
              )
summary(glm_fit)
```

```
##
## Call:
## glm(formula = cardio ~ ., family = binomial(link = "logit"),
##     data = df_sample, subset = training_tag)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5713  -0.9408   0.0311   0.9972   3.2204
##
## Coefficients:
##                               Estimate Std. Error z value Pr(>|z|)
## (Intercept)                 -9.663e+00  2.236e+00  -4.322 1.54e-05 ***
## age_day                      1.501e-04  3.707e-05   4.049 5.13e-05 ***
## height                      -5.518e-03  1.213e-02  -0.455  0.64931
## weight                       1.377e-02  6.885e-03   2.000  0.04555 *
## ap_hi                        5.216e-02  6.995e-03   7.457 8.86e-14 ***
## ap_lo                        1.214e-03  9.387e-04   1.293  0.19606
## gender1                      1.335e-01  2.127e-01   0.627  0.53036
## cholesterolabove normal      1.997e-01  2.708e-01   0.738  0.46081
## cholesterolwell above normal  8.861e-01  3.318e-01   2.671  0.00757 **
## glucabove normal            -2.142e-02  3.362e-01  -0.064  0.94919
## glucwell above normal        9.407e-02  3.552e-01   0.265  0.79112
## smoke1                      -4.787e-01  3.356e-01  -1.426  0.15376
## alco1                       -2.794e-01  4.308e-01  -0.648  0.51670
## active1                     -2.280e-01  2.058e-01  -1.108  0.26785
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 970.41  on 699  degrees of freedom
## Residual deviance: 804.40  on 686  degrees of freedom
## AIC: 832.4
##
## Number of Fisher Scoring iterations: 5
```

```r
test_pred_prob = predict(glm_fit, newdata = test_data,
                         type = "response"
                         )

test_pred = rep("nondiseased", length(test_pred_prob))

test_pred[test_pred_prob > 0.5] = "diseased"

confusionMatrix(data = as.factor(test_pred),
                reference = test_outcome_y
                )
```

```
## Warning in confusionMatrix.default(data = as.factor(test_pred), reference
## = test_outcome_y): Levels are not in the same order for reference and data.
## Refactoring data to match.

## Confusion Matrix and Statistics
##
##              Reference
## Prediction    nondiseased diseased
##   nondiseased         111       34
##   diseased             39      116
##
##                Accuracy : 0.7567
##                  95% CI : (0.704, 0.8041)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.5133
##
##  Mcnemar's Test P-Value : 0.6397
##
##             Sensitivity : 0.7400
##             Specificity : 0.7733
##          Pos Pred Value : 0.7655
##          Neg Pred Value : 0.7484
##              Prevalence : 0.5000
##          Detection Rate : 0.3700
##    Detection Prevalence : 0.4833
##       Balanced Accuracy : 0.7567
##
##        'Positive' Class : nondiseased
##
```

```r
auc(test_outcome_y, test_pred_prob)
```

```
## Setting levels: control = nondiseased, case = diseased

## Setting direction: controls < cases

## Area under the curve: 0.8053
```

```
#caret logistic for model selection
set.seed(2022)
logistic_caret = train(x = training_predictors_x,
                       y = training_outcome_y,
                       method = "glm",
                       metric = "ROC",
                       trControl = control
                       )
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(logistic_caret)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5713  -0.9408   0.0311   0.9972   3.2204
##
## Coefficients:
##                                Estimate Std. Error z value Pr(>|z|)
## (Intercept)                  -9.663e+00  2.236e+00  -4.322 1.54e-05 ***
## age_day                       1.501e-04  3.707e-05   4.049 5.13e-05 ***
## height                       -5.518e-03  1.213e-02  -0.455  0.64931
## weight                        1.377e-02  6.885e-03   2.000  0.04555 *
## ap_hi                         5.216e-02  6.995e-03   7.457 8.86e-14 ***
## ap_lo                         1.214e-03  9.387e-04   1.293  0.19606
## gender1                       1.335e-01  2.127e-01   0.627  0.53036
## `cholesterolabove normal`     1.997e-01  2.708e-01   0.738  0.46081
## `cholesterolwell above normal` 8.861e-01  3.318e-01   2.671  0.00757 **
## `glucabove normal`           -2.142e-02  3.362e-01  -0.064  0.94919
## `glucwell above normal`       9.407e-02  3.552e-01   0.265  0.79112
## smoke1                       -4.787e-01  3.356e-01  -1.426  0.15376
## alco1                        -2.794e-01  4.308e-01  -0.648  0.51670
## active1                      -2.280e-01  2.058e-01  -1.108  0.26785
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 970.41  on 699  degrees of freedom
## Residual deviance: 804.40  on 686  degrees of freedom
## AIC: 832.4
##
## Number of Fisher Scoring iterations: 5
```

2. MARS

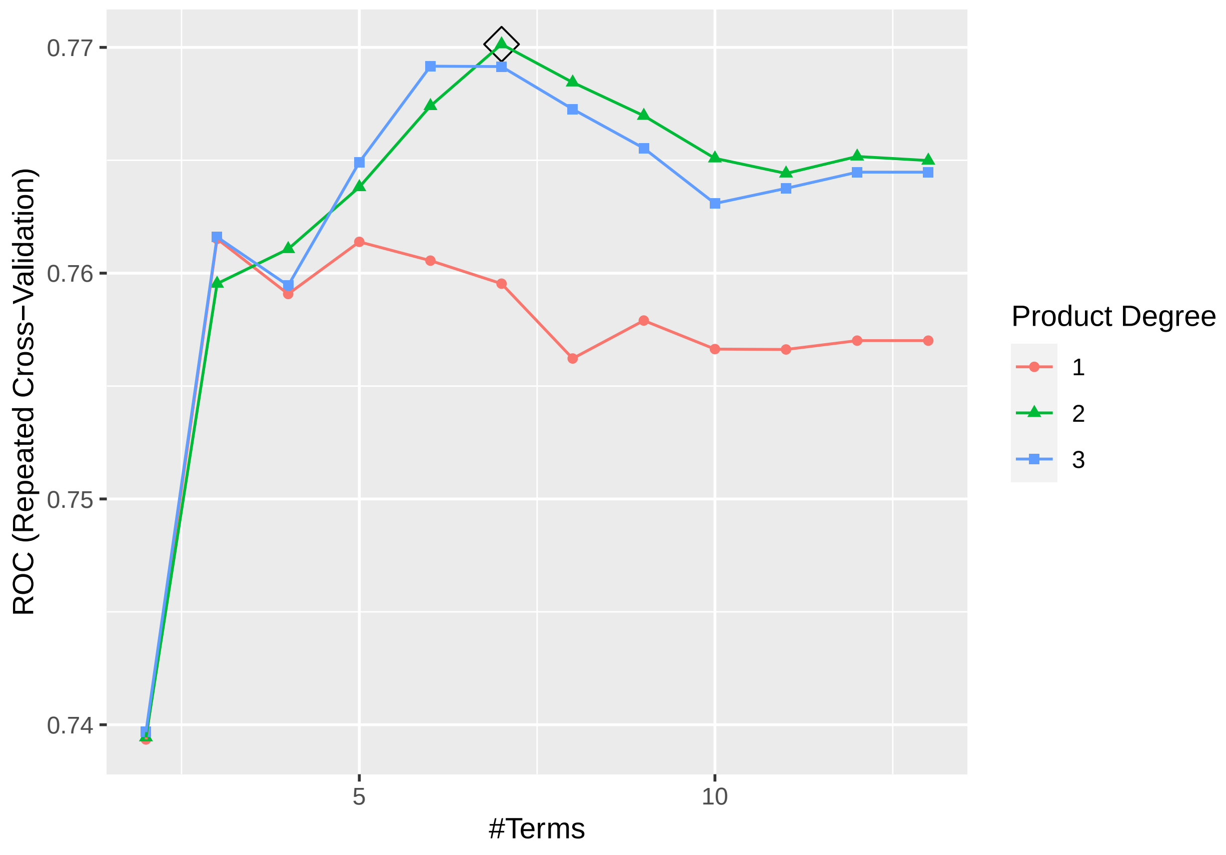
```r
#adjust cardio to be dummy

set.seed(2022)
mars_model = train(x = training_predictors_x,
                   y = training_outcome_y,
                   method = "earth",
                   tuneGrid = expand.grid(degree = 1:3, nprune = 2:13),
                   metric = "ROC",
                   trControl = control
                   )
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(mars_model)
```

```
## Call: earth(x=matrix[700,13], y=factor.object, keepxy=TRUE,
##             glm=list(family=function.object, maxit=100), degree=2, nprune=7)
##
## GLM coefficients
##                                                    diseased
## (Intercept)                                     -2.85083364
## cholesterolwell above normal                     2.29450622
## h(16739-age_day)                                -0.00034732
## h(ap_hi-100)                                     0.09199229
## h(ap_hi-100) * cholesterolwell above normal -0.05256094
## h(age_day-16739) * h(ap_hi-150)                 -0.00000281
## h(age_day-16739) * h(150-ap_hi)                  0.00000620
##
## GLM (family binomial, link logit):
##  nulldev  df       dev  df   devratio     AIC iters converged
##  970.406 699   778.558 693      0.198   792.6     5         1
##
## Earth selected 7 of 21 terms, and 3 of 13 predictors (nprune=7)
## Termination condition: Reached nk 27
## Importance: ap_hi, age_day, cholesterolwell above normal, height-unused, ...
## Number of terms at each degree of interaction: 1 3 3
## Earth GCV 0.1996215    RSS 133.4202    GRSq 0.2037937    RSq 0.237599
```

```
ggplot(mars_model, highlight = T)
```



```
mars_model$bestTune
```

```
##    nprune degree
## 18      7      2
```

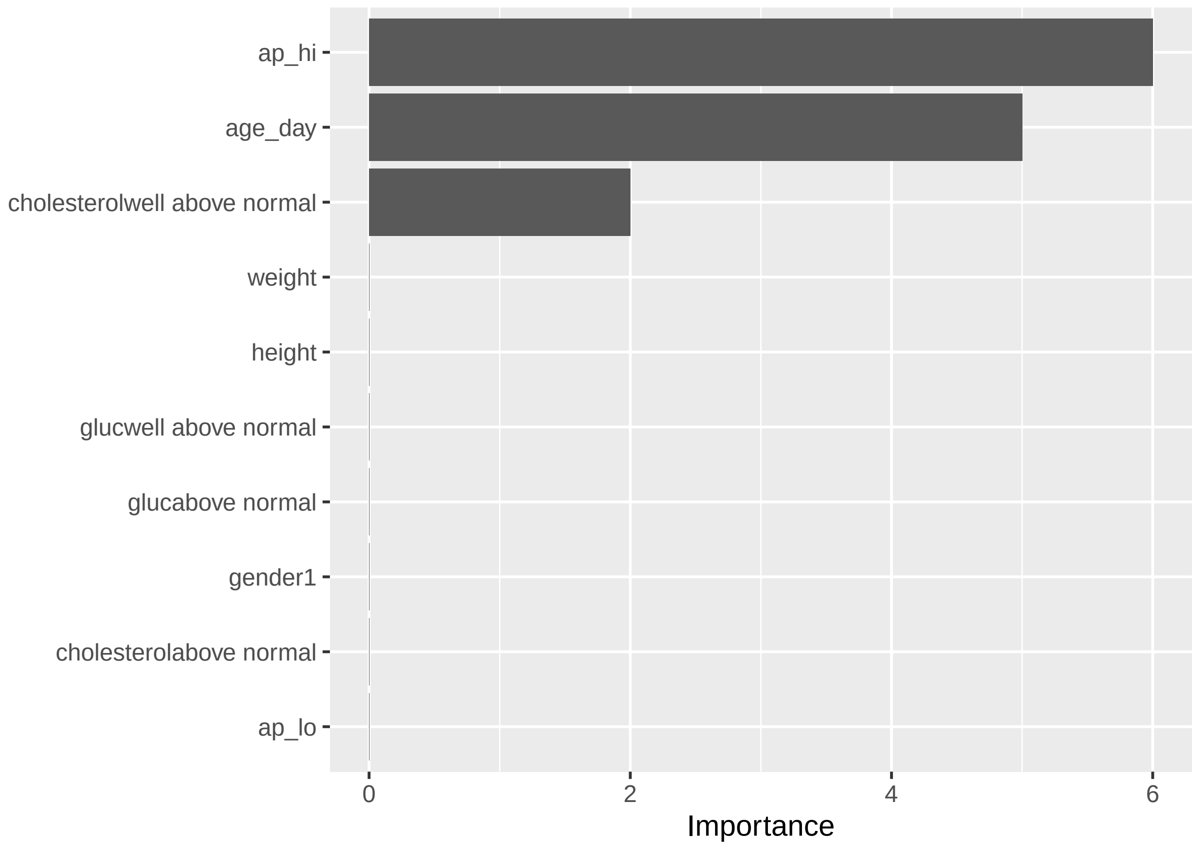
```
mars_model$results
```

```
##    degree nprune       ROC      Sens      Spec      ROCSD     SensSD     SpecSD
## 1       1      2 0.7393469 0.7800000 0.5971429 0.05727936 0.07982697 0.08909747
## 13      2      2 0.7394449 0.7800000 0.5971429 0.05720819 0.07982697 0.08909747
## 25      3      2 0.7396816 0.7800000 0.5971429 0.05739157 0.07982697 0.08909747
## 2       1      3 0.7615184 0.7554286 0.6342857 0.05905114 0.08870581 0.09537405
## 14      2      3 0.7595347 0.7285714 0.6571429 0.05778705 0.10740977 0.09361098
## 26      3      3 0.7616000 0.7222857 0.6640000 0.06089484 0.09881261 0.09788433
## 3       1      4 0.7590776 0.7702857 0.6200000 0.06109256 0.09052763 0.11504848
## 15      2      4 0.7610694 0.7177143 0.6668571 0.05688798 0.09460939 0.09389707
## 27      3      4 0.7594531 0.7194286 0.6531429 0.06204869 0.09531989 0.09641815
## 4       1      5 0.7613878 0.7720000 0.6188571 0.05886146 0.08820294 0.11379047
## 16      2      5 0.7638122 0.7240000 0.6548571 0.05750386 0.08727634 0.09499600
## 28      3      5 0.7649143 0.7251429 0.6588571 0.06261853 0.09215636 0.09904332
## 5       1      6 0.7605551 0.7760000 0.6137143 0.05871067 0.08836711 0.09799999
## 17      2      6 0.7674041 0.7285714 0.6640000 0.06010477 0.08468773 0.08649980
## 29      3      6 0.7691673 0.7285714 0.6594286 0.06146664 0.08449078 0.10483362
## 6       1      7 0.7595347 0.7702857 0.6154286 0.05501890 0.08638417 0.09782474
## 18      2      7 0.7701388 0.7302857 0.6640000 0.05934579 0.07875753 0.09317573
## 30      3      7 0.7691510 0.7251429 0.6611429 0.06378316 0.08579005 0.10114052
## 7       1      8 0.7562204 0.7708571 0.6165714 0.05571805 0.08173259 0.09982742
## 19      2      8 0.7684490 0.7314286 0.6594286 0.05427342 0.08224262 0.08976623
## 31      3      8 0.7672653 0.7274286 0.6605714 0.06017651 0.08349109 0.09566010
## 8       1      9 0.7579020 0.7720000 0.6205714 0.05795684 0.08571234 0.09998584
## 20      2      9 0.7669714 0.7348571 0.6588571 0.05782721 0.08385545 0.09042636
## 32      3      9 0.7655347 0.7251429 0.6582857 0.06142293 0.07954263 0.09846130
## 9       1     10 0.7566367 0.7725714 0.6234286 0.05599165 0.08162449 0.09858474
## 21      2     10 0.7650857 0.7331429 0.6594286 0.05806574 0.08337128 0.09482046
## 33      3     10 0.7630857 0.7234286 0.6582857 0.06324309 0.07775269 0.09997251
## 10      1     11 0.7566204 0.7708571 0.6240000 0.05497303 0.08374611 0.09699526
## 22      2     11 0.7644163 0.7337143 0.6588571 0.05822727 0.08472510 0.09474488
## 34      3     11 0.7637551 0.7251429 0.6582857 0.06189125 0.07501548 0.09863035
## 11      1     12 0.7570122 0.7708571 0.6240000 0.05451333 0.08374611 0.09699526
## 23      2     12 0.7651673 0.7354286 0.6588571 0.05762884 0.08538518 0.09474488
## 35      3     12 0.7644735 0.7274286 0.6588571 0.06146464 0.07552896 0.09751771
## 12      1     13 0.7570122 0.7708571 0.6240000 0.05451333 0.08374611 0.09699526
## 24      2     13 0.7649878 0.7337143 0.6594286 0.05778366 0.08781488 0.09517121
## 36      3     13 0.7644735 0.7274286 0.6588571 0.06146464 0.07552896 0.09751771
```

```
coef(mars_model$finalModel)
```

```
##                                    (Intercept)
##                                  -2.850834e+00
##                                   h(ap_hi-100)
##                                   9.199229e-02
##                              h(16739-age_day)
##                                 -3.473246e-04
##             h(age_day-16739) * h(ap_hi-150)
##                                 -2.808598e-06
##             h(age_day-16739) * h(150-ap_hi)
##                                  6.204302e-06
##                cholesterolwell above normal
##                                  2.294506e+00
## h(ap_hi-100) * cholesterolwell above normal
##                                 -5.256094e-02
```

```
vip(mars_model$finalModel)
```



```
mars_test_pred_prob_df = predict(mars_model, newdata = test_predictors_x,
                          type = "prob"
                          )

mars_test_pred_prob = mars_test_pred_prob_df$diseased

mars_test_pred = rep("nondiseased", length(mars_test_pred_prob))

mars_test_pred[mars_test_pred_prob > 0.5] = "diseased"

confusionMatrix(data = as.factor(mars_test_pred),
                reference = test_outcome_y
                )
```

```
## Warning in confusionMatrix.default(data = as.factor(mars_test_pred), reference
## = test_outcome_y): Levels are not in the same order for reference and data.
## Refactoring data to match.
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction    nondiseased diseased
##   nondiseased         109       31
##   diseased             41      119
##
##                Accuracy : 0.76
##                  95% CI : (0.7076, 0.8072)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.52
##
##  Mcnemar's Test P-Value : 0.2888
##
##             Sensitivity : 0.7267
##             Specificity : 0.7933
##          Pos Pred Value : 0.7786
##          Neg Pred Value : 0.7438
##              Prevalence : 0.5000
##          Detection Rate : 0.3633
##    Detection Prevalence : 0.4667
##       Balanced Accuracy : 0.7600
##
##        'Positive' Class : nondiseased
##
```

```
auc(test_outcome_y, mars_test_pred_prob)
```

```
## Area under the curve: 0.8135
```

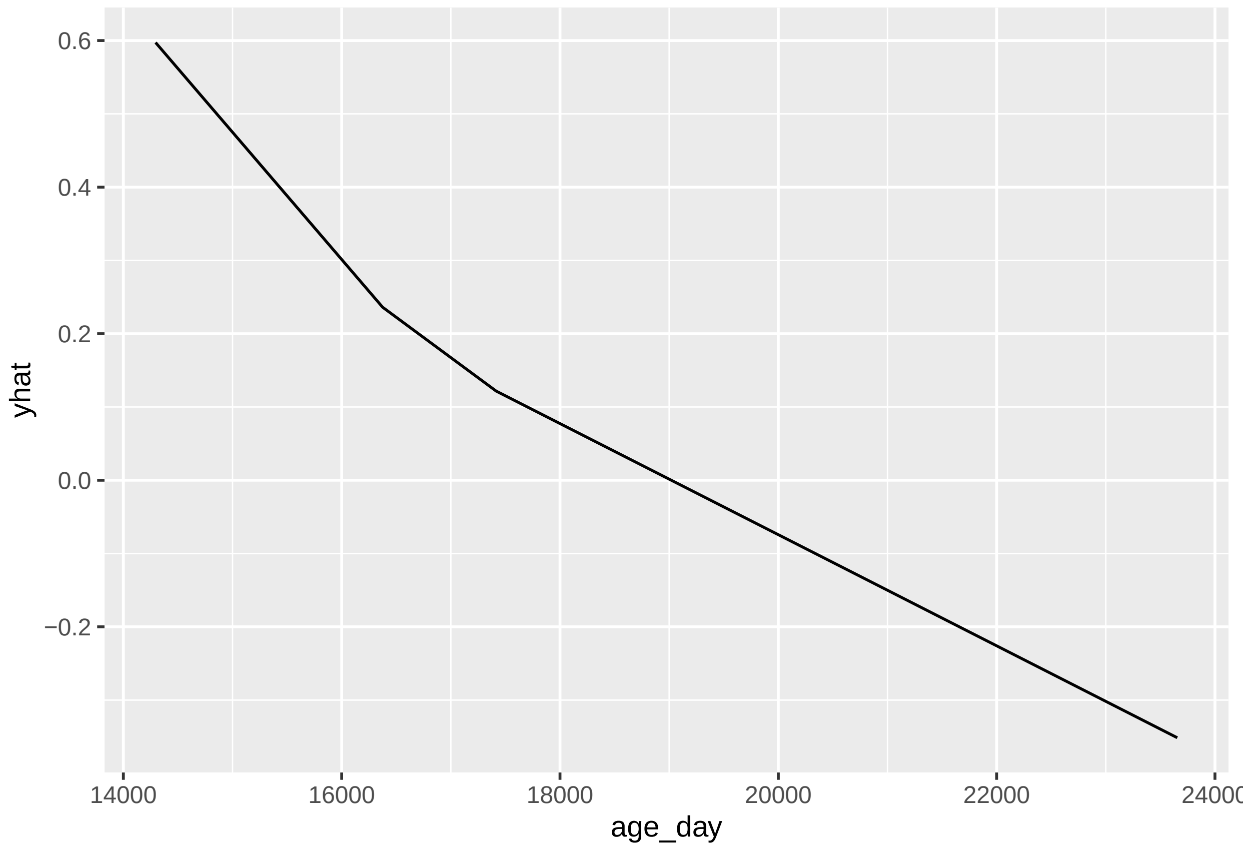
```
coef(mars_model$finalModel)
```

```
##                               (Intercept)
##                             -2.850834e+00
##                              h(ap_hi-100)
##                              9.199229e-02
##                          h(16739-age_day)
##                             -3.473246e-04
##              h(age_day-16739) * h(ap_hi-150)
##                             -2.808598e-06
##              h(age_day-16739) * h(150-ap_hi)
##                              6.204302e-06
##                 cholesterolwell above normal
##                              2.294506e+00
## h(ap_hi-100) * cholesterolwell above normal
##                             -5.256094e-02
```

```
p1  = pdp::partial(mars_model, pred.var = c("age_day"), grid.resolution = 10) %>%
  autoplot()
p1
```

```
## Warning: Use of `object[[1L]]` is discouraged. Use `.data[[1L]]` instead.
```

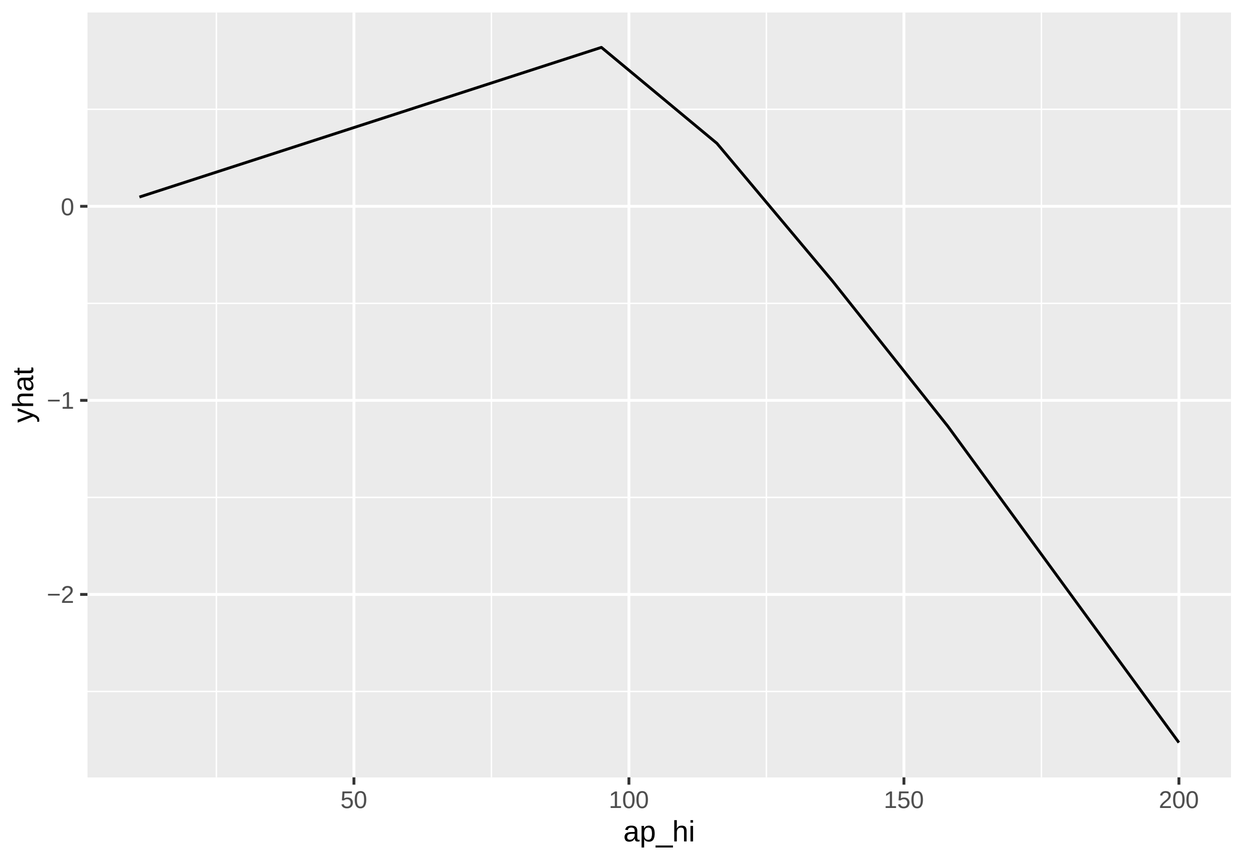
```
## Warning: Use of `object[["yhat"]]` is discouraged. Use `.data[["yhat"]]`
## instead.
```



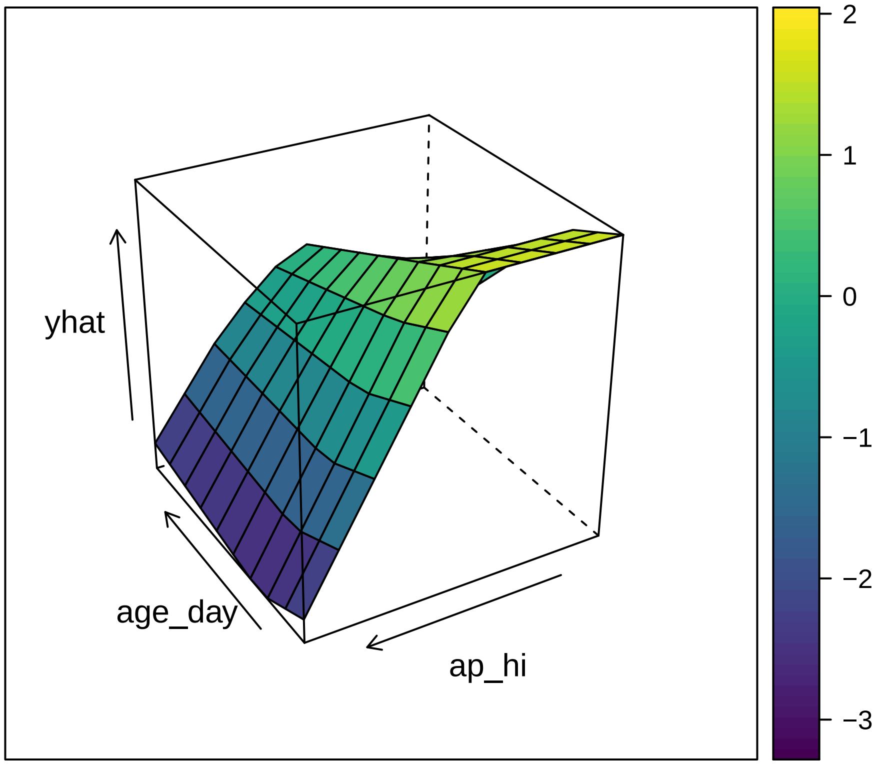
```
p2  = pdp::partial(mars_model, pred.var = c("ap_hi"), grid.resolution = 10) %>%
  autoplot()
p2
```

```
## Warning: Use of `object[[1L]]` is discouraged. Use `.data[[1L]]` instead.
```

```
## Warning: Use of `object[["yhat"]]` is discouraged. Use `.data[["yhat"]]`
## instead.
```



```r
p3 = pdp::partial(mars_model, pred.var = c("age_day", "ap_hi"), grid.resolution = 10) %>%
  pdp::plotPartial(levelplot = FALSE, zlab = "yhat", drape = TRUE, screen = list(z = 120, x = -60))
p3
```

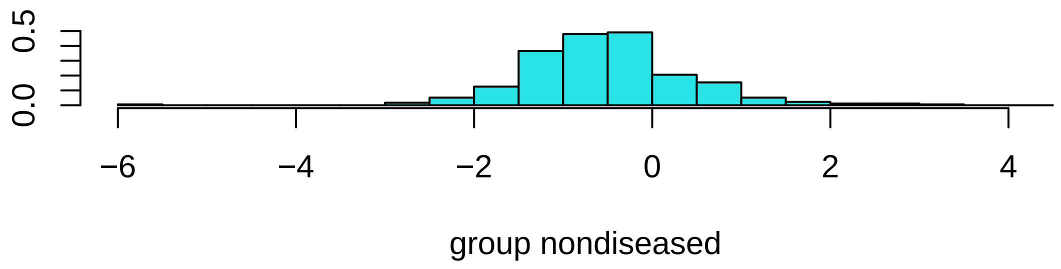


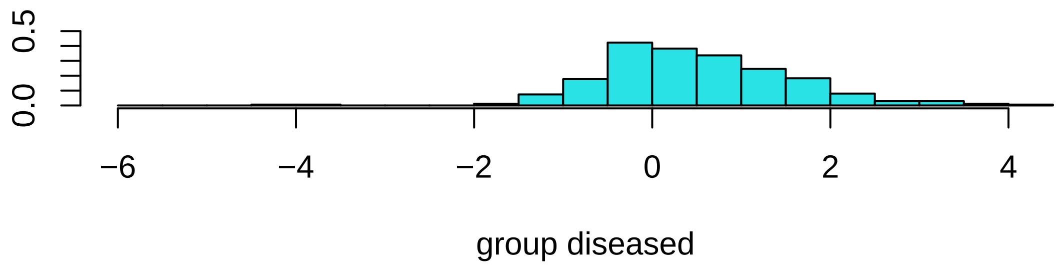
3. LDA

```
set.seed(2022)
lda_model = train(x = training_predictors_x,
                  y = training_outcome_y,
                  data = training_data,
                  method = "lda",
                  metric = "ROC",
                  trControl = control)
lda_model$results
```

```
##   parameter       ROC      Sens      Spec      ROCSD     SensSD    SpecSD
## 1      none 0.7595429 0.7525714 0.6308571 0.05843652 0.08175705 0.0930469
```

```
lda_fit = lda(cardio ~., data = df_sample, subset = training_tag)
plot(lda_fit)
```





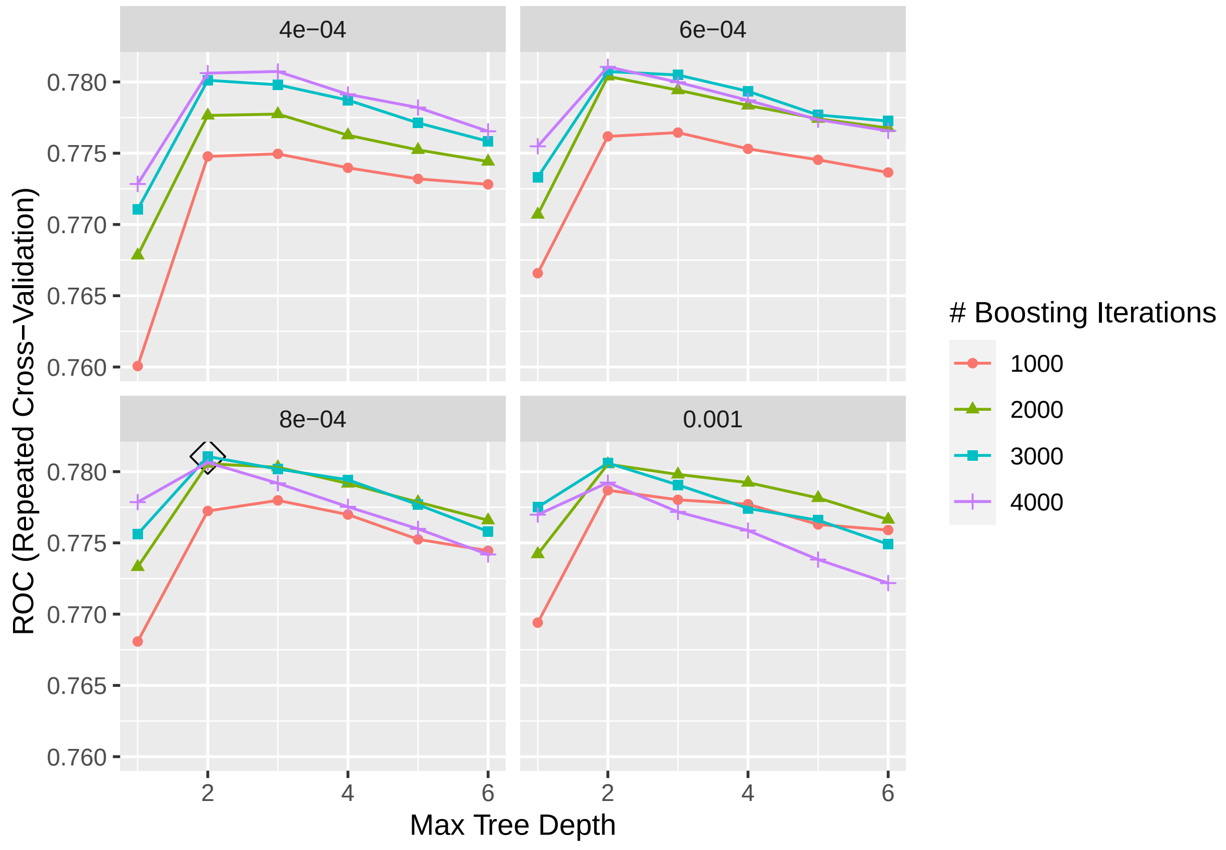
```
lda_fit$scaling
```

```
##                                        LD1
## age_day                       0.0001587505
## height                       -0.0063718174
## weight                        0.0150851142
## ap_hi                         0.0429264353
## ap_lo                         0.0008343669
## gender1                       0.1539603773
## cholesterolabove normal       0.2990711209
## cholesterolwell above normal  0.8731256286
## glucabove normal             -0.0395037032
## glucwell above normal         0.1133729473
## smoke1                       -0.4623471382
## alco1                        -0.3063815769
## active1                      -0.1654022160
```

4. Boosting

```
set.seed(2022)

boost_grid = expand.grid(n.trees = c(1000, 2000, 3000, 4000),
                            interaction.depth = 1:6,
                            shrinkage = c(0.0004, 0.0006, 0.0008, 0.001),
                            n.minobsinnode = 1)
```

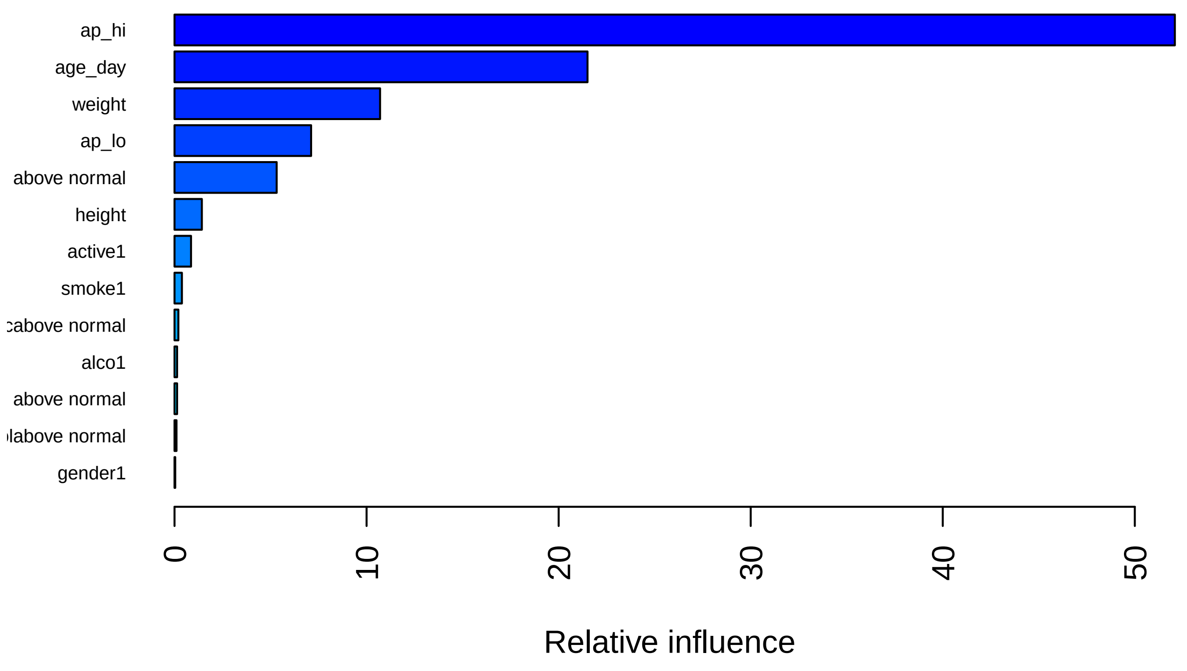
```
control_class = trainControl(method = "repeatedcv",
                             number = 10,
                             repeats = 5,
                             classProbs = TRUE,
                             summaryFunction = twoClassSummary
                             )

# Using caret perform boosting on the training data
boost_caret = train(cardio ~ .,
                    data = training_data,
                    method = "gbm",
                    tuneGrid = boost_grid,
                    trControl = control_class,
                    distribution = "adaboost",
                    metric = "ROC",
                    verbose = FALSE)

ggplot(boost_caret, highlight = TRUE)
```



```
boost_caret$bestTune
```

```
##    n.trees interaction.depth shrinkage n.minobsinnode
## 55    3000                 2     8e-04              1
```

```
# Plot the variable importance
summary(boost_caret$finalModel, las = 2, cBars = 19, cex.names = 0.6)
```

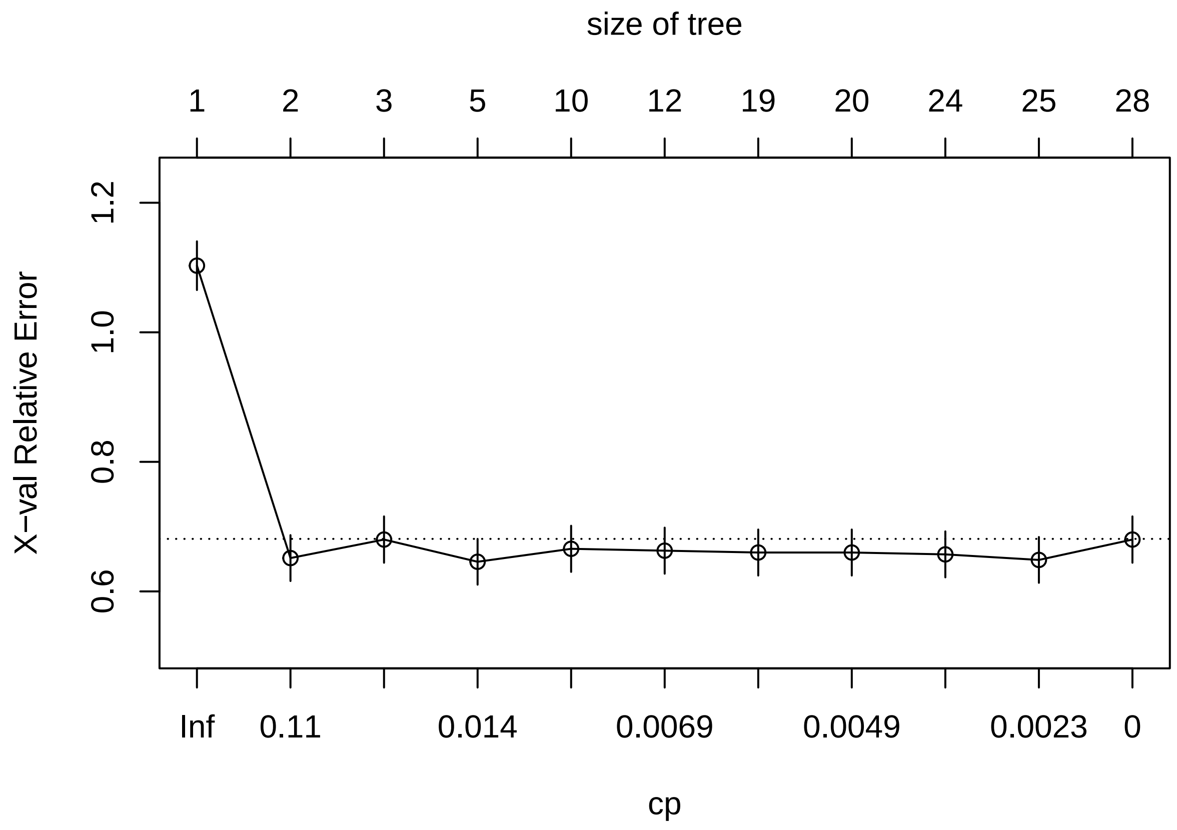


```
##                                                       var     rel.inf
## ap_hi                                               ap_hi 52.08437799
## age_day                                           age_day 21.50366721
## weight                                             weight 10.70296143
## ap_lo                                               ap_lo  7.11147104
## cholesterolwell above normal cholesterolwell above normal  5.31793794
## height                                             height  1.42389506
## active1                                           active1  0.85988309
## smoke1                                             smoke1  0.38443486
## glucabove normal                         glucabove normal  0.20070554
## alco1                                               alco1  0.13534147
## glucwell above normal                glucwell above normal  0.13509642
## cholesterolabove normal            cholesterolabove normal  0.10325443
## gender1                                           gender1  0.03697351
```

5. Classification Tree

```
set.seed(2022)
classification_tree_minMSE = rpart(formula = cardio ~ . ,
                                   data = training_data,
                                   control = rpart.control(cp = 0))
```

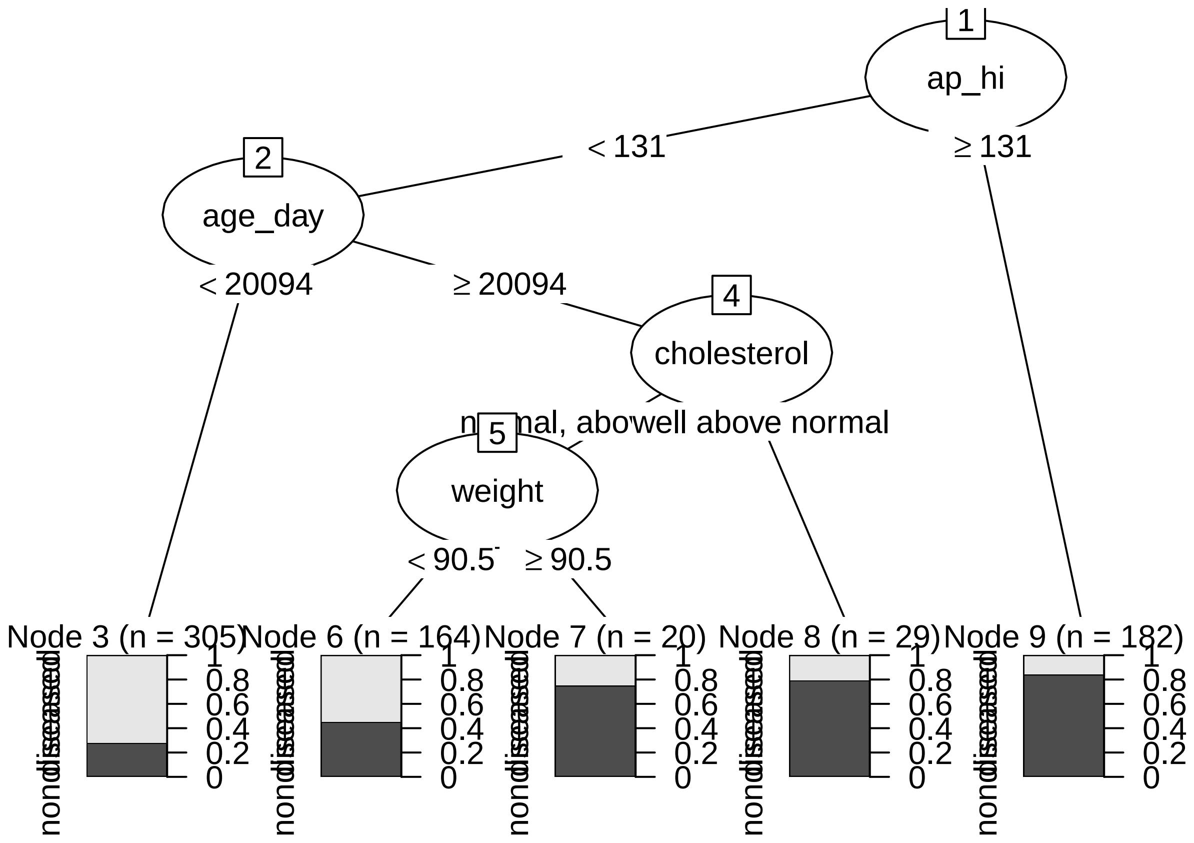
```r
plotcp(classification_tree_minMSE)
```



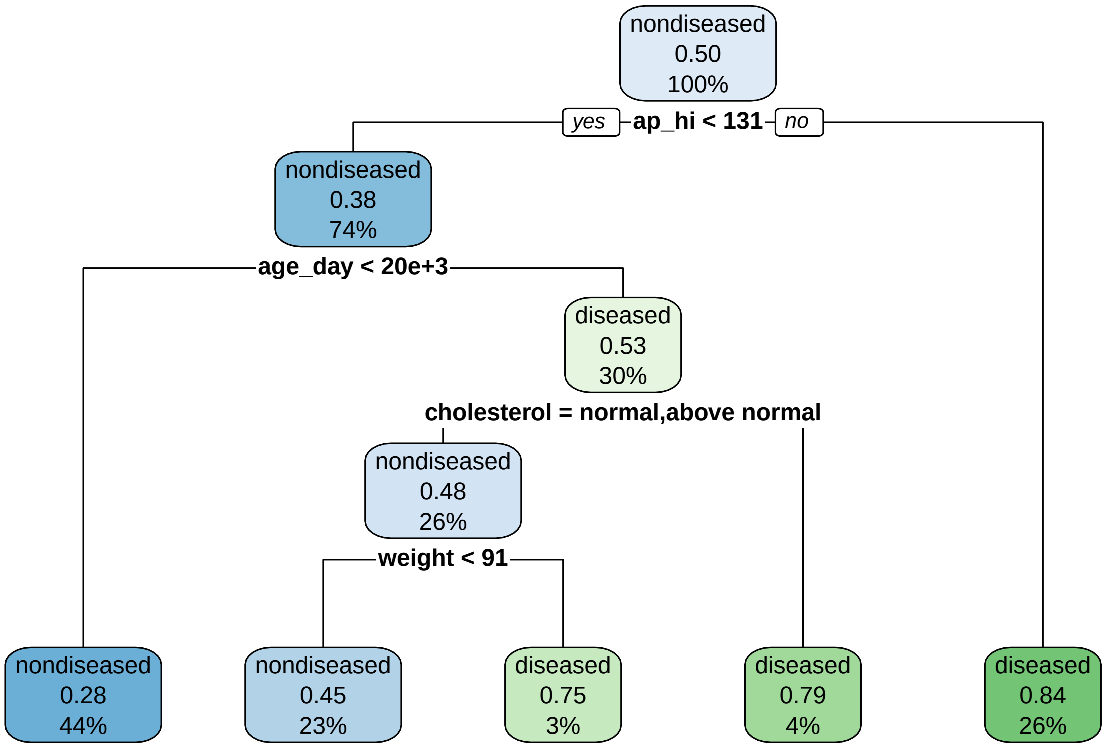
```r
# Obtain cp table
cp_table = printcp(classification_tree_minMSE)
```

```
##
## Classification tree:
## rpart(formula = cardio ~ ., data = training_data, control = rpart.control(cp = 0))
##
## Variables actually used in tree construction:
## [1] active      age_day     alco        ap_hi       ap_lo       cholesterol
## [7] height      weight
##
## Root node error: 350/700 = 0.5
##
## n= 700
##
##           CP nsplit rel error  xerror     xstd
## 1  0.3542857      0   1.00000 1.10286 0.037596
## 2  0.0314286      1   0.64571 0.65143 0.035426
## 3  0.0228571      2   0.61429 0.68000 0.035809
## 4  0.0085714      4   0.56857 0.64571 0.035345
## 5  0.0071429      9   0.52000 0.66571 0.035622
## 6  0.0066667     11   0.50571 0.66286 0.035584
## 7  0.0057143     18   0.45143 0.66000 0.035545
## 8  0.0042857     19   0.44571 0.66000 0.035545
## 9  0.0028571     23   0.42857 0.65714 0.035506
## 10 0.0019048     24   0.42571 0.64857 0.035386
## 11 0.0000000     27   0.42000 0.68000 0.035809
```

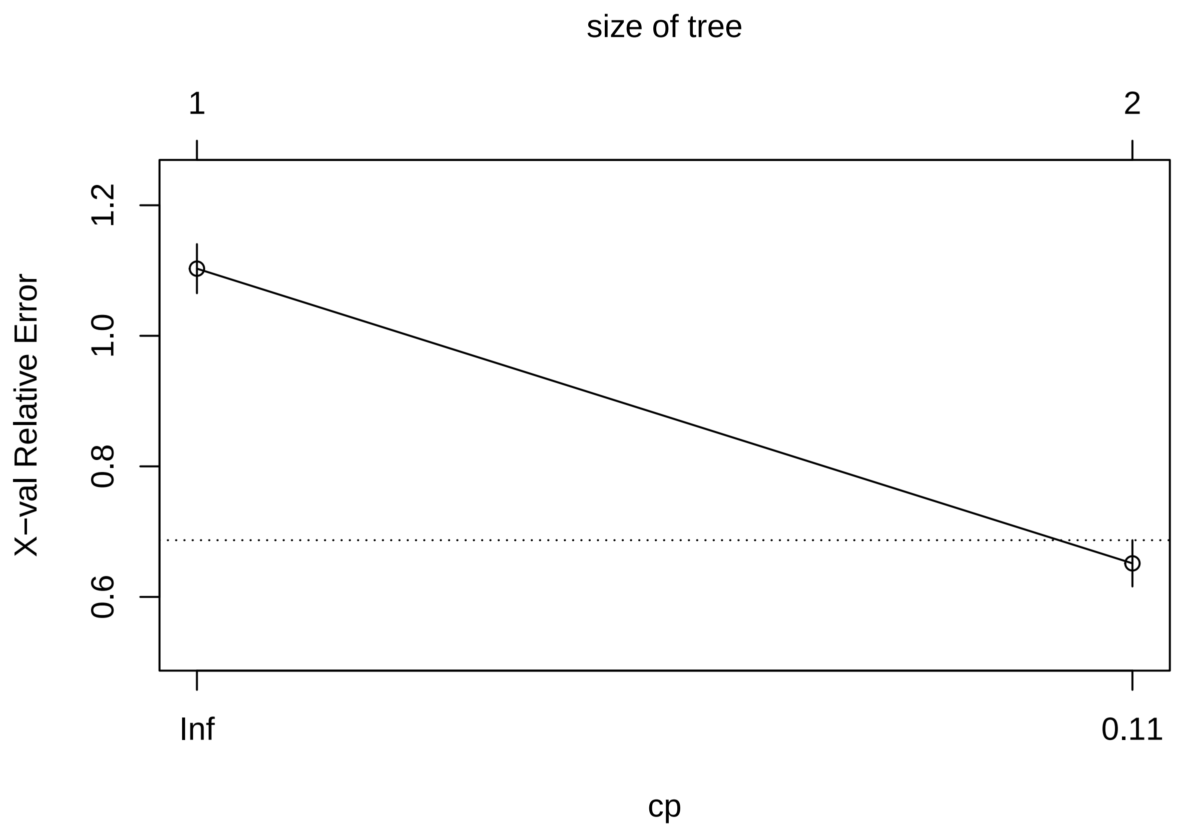
```
df_MSE_min = which.min(cp_table[, 4])
final_class_tree_minMSE = prune(classification_tree_minMSE, cp = cp_table[df_MSE_min, 1])

# plot the minimum MSE classification tree
plot(as.party(final_class_tree_minMSE))
```



```
rpart.plot(final_class_tree_minMSE)
```



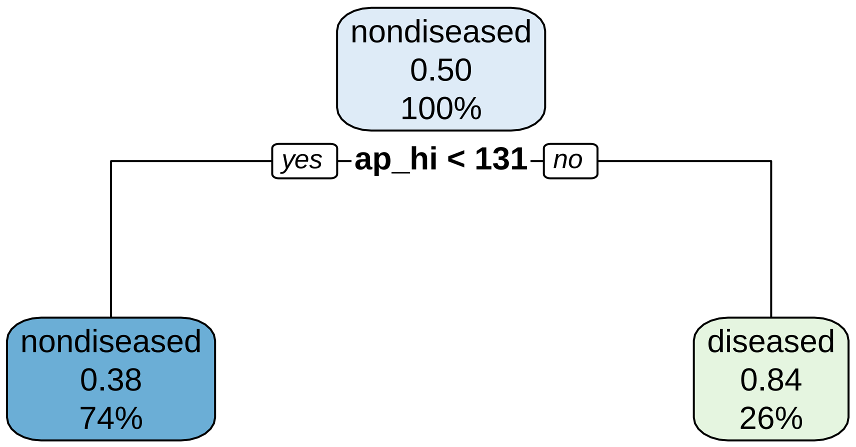
```
# Build classification tree using training dataset
classification_tree_1SE = prune(classification_tree_minMSE,
                                cp = cp_table[cp_table[, 4] < cp_table[df_MSE_min, 4] + cp_table[df_MSE_mi

plotcp(classification_tree_1SE)
```



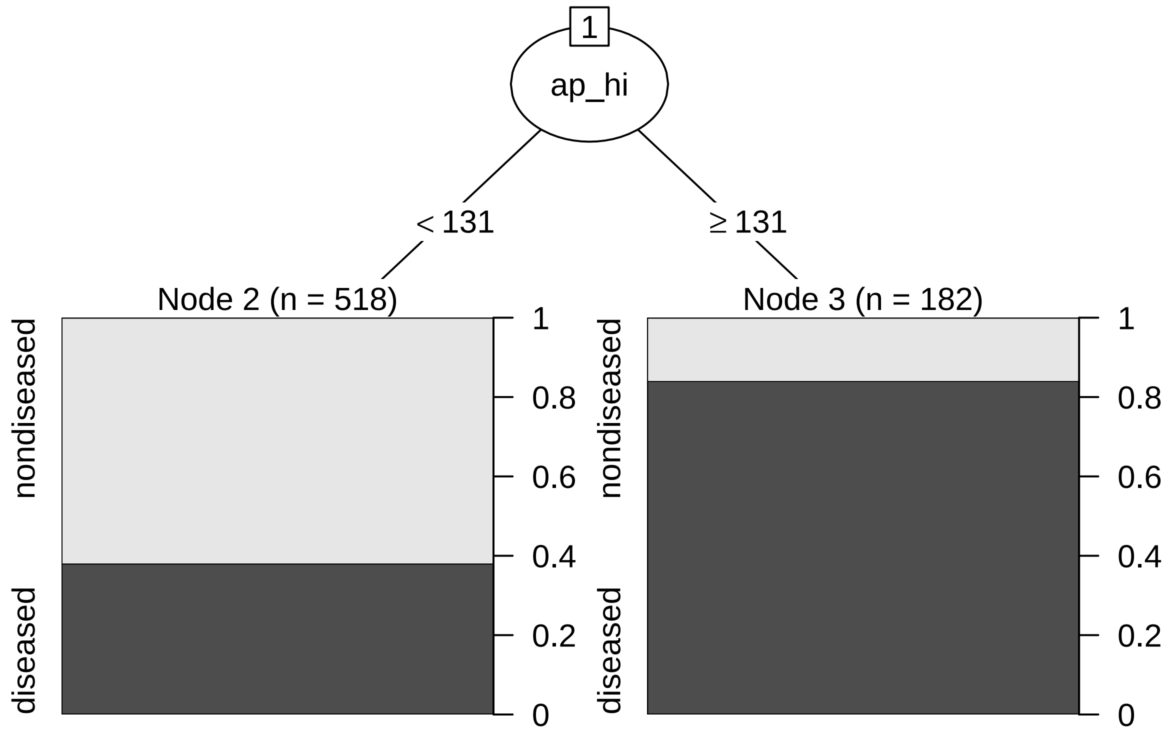
```r
# Obtain cp table
cp_table_1se = printcp(classification_tree_1SE)
```

```
##
## Classification tree:
## rpart(formula = cardio ~ ., data = training_data, control = rpart.control(cp = 0))
##
## Variables actually used in tree construction:
## [1] ap_hi
##
## Root node error: 350/700 = 0.5
##
## n= 700
##
##         CP nsplit rel error  xerror     xstd
## 1 0.354286      0   1.00000 1.10286 0.037596
## 2 0.031429      1   0.64571 0.65143 0.035426
```

```r
# Plot the 1SE tree
rpart.plot(classification_tree_1SE)
```



```
plot(as.party(classification_tree_1SE))
```



```
auc(test_outcome_y, predict(classification_tree_1SE, newdata = test_data)[, 2])
```

```
## Setting levels: control = nondiseased, case = diseased
```

```
## Setting direction: controls < cases
```

```
## Area under the curve: 0.7133
```

```
ct_test_pred_prob = predict(classification_tree_1SE, newdata = test_data)[, 2]

ct_test_pred = rep("nondiseased", length(ct_test_pred_prob))

ct_test_pred[ct_test_pred_prob > 0.5] = "diseased"

confusionMatrix(data = as.factor(ct_test_pred),
                reference = test_outcome_y
                )
```

```
## Warning in confusionMatrix.default(data = as.factor(ct_test_pred), reference
## = test_outcome_y): Levels are not in the same order for reference and data.
## Refactoring data to match.
```

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

```
##                Reference
## Prediction    nondiseased diseased
##   nondiseased         135       71
##   diseased             15       79
##
##                Accuracy : 0.7133
##                  95% CI : (0.6586, 0.7638)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 4.757e-14
##
##                   Kappa : 0.4267
##
##  Mcnemar's Test P-Value : 3.015e-09
##
##             Sensitivity : 0.9000
##             Specificity : 0.5267
##          Pos Pred Value : 0.6553
##          Neg Pred Value : 0.8404
##              Prevalence : 0.5000
##          Detection Rate : 0.4500
##    Detection Prevalence : 0.6867
##       Balanced Accuracy : 0.7133
##
##        'Positive' Class : nondiseased
##
```

Random forest

```
# Train caret random forest model
set.seed(2022)
# Grid of tuning parameters
rf_grid = expand.grid(mtry = 1:12,
                      splitrule = "gini",
                      min.node.size = seq(from = 2, to = 10, by = 2)
                      )

# Find best-fitting model after model fitting to optimize computational efficiency
rf_fit = train(cardio ~ .,
              data = training_data,
              method = "ranger",
              tuneGrid = rf_grid,
              metric = "ROC",
              trControl = control_class)

summary(rf_fit)
```

```
##                           Length Class         Mode
## predictions               1400   -none-        numeric
## num.trees                    1   -none-        numeric
## num.independent.variables    1   -none-        numeric
## mtry                         1   -none-        numeric
## min.node.size                1   -none-        numeric
## prediction.error            1   -none-        numeric
## forest                     10   ranger.forest list
## splitrule                   1   -none-        character
## treetype                    1   -none-        character
## call                        9   -none-        call
## importance.mode             1   -none-        character
## num.samples                 1   -none-        numeric
## replace                     1   -none-        logical
## xNames                     13   -none-        character
## problemType                 1   -none-        character
## tuneValue                   3   data.frame    list
## obsLevels                   2   -none-        character
## param                       0   -none-        list
```

```
rf_pred = predict(rf_fit, newdata = test_data, type = "prob")[,2]
#rf_pred


#ConfusionMatrix
test_pred_rf = rep("nondiseased", length(rf_pred ))

test_pred_rf[rf_pred > 0.5] = "diseased"

confusionMatrix(data = as.factor(test_pred_rf),
                reference = test_outcome_y
                )
```

```
## Warning in confusionMatrix.default(data = as.factor(test_pred_rf), reference
## = test_outcome_y): Levels are not in the same order for reference and data.
## Refactoring data to match.
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction    nondiseased diseased
##   nondiseased         116       37
##   diseased             34      113
##
##                Accuracy : 0.7633
##                  95% CI : (0.7111, 0.8103)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.5267
##
##  Mcnemar's Test P-Value : 0.8124
##
##             Sensitivity : 0.7733
##             Specificity : 0.7533
##          Pos Pred Value : 0.7582
##          Neg Pred Value : 0.7687
##              Prevalence : 0.5000
##          Detection Rate : 0.3867
##    Detection Prevalence : 0.5100
##       Balanced Accuracy : 0.7633
##
##        'Positive' Class : nondiseased
##
```

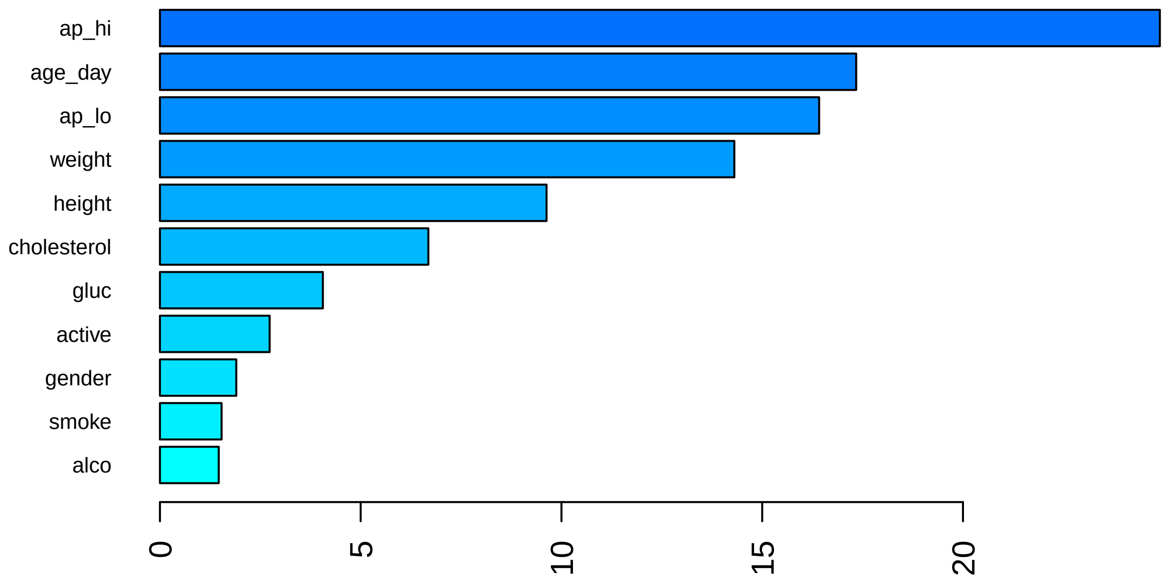
```
auc(test_outcome_y, rf_pred)
```

```
## Setting levels: control = nondiseased, case = diseased
```

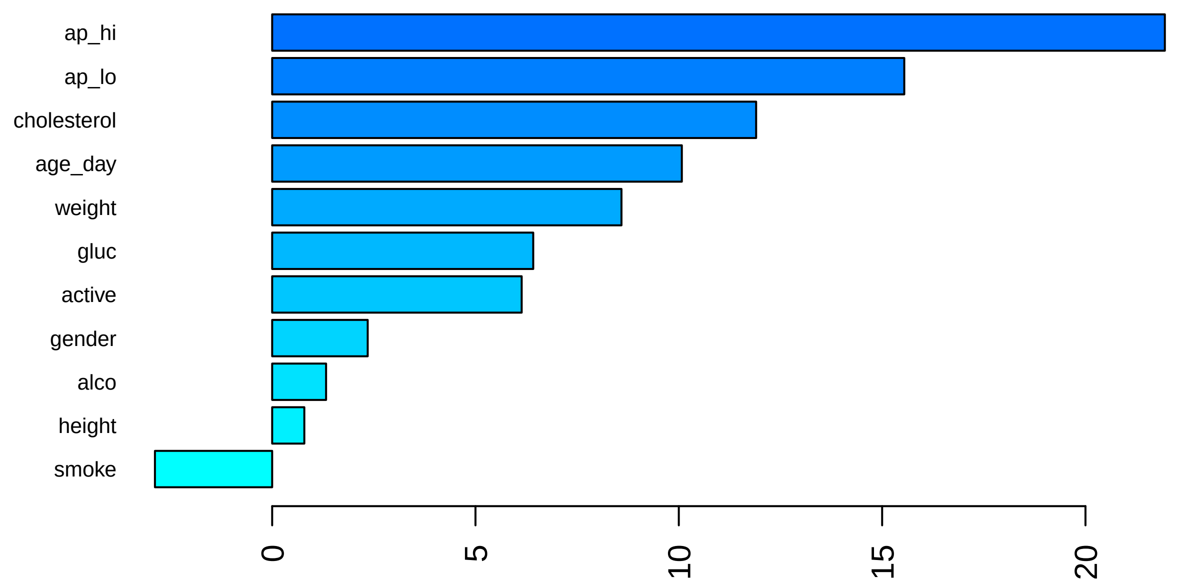
```
## Setting direction: controls < cases
```

```
## Area under the curve: 0.799
```

```
# Using impurity method to obtain variable importance
set.seed(2022)
rf_impurity_variable_importance = ranger(cardio ~ . ,
                             data = training_data,
                             mtry = rf_fit$bestTune[[1]],
                             splitrule = "gini",
                             min.node.size = rf_fit$bestTune[[3]],
                             importance = "impurity")


# plot of variable importance using impurity
barplot(sort(ranger::importance(rf_impurity_variable_importance),
             decreasing = FALSE),
        las = 2,
        horiz = TRUE,
        cex.names = 0.7,
        col = colorRampPalette(colors = c("cyan", "blue"))(19)
        )
```



```r
# Using permutation method to obtain variable importance
rf_permutation_variable_importance = ranger(cardio ~ . ,
                              data = training_data,
                              mtry = rf_fit$bestTune[[1]],
                              splitrule = "gini",
                              min.node.size = rf_fit$bestTune[[3]],
                              importance = "permutation",
                              scale.permutation.importance = TRUE)

# plot of variable importance using permutation
barplot(sort(ranger::importance(rf_permutation_variable_importance),
             decreasing = FALSE),
        las = 2,
        horiz = TRUE,
        cex.names = 0.7,
        col = colorRampPalette(colors = c("cyan", "blue"))(19)
        )
```

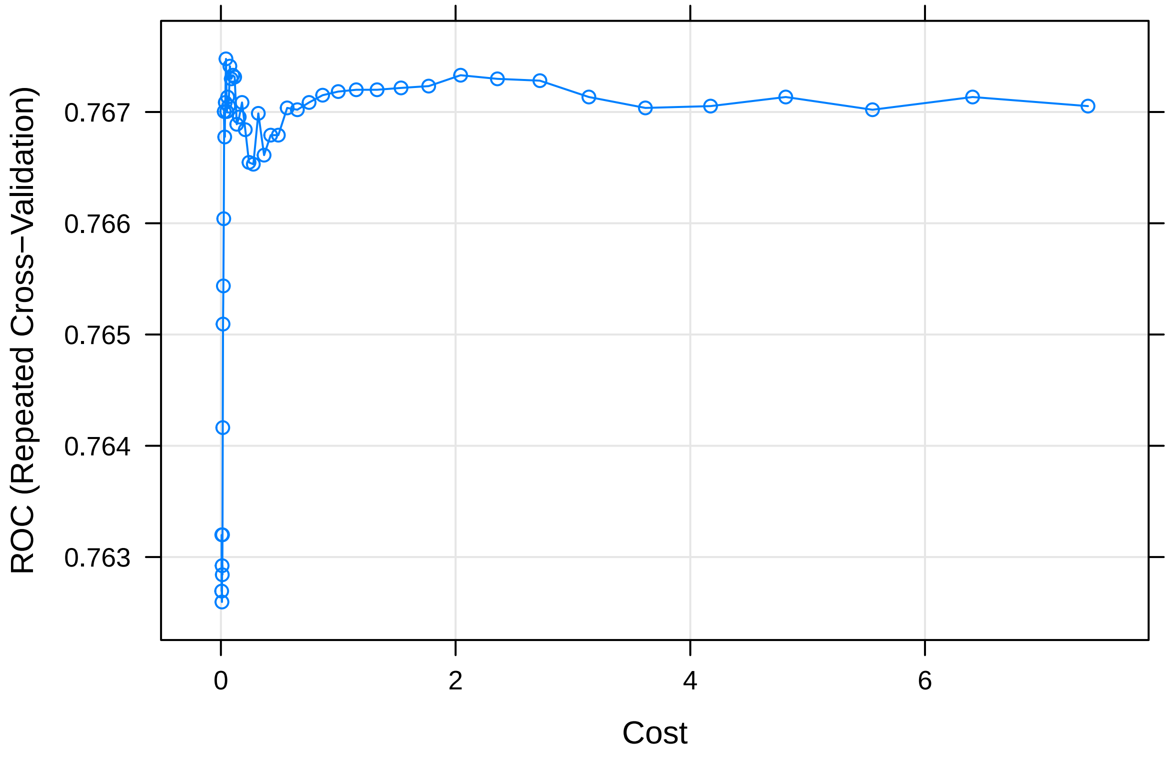


6. SVM

```
set.seed(2022)
linear_svc = train(cardio ~ .,
                      data = training_data,
                      method = "svmLinear",
                      tuneGrid = data.frame(C = exp(seq(-5, 2, len = 50))),
                      trControl = control_class,
                   scale = TRUE)
```

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

```
plot(linear_svc)
```



```r
linear_svc$bestTune
```

```
##             C
## 14 0.04315931
```

```r
svm_pred = predict(linear_svc, newdata = test_data, type = "prob")[, 2]

test_pred_svm = rep("nondiseased", length(svm_pred))

test_pred_svm[svm_pred > 0.5] = "diseased"

confusionMatrix(data = as.factor(test_pred_svm),
                reference = test_outcome_y)
```

```
## Warning in confusionMatrix.default(data = as.factor(test_pred_svm), reference
## = test_outcome_y): Levels are not in the same order for reference and data.
## Refactoring data to match.
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction    nondiseased diseased
##   nondiseased         116       36
##   diseased             34      114
```

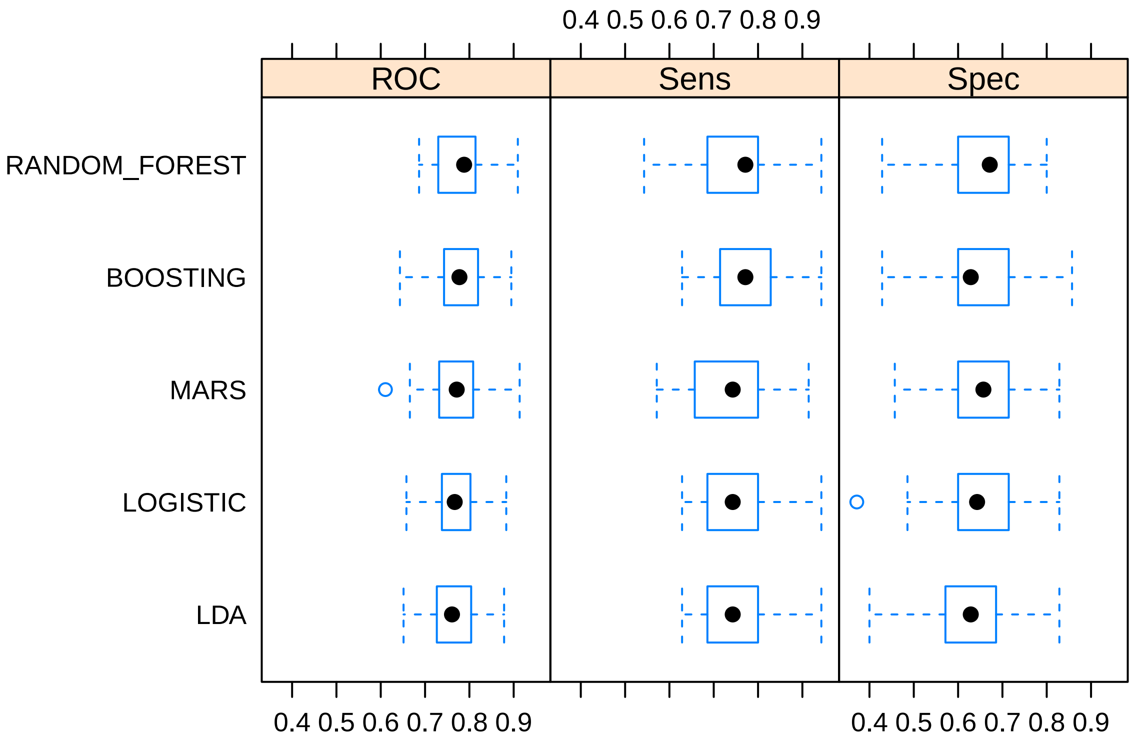
```
##
##                Accuracy : 0.7667
##                  95% CI : (0.7146, 0.8134)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.5333
##
##  Mcnemar's Test P-Value : 0.9049
##
##             Sensitivity : 0.7733
##             Specificity : 0.7600
##          Pos Pred Value : 0.7632
##          Neg Pred Value : 0.7703
##              Prevalence : 0.5000
##          Detection Rate : 0.3867
##    Detection Prevalence : 0.5067
##       Balanced Accuracy : 0.7667
##
##        'Positive' Class : nondiseased
##
```

Final Model Selection:

```
set.seed(2022)

resamp = resamples(list(MARS = mars_model,
                        LDA = lda_model,
                        LOGISTIC = logistic_caret,
                        BOOSTING = boost_caret,
                        RANDOM_FOREST =  rf_fit
                        #SVM = linear_svc
                        ))
summary(resamp)
```

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: MARS, LDA, LOGISTIC, BOOSTING, RANDOM_FOREST
## Number of resamples: 50
##
## ROC
##                    Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## MARS          0.6106122 0.7358163 0.7714286 0.7701388 0.8077551 0.9134694    0
## LDA           0.6514286 0.7289796 0.7608163 0.7595429 0.8022449 0.8783673    0
## LOGISTIC      0.6579592 0.7383673 0.7669388 0.7655347 0.8018367 0.8832653    0
## BOOSTING      0.6432653 0.7442857 0.7775510 0.7810612 0.8189796 0.8946939    0
## RANDOM_FOREST 0.6865306 0.7314286 0.7881633 0.7795755 0.8126531 0.9093878    0
##
## Sens
##                    Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## MARS          0.5714286 0.6571429 0.7428571 0.7302857 0.7928571 0.9142857    0
## LDA           0.6285714 0.6857143 0.7428571 0.7525714 0.8000000 0.9428571    0
## LOGISTIC      0.6285714 0.6928571 0.7428571 0.7571429 0.8000000 0.9428571    0
## BOOSTING      0.6285714 0.7214286 0.7714286 0.7714286 0.8285714 0.9428571    0
## RANDOM_FOREST 0.5428571 0.6928571 0.7714286 0.7582857 0.8000000 0.9428571    0
##
## Spec
##                    Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## MARS          0.4571429 0.6000000 0.6571429 0.6640000 0.7142857 0.8285714    0
## LDA           0.4000000 0.5714286 0.6285714 0.6308571 0.6857143 0.8285714    0
## LOGISTIC      0.3714286 0.6000000 0.6428571 0.6491429 0.7142857 0.8285714    0
## BOOSTING      0.4285714 0.6000000 0.6285714 0.6405714 0.7071429 0.8571429    0
## RANDOM_FOREST 0.4285714 0.6000000 0.6714286 0.6594286 0.7142857 0.8000000    0
```

```
bwplot(resamp)
```


ROC
Sens
Spec
RANDOM_FOREST
BOOSTING
MARS
LOGISTIC
LDA