P8106_group2recovery_secondaryanalysis

Yimin Chen (yc4195), Yang Yi (yy3307), Qingyue Zhuo (qz2493)

Contents

Import and data manipulation

```
# Load recovery.RData environment

load('./recovery.Rdata')

dat %>% na.omit()
```

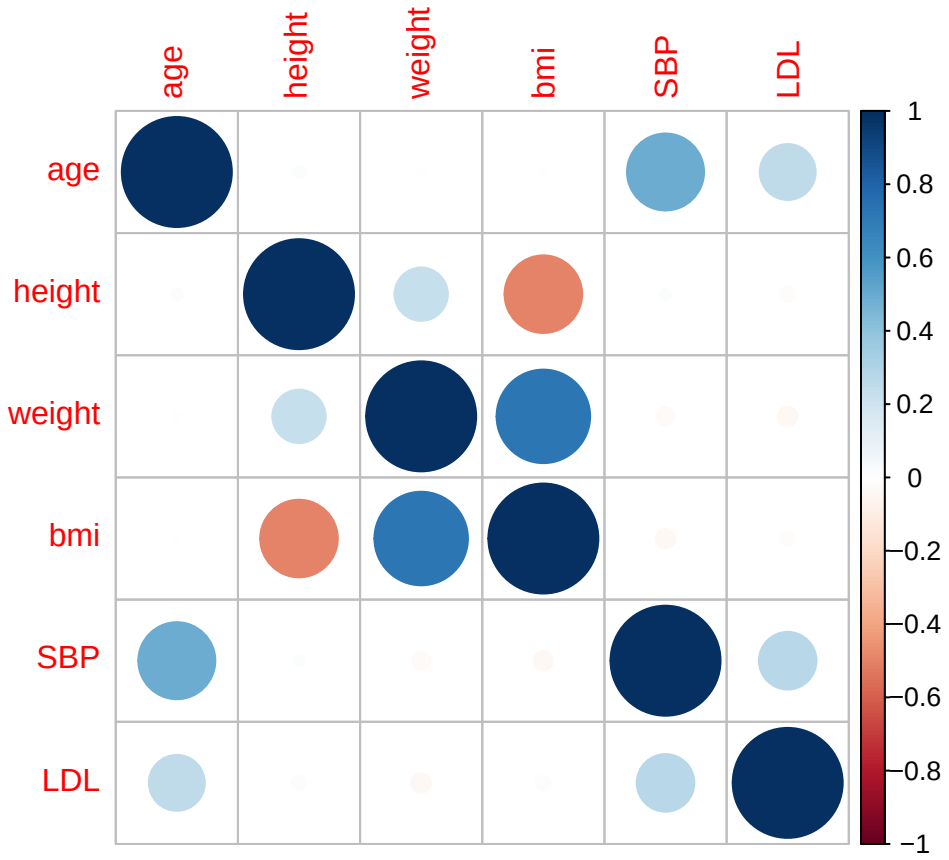
```r
# dat1 draw a random sample of 2000 participants Uni:3307
set.seed(3307)

dat1 = dat[sample(1:10000, 2000),]

dat1 =
  dat1[, -1] %>%
  mutate(
    recovery_time = as.factor(
      case_when(recovery_time <= 30 ~ "long", recovery_time > 30 ~ "short")
    ),
    gender = as.factor(gender),
    race = as.factor(race),
    smoking = as.factor(smoking),
    hypertension = as.factor(hypertension),
    diabetes = as.factor(diabetes),
    vaccine = as.factor(vaccine),
    severity = as.factor(severity),
    study = as.factor(
      case_when(study == "A" ~ 1, study == "B" ~ 2, study == "C" ~ 3)
      )
    )

# dat2 draw a random sample of 2000 participants Uni:2493
set.seed(2493)

dat2 = dat[sample(1:10000, 2000),]

dat2 =
  dat2[, -1] %>%
  mutate(
    recovery_time = as.factor(
      case_when(recovery_time <= 30 ~ "long", recovery_time > 30 ~ "short")
    ),
    gender = as.factor(gender),
    race = as.factor(race),
    smoking = as.factor(smoking),
    hypertension = as.factor(hypertension),
    diabetes = as.factor(diabetes),
    vaccine = as.factor(vaccine),
    severity = as.factor(severity),
    study = as.factor(
      case_when(study == "A" ~ 1, study == "B" ~ 2, study == "C" ~ 3)
      )
    )

# Merged dataset with unique observation
covid_dat = rbind(dat1, dat2) %>%
  unique()

covid_dat2 = model.matrix(recovery_time ~ ., covid_dat)[, -1]

# Partition dataset into two parts: training data (70%) and test data (30%)
```

```
rowTrain = createDataPartition(y = covid_dat$recovery_time, p = 0.7, list = FALSE)

trainData = covid_dat[rowTrain, ]
testData = covid_dat[-rowTrain, ]

ctrl = trainControl(method = "cv", number = 10)
ctrl1 = trainControl(method = "repeatedcv", number = 10, repeats = 5)
ctrl2 = trainControl(method = "cv",
                           classProbs = TRUE,
                           summaryFunction = twoClassSummary)
```

1. Data visualization

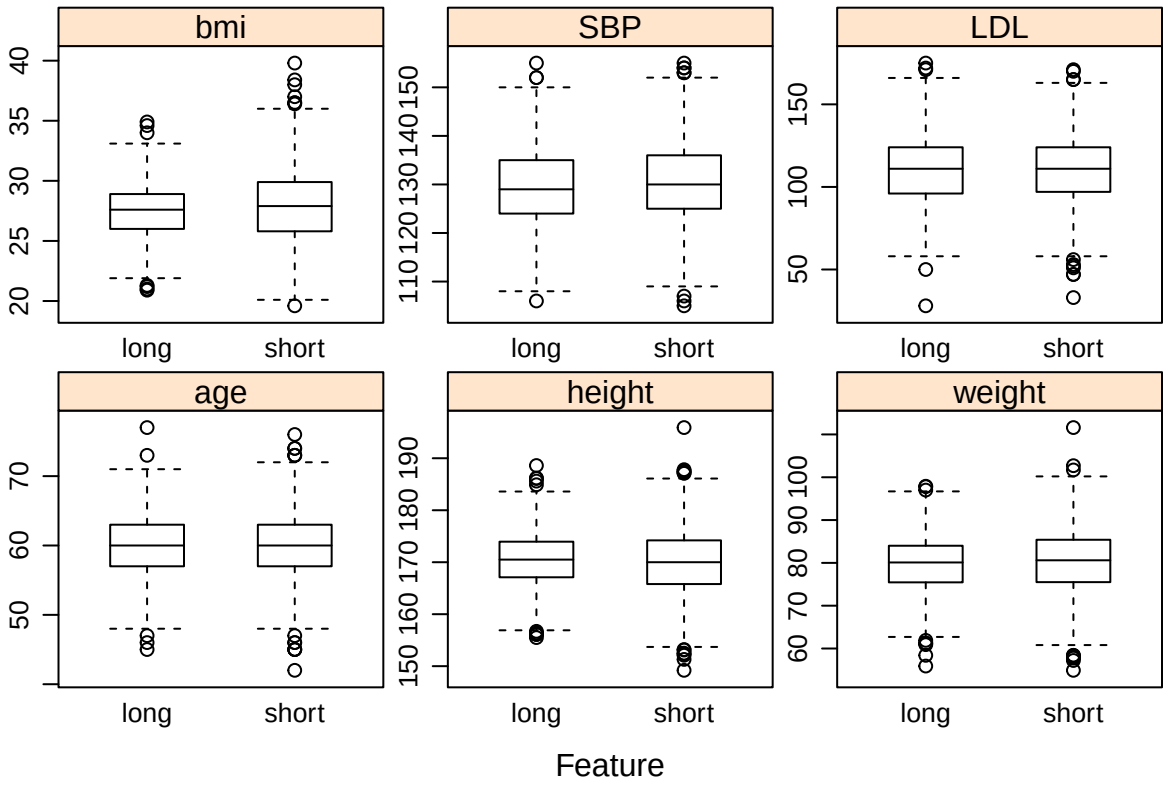
1.1 Correlation plot

```
corr_dat = covid_dat[rowTrain,] %>%
  dplyr::select('age', 'height', 'weight', 'bmi', 'SBP', 'LDL')
corrplot(cor(corr_dat), method = "circle", type = "full")
```



1.2 Feature plot

```
vis_trdat = trainData %>%
  dplyr::select('age', 'height', 'weight', 'bmi', 'SBP', 'LDL', 'recovery_time')

theme1 = transparentTheme(trans = .4)
trellis.par.set(theme1)

featurePlot(x = vis_trdat[, 1:6],
            y = vis_trdat[, 7],
            scales = list(x = list(relation = "free"),
                          y = list(relation = "free")),
            plot = "box", pch = "|",
            auto.key = list(columns = 2))
```



2. Model training

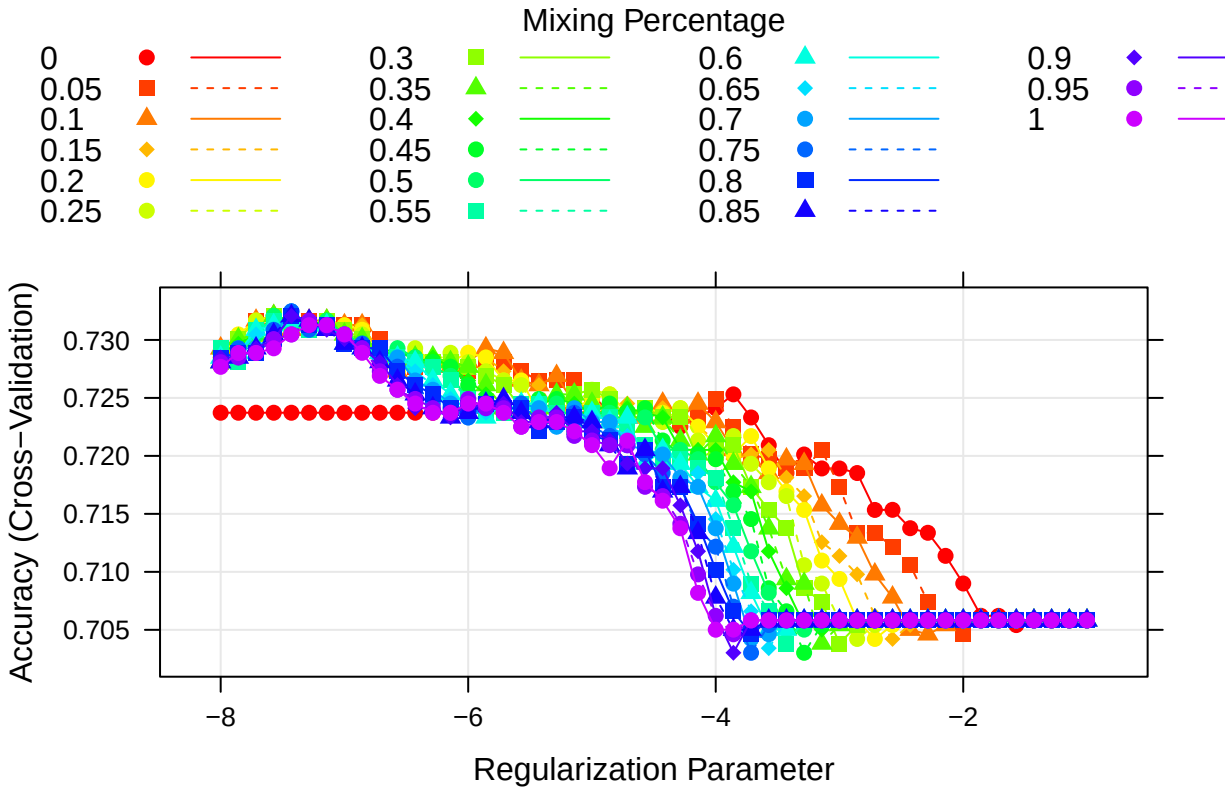
2.1 GLM

```
set.seed(2)
model.glm <- train(x = covid_dat2[rowTrain,],
```

```
                    y = covid_dat$recovery_time[rowTrain],
                    method = "glm",
                    trControl = ctrl)
```

2.2 Penalized logistic regression

```
glmnGrid <- expand.grid(.alpha = seq(0, 1, length = 21),
                        .lambda = exp(seq(-8, -1, length = 50)))
set.seed(2)
model.glmn <- train(x = covid_dat2[rowTrain,],
                    y = covid_dat$recovery_time[rowTrain],
                     method = "glmnet",
                     tuneGrid = glmnGrid,
                     trControl = ctrl)

myCol<- rainbow(25)
myPar <- list(superpose.symbol = list(col = myCol),
              superpose.line = list(col = myCol))

plot(model.glmn, par.settings = myPar, xTrans = function(x) log(x))
```

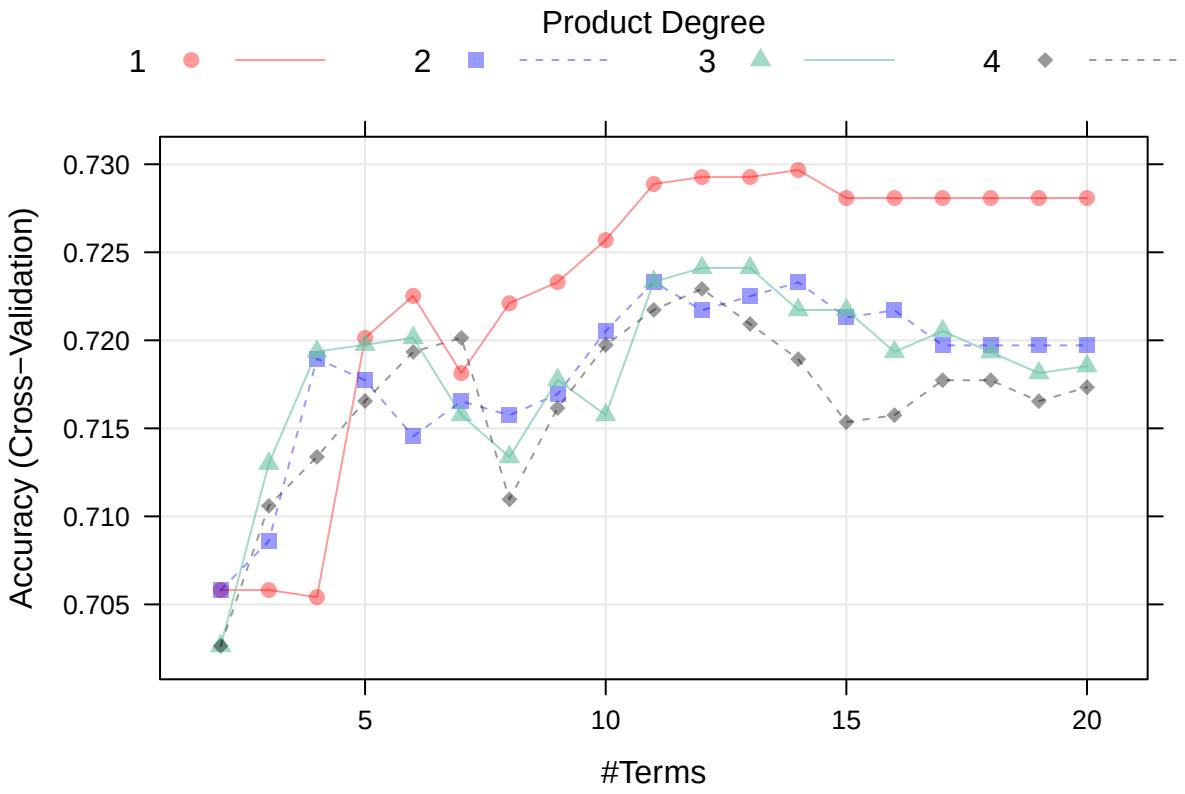


2.3 GAM

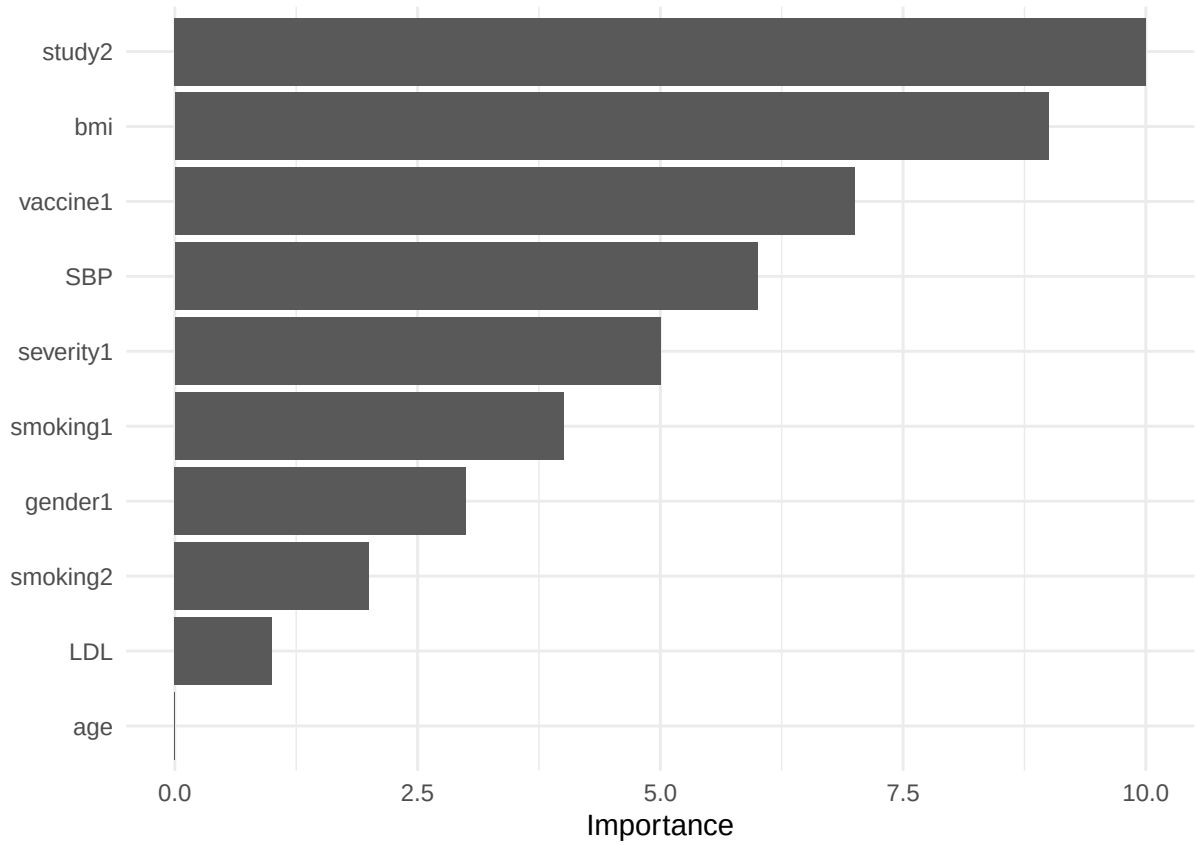
```
set.seed(2)
model.gam <- train(x = covid_dat2[rowTrain,],
                   y = covid_dat$recovery_time[rowTrain],
                   method = "gam",
                   trControl = ctrl)
```

2.4 MARS

```
set.seed(2)
model.mars <- train(x = covid_dat2[rowTrain,],
                   y = covid_dat$recovery_time[rowTrain],
                    method = "earth",
                    tuneGrid = expand.grid(degree = 1:4,
                                           nprune = 2:20),
                    trControl = ctrl)

plot(model.mars)
```




```
vip(model.mars$finalModel)
```



2.5 LDA

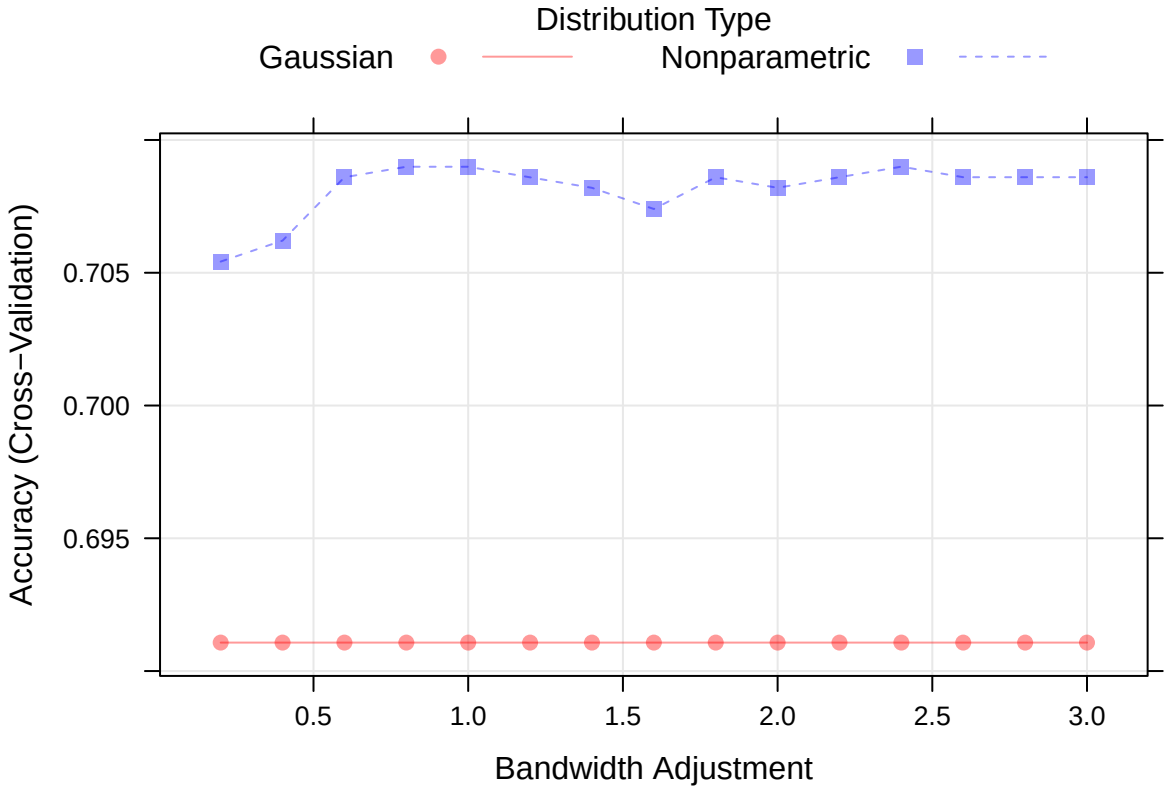
```
set.seed(2)

model.lda <- train(x = covid_dat2[rowTrain,],
                   y = covid_dat$recovery_time[rowTrain],
                   method = "lda",
                   trControl = ctrl)
```

2.6 QDA

```
set.seed(2)
model.qda <- train(x = covid_dat2[rowTrain,],
                   y = covid_dat$recovery_time[rowTrain],
                   method = "qda",
                   trControl = ctrl)
```

2.7 Naive Bayes (NB)

```
nbGrid <- expand.grid(usekernel = c(FALSE,TRUE),
                      fL = 1,
                      adjust = seq(.2, 3, by = .2))

set.seed(2)
model.nb <- train(x = covid_dat2[rowTrain,],
                  y = covid_dat$recovery_time[rowTrain],
                  method = "nb",
                  tuneGrid = nbGrid,
                  trControl = ctrl)

plot(model.nb)
```

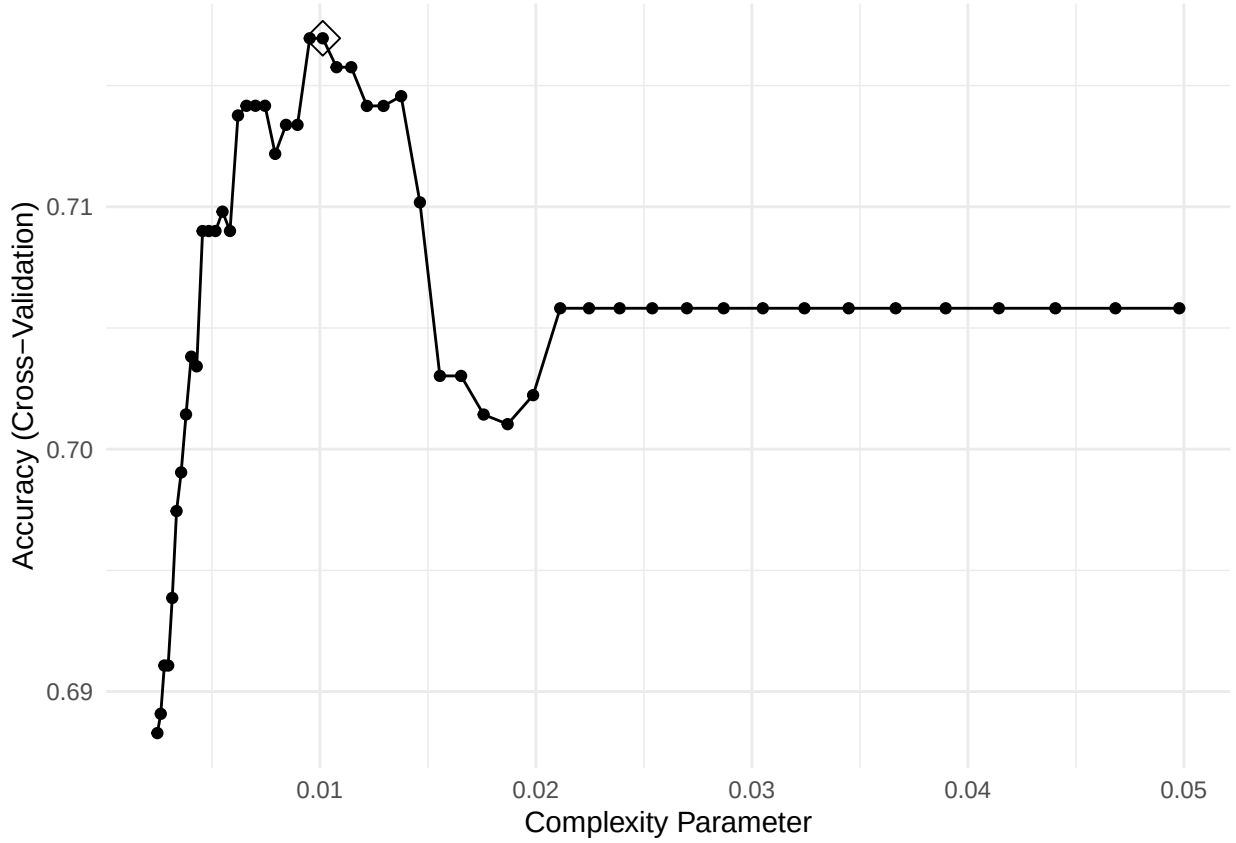


2.8 classification tree models

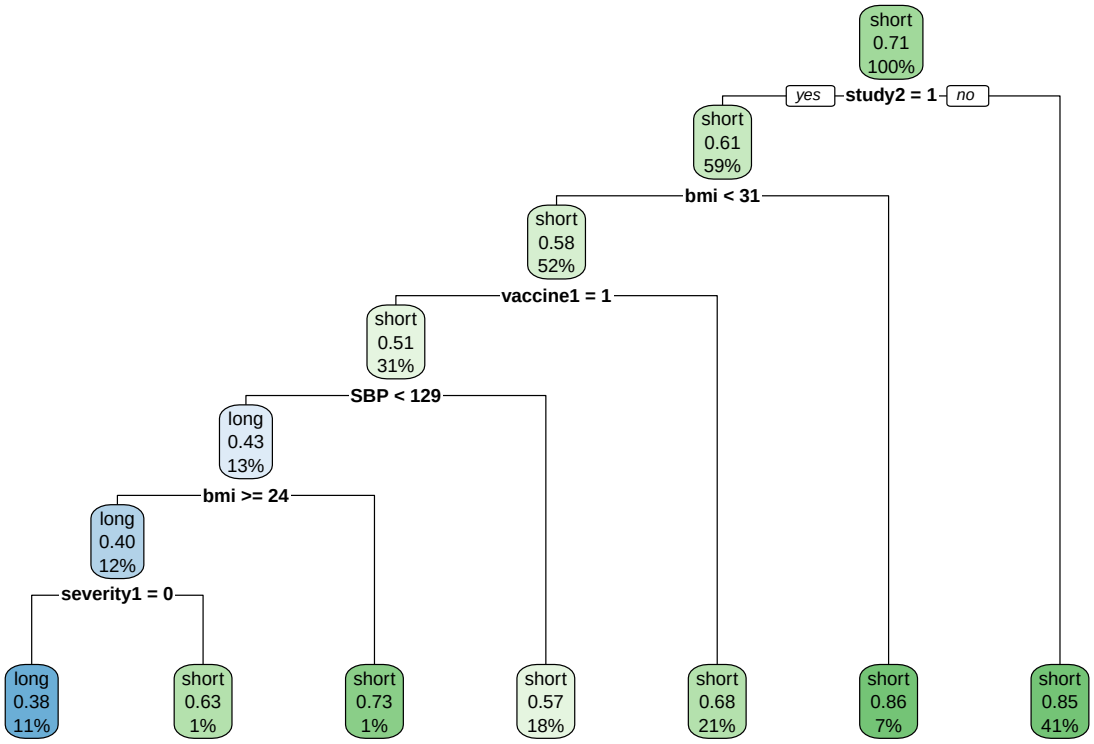
2.8.1 classification tree-rpart

```
num_cores <- detectCores()
cl <- makePSOCKcluster(num_cores)
registerDoParallel(cl)
set.seed(2)
```

```
model.rpart = train(recovery_time ~ .,
                    covid_dat,
                    subset = rowTrain,
                    method = "rpart",
                    tuneGrid = data.frame(cp = exp(seq(-6, -3, len = 50))),
                    trControl = ctrl)

ggplot(model.rpart, highlight = TRUE)
```



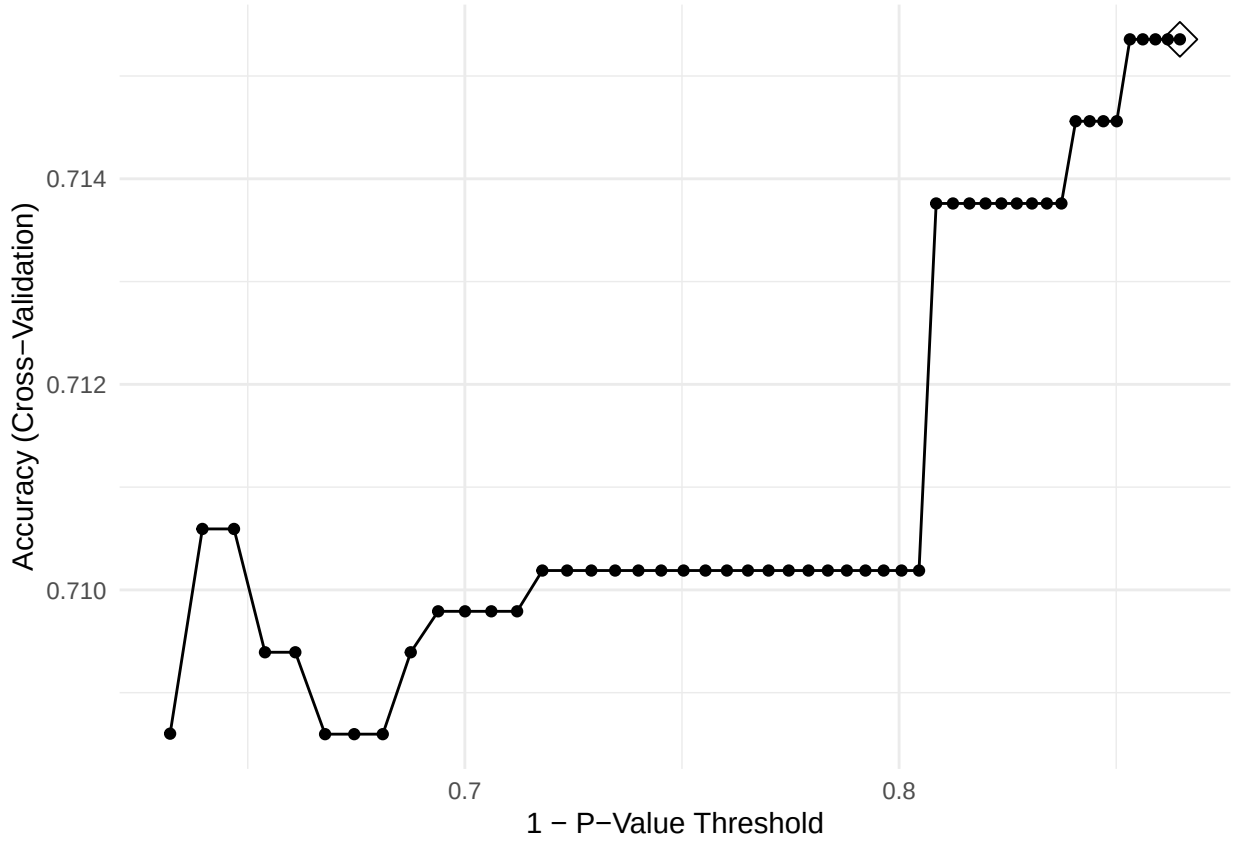

```
rpart.plot(model.rpart$finalModel)
```



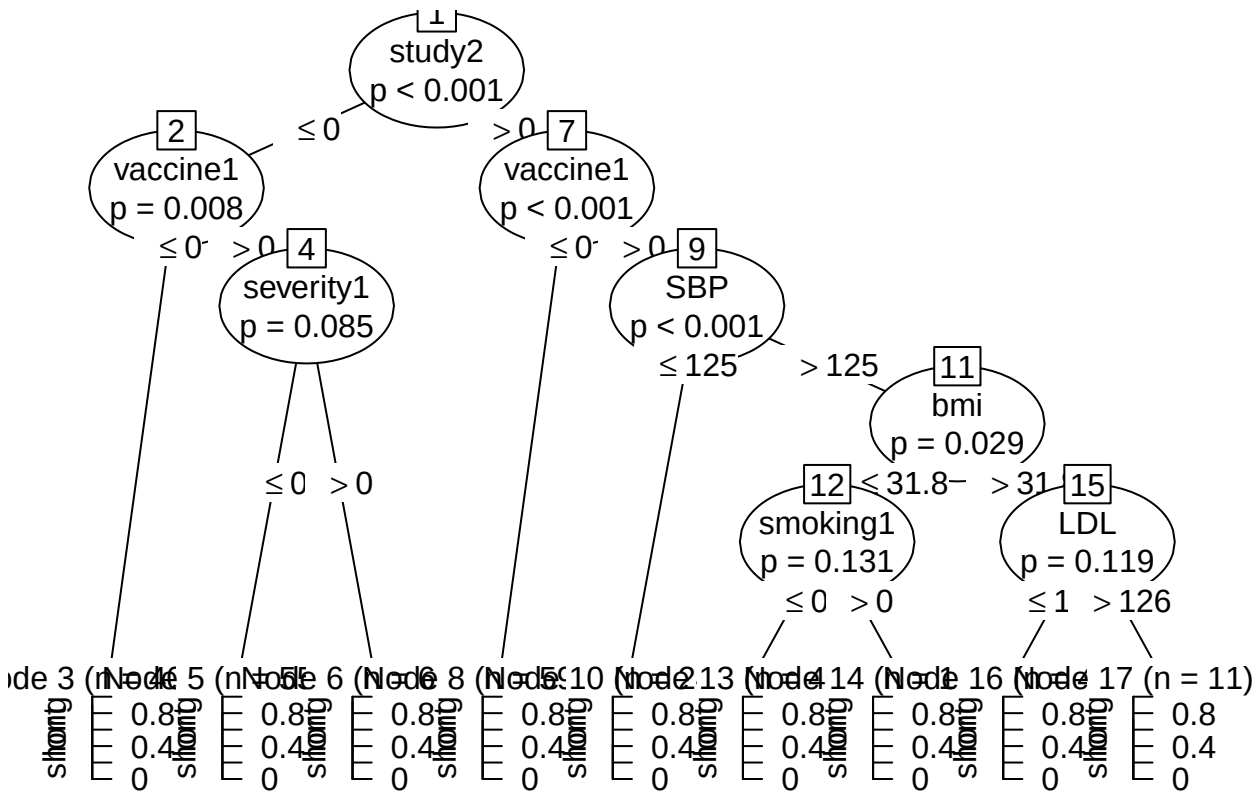

```
stopCluster(cl)
registerDoSEQ()
```

2.8.2 classification ctree-ctree

```
num_cores <- detectCores()
cl <- makePSOCKcluster(num_cores)
registerDoParallel(cl)

set.seed(2)


model.ctree = train(recovery_time ~ .,
                    covid_dat,
                    subset = rowTrain,
                    method = "ctree",
                    tuneGrid = data.frame(mincriterion = 1 - exp(seq(-2, -1, length = 50))),
                    trControl = ctrl)

ggplot(model.ctree, highlight = TRUE)
```



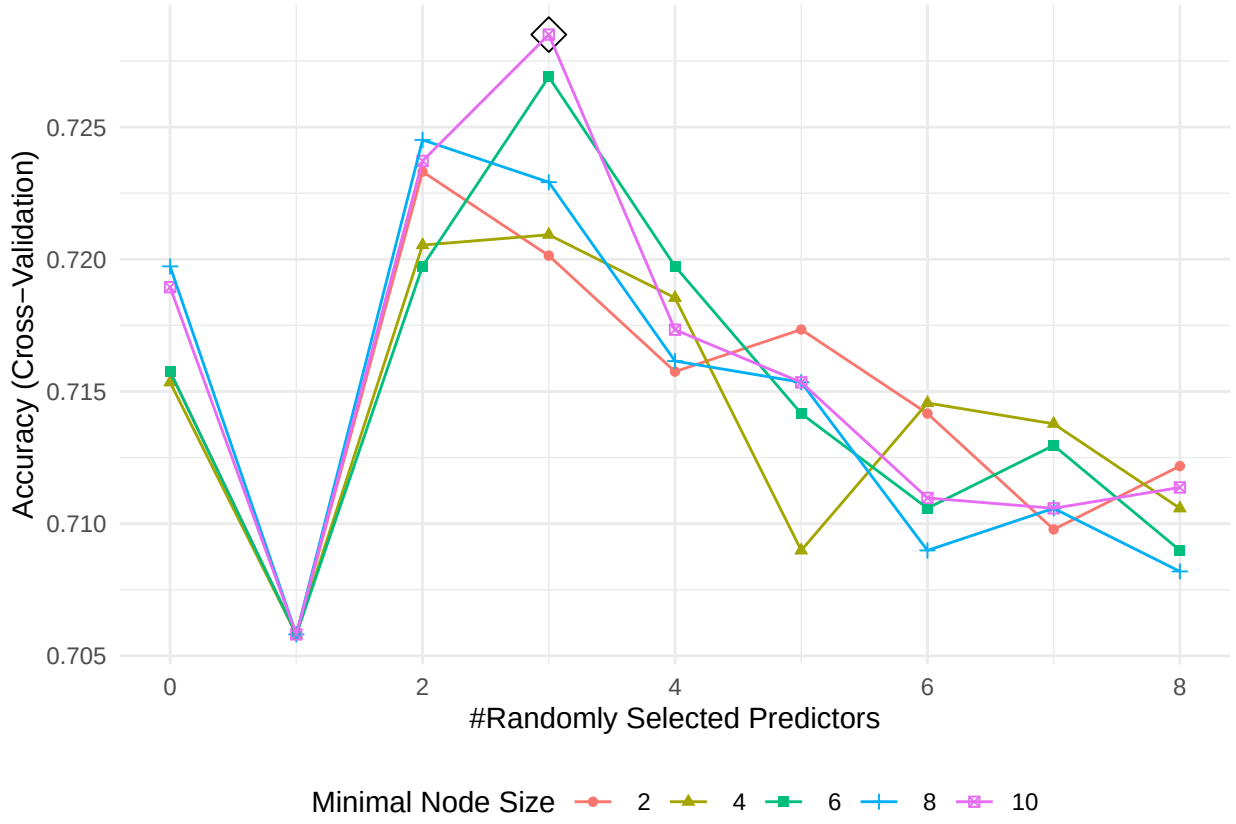

```
plot(model.ctree$finalModel)
```



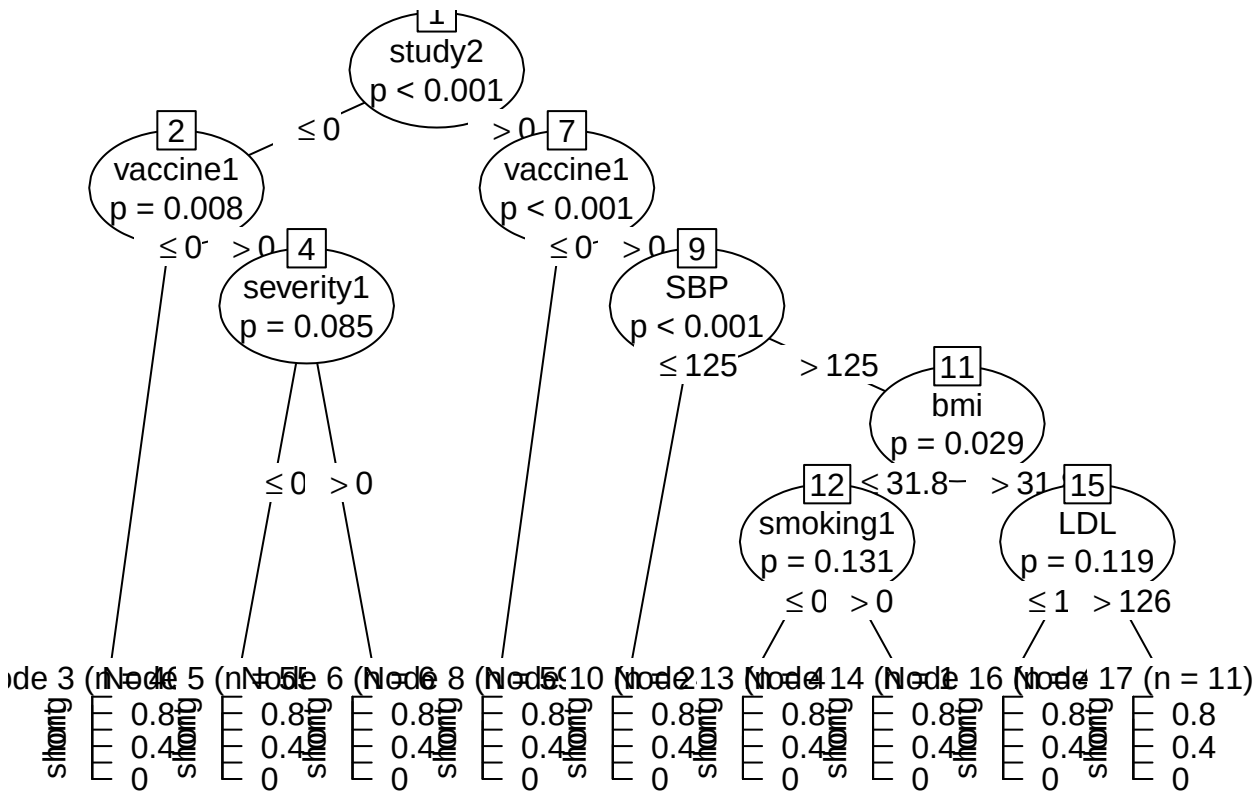

```
stopCluster(cl)
registerDoSEQ()
```

2.9 Random forests

```
num_cores <- detectCores()
cl <- makePSOCKcluster(num_cores)
registerDoParallel(cl)

rf.grid = expand.grid(mtry = 0:8,
                      splitrule = "gini",
                      min.node.size = seq(from = 2, to = 10, by = 2))
set.seed(2)
rf.fit = train(recovery_time ~ .,
               covid_dat,
               subset = rowTrain,
               method = "ranger",
               tuneGrid = rf.grid,
               trControl = ctrl)
ggplot(rf.fit, highlight = TRUE)
```




```
plot(model.ctree$finalModel)
```



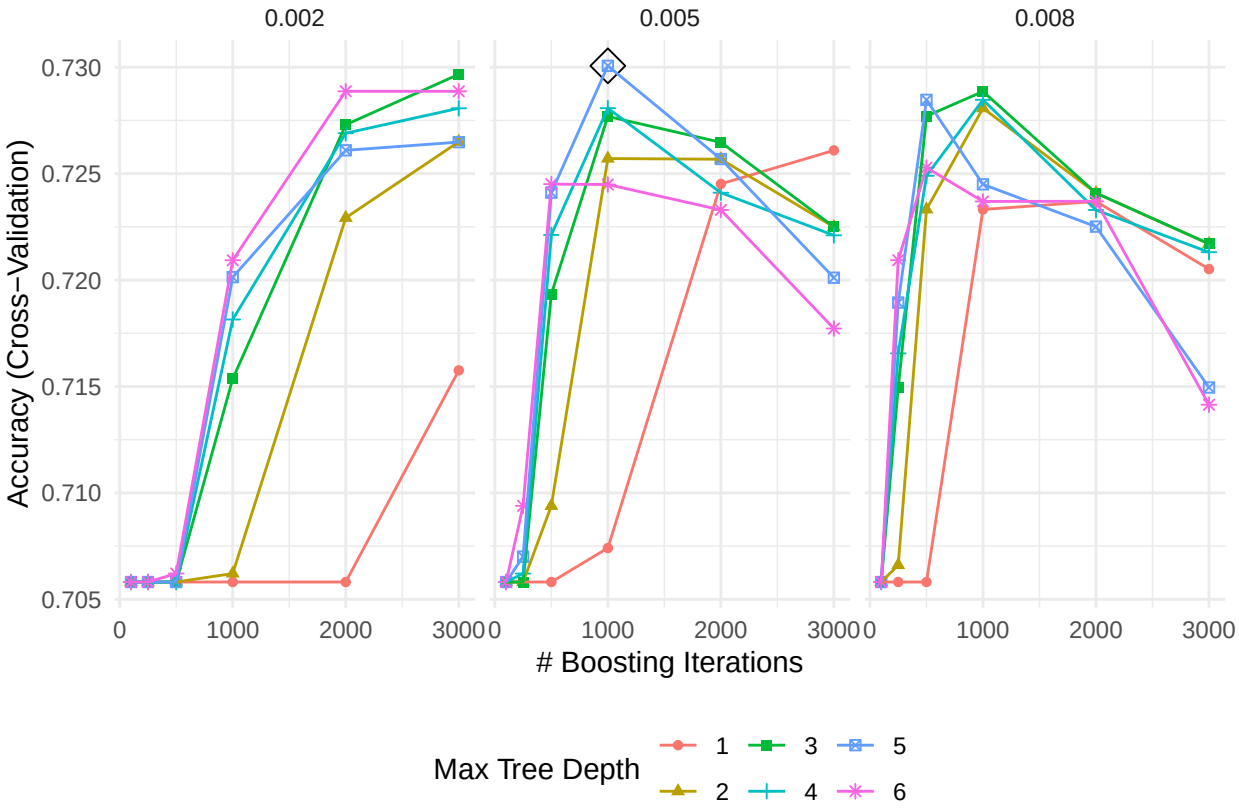

```
stopCluster(cl)
registerDoSEQ()
```

2.10 Boosting

```
num_cores <- detectCores()
cl <- makePSOCKcluster(num_cores)
registerDoParallel(cl)

gbmA_grid = expand.grid(n.trees = c(100, 250, 500, 1000, 2000, 3000),
                        interaction.depth = 1:6,
                        shrinkage = c(0.002, 0.005, 0.008),
                        n.minobsinnode = 1)
set.seed(2)
gbmA.fit = train(recovery_time ~ . ,
                 covid_dat[rowTrain,],
                 tuneGrid = gbmA_grid,
                 trControl = ctrl,
                 method = "gbm",
                 distribution = "adaboost",
                 verbose = FALSE)

gbmA.fit$bestTune
```

```
##    n.trees interaction.depth shrinkage n.minobsinnode
## 64    1000                 5     0.005              1
```

```
ggplot(gbmA.fit, highlight = TRUE)
```



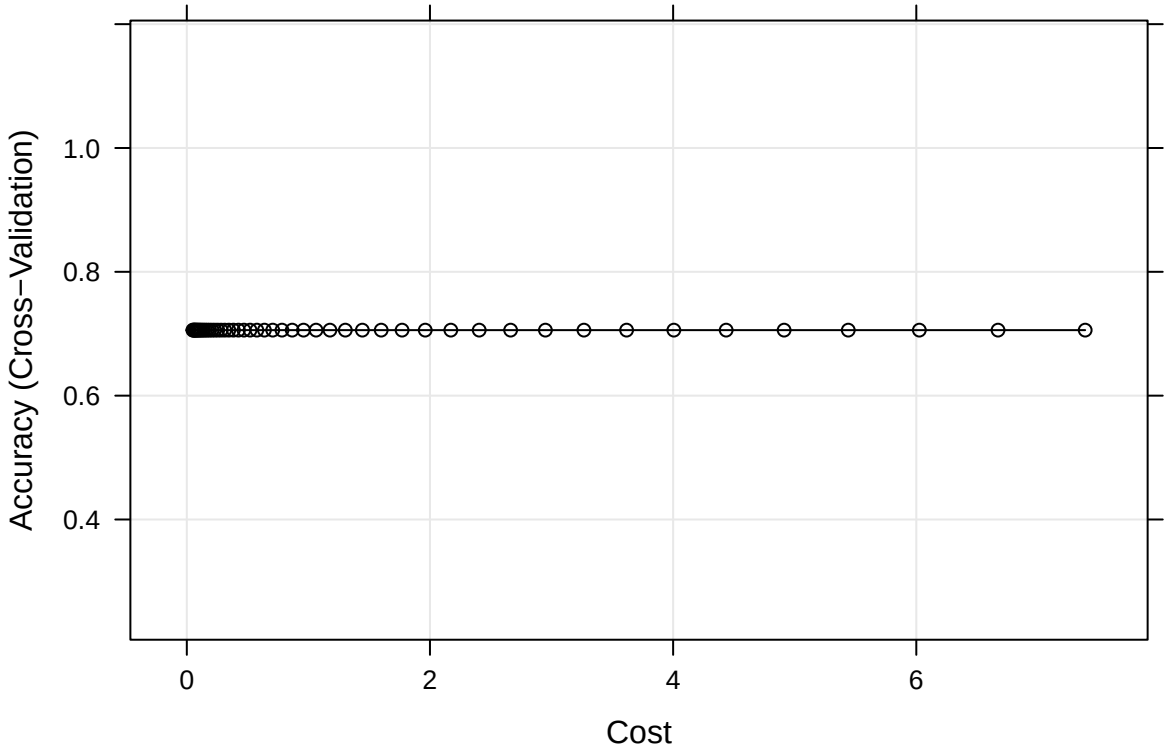

```
stopCluster(cl)
registerDoSEQ()
```

2.11 Support Vector Machines

2.11.1 Support Vecotor Machines Linear

```
num_cores <- detectCores()
cl <- makePSOCKcluster(num_cores)
registerDoParallel(cl)
set.seed(2)

model.svml <- train(recovery_time ~ .,
                    data = covid_dat[rowTrain, ],
                    method = "svmLinear2",
                    preProcess = c("center", "scale"),
                    tuneGrid = data.frame(cost = exp(seq(-3,2,len = 50))),
                    trControl = ctrl)
```

```
plot(model.svml, highlight = TRUE)
```




```
model.svml$bestTune
```

```
##         cost
## 1 0.04978707
```

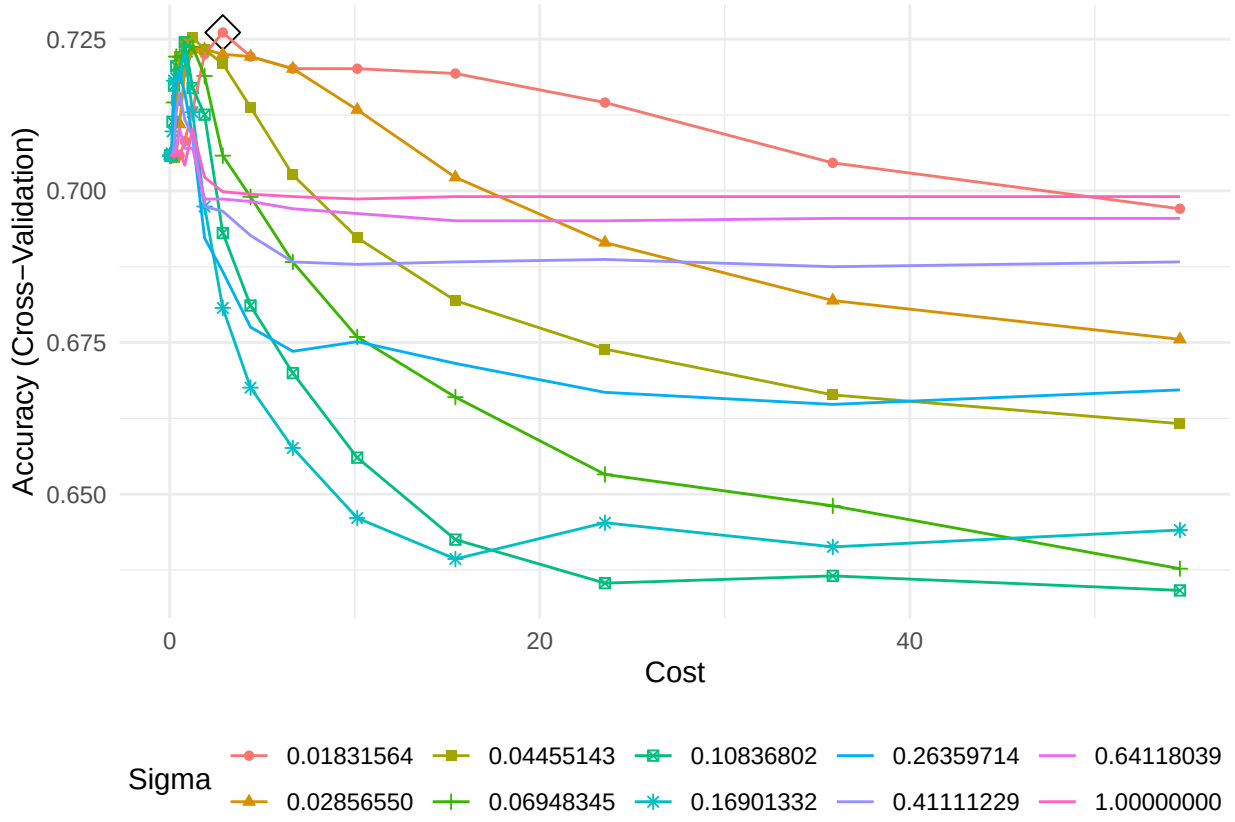
```
model.svml$finalModel
```

```
##
## Call:
## svm.default(x = as.matrix(x), y = y, kernel = "linear", cost = param$cost,
##     probability = classProbs)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.04978707
##
## Number of Support Vectors:  1655
```

```
stopCluster(cl)
registerDoSEQ()
```

2.11.2 Support Vecotor Machines Radial Kernal

```
num_cores <- detectCores()
cl <- makePSOCKcluster(num_cores)
registerDoParallel(cl)
svmr.grid <- expand.grid(C = exp(seq(-4,4,len=20)),
                         sigma = exp(seq(-4,0,len=10)))
#radial kernel
set.seed(2)

model.svmr <- train(recovery_time ~ .,
                    data = covid_dat[rowTrain, ],
                    method = "svmRadialSigma",
                    preProcess = c("center", "scale"),
                    tuneGrid = svmr.grid,
                    trControl = ctrl)

myCol<- rainbow(20)
myPar <- list(superpose.symbol = list(col = myCol),
              superpose.line = list(col = myCol))
ggplot(model.svmr, highlight = TRUE, par.settings = myPar)
```



3. Model Selection

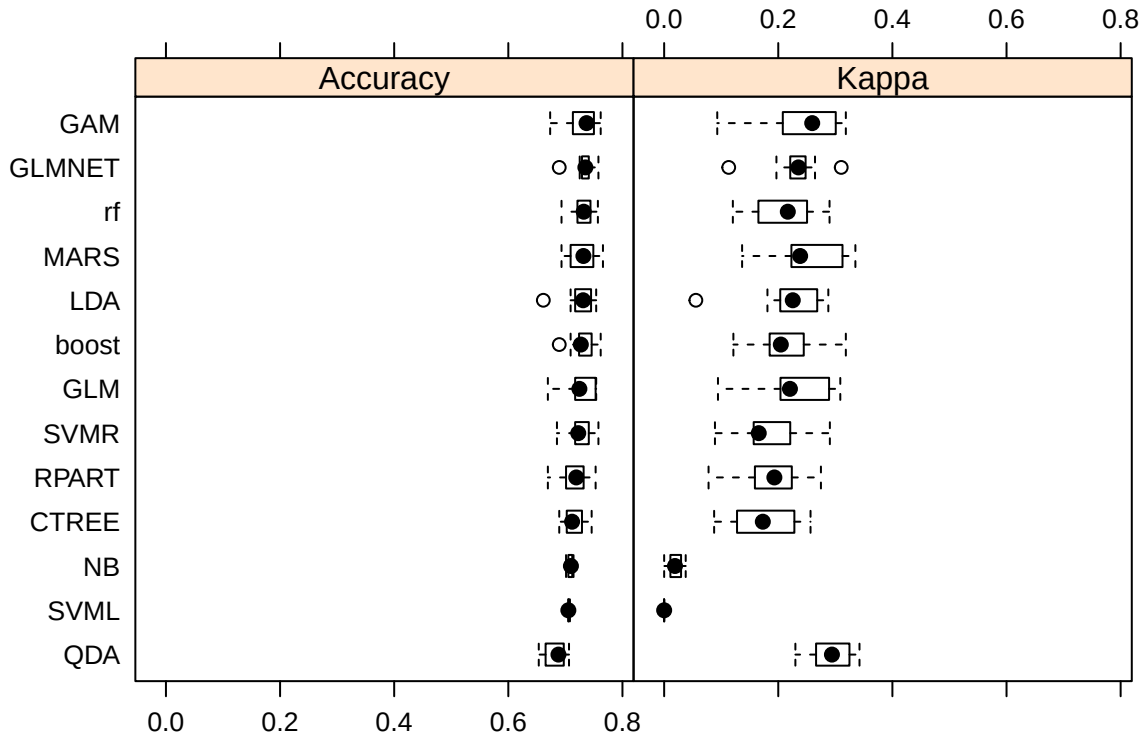
3.1 Model Comparison

```
res <- resamples(list(GLM = model.glm,
                      GLMNET = model.glmn,
                      GAM = model.gam,
                      MARS = model.mars,
                      CTREE = model.ctree,
                      RPART = model.rpart,
                      LDA = model.lda,
                      QDA = model.qda,
                      NB = model.nb,
                      SVML=model.svml,
                      SVMR=model.svmr,
                      rf = rf.fit,
                      boost = gbmA.fit))

summary(res)
```

```
##
## Call:
## summary.resamples(object = res)
##
## Models: GLM, GLMNET, GAM, MARS, CTREE, RPART, LDA, QDA, NB, SVML, SVMR, rf, boost
## Number of resamples: 10
##
## Accuracy
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## GLM    0.6693227 0.7171315 0.7245498 0.7264852 0.7492648 0.7539683    0
## GLMNET 0.6892430 0.7290837 0.7350476 0.7324677 0.7410359 0.7579365    0
## GAM    0.6733068 0.7138606 0.7370518 0.7280694 0.7487550 0.7619048    0
## MARS   0.6932271 0.7128725 0.7316211 0.7296709 0.7460159 0.7658730    0
## CTREE  0.6892430 0.7027857 0.7117324 0.7153567 0.7280876 0.7460317    0
## RPART  0.6693227 0.7044678 0.7191235 0.7169504 0.7315397 0.7529880    0
## LDA    0.6613546 0.7178486 0.7310757 0.7252916 0.7422967 0.7539683    0
## QDA    0.6533865 0.6673307 0.6878597 0.6823142 0.6965195 0.7063492    0
## NB     0.7011952 0.7061753 0.7097404 0.7089949 0.7115794 0.7142857    0
## SVML   0.7051793 0.7051793 0.7051793 0.7058123 0.7063492 0.7080000    0
## SVMR   0.6852590 0.7174121 0.7225578 0.7261011 0.7393086 0.7579365    0
## rf     0.6932271 0.7221116 0.7321429 0.7284995 0.7432590 0.7569721    0
## boost  0.6892430 0.7242749 0.7270916 0.7300662 0.7447828 0.7619048    0
##
## Kappa
##              Min.    1st Qu.     Median       Mean    3rd Qu.       Max. NA's
## GLM    0.09433552 0.20388653 0.22045423 0.22744909 0.27757554 0.30867257    0
## GLMNET 0.11315456 0.22287843 0.23528886 0.22981947 0.24732107 0.31030151    0
## GAM    0.09306425 0.21215140 0.25949364 0.24084158 0.29697083 0.31842770    0
## MARS   0.13660934 0.22335521 0.23832745 0.24494581 0.29855843 0.33512111    0
## CTREE  0.08761301 0.13147055 0.17293133 0.17187331 0.21649480 0.25652993    0
## RPART  0.07781860 0.16651610 0.19316349 0.19104015 0.22286987 0.27476932    0
## LDA    0.05559736 0.20356043 0.22539191 0.21647634 0.26163901 0.28768844    0
```

```
## QDA    0.22997990 0.26898706 0.29407403 0.29098635 0.32226971 0.34233914    0
## NB     0.00000000 0.01296529 0.01898464 0.01953669 0.02706407 0.03775986    0
## SVML   0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000    0
## SVMR   0.08903386 0.15770571 0.16579778 0.18231483 0.21708158 0.29043575    0
## rf     0.12042052 0.17112041 0.21671598 0.20609235 0.24407350 0.28987524    0
## boost  0.12135356 0.18589363 0.20446587 0.21653752 0.24432311 0.31842770    0
```

```
bwplot(res)
```



3.2 Final Model- GAM

```
# summary
model.gam$finalModel
```

```
##
## Family: binomial
## Link function: logit
##
## Formula:
## .outcome ~ gender1 + race3 + race4 + smoking1 + smoking2 + hypertension1 +
##     diabetes1 + vaccine1 + severity1 + study2 + study3 + s(age) +
##     s(SBP) + s(LDL) + s(bmi) + s(height) + s(weight)
##
```

```
## Estimated degrees of freedom:
## 0.7814 0.0000 0.0000 3.5173 1.4493 0.0001  total = 17.75
##
## UBRE score: 0.06658708
```

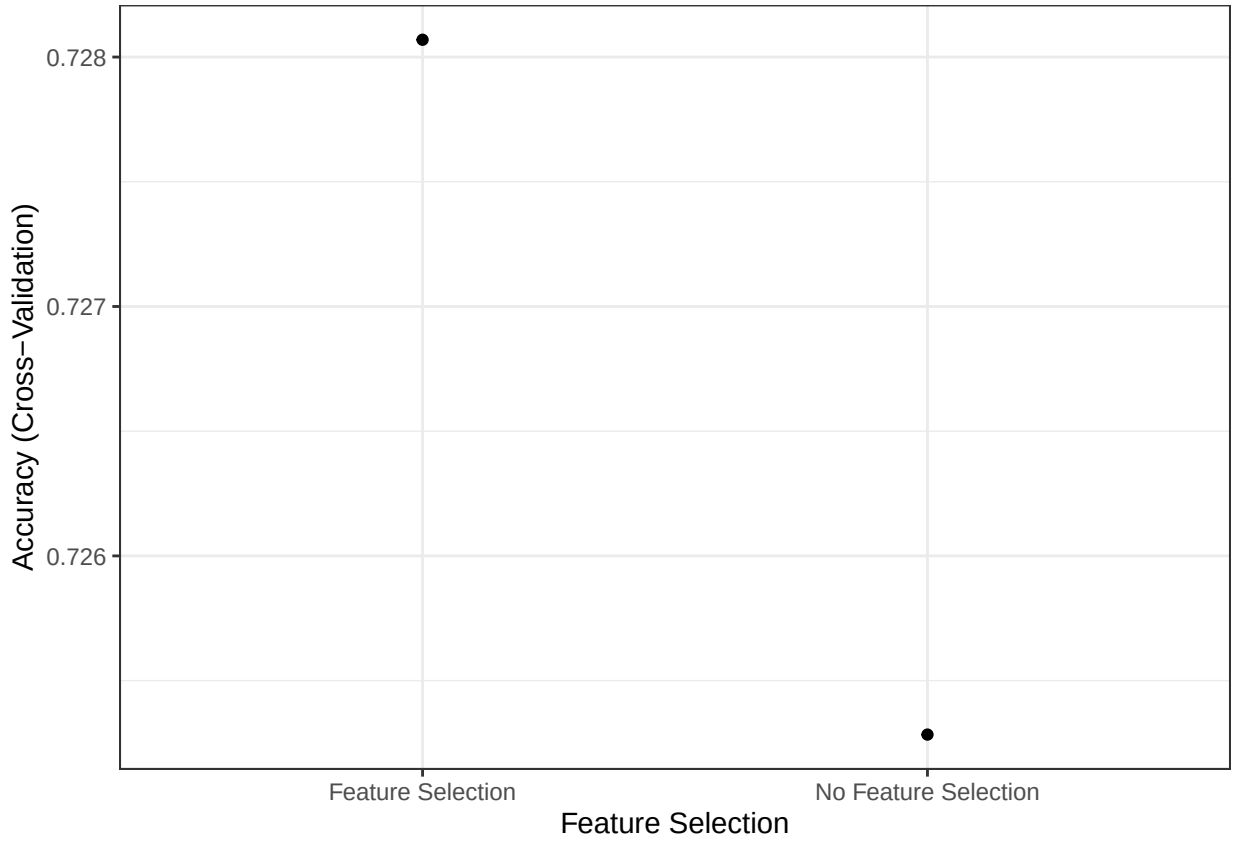
```
model.gam$bestTune
```

```
##   select method
## 2   TRUE GCV.Cp
```

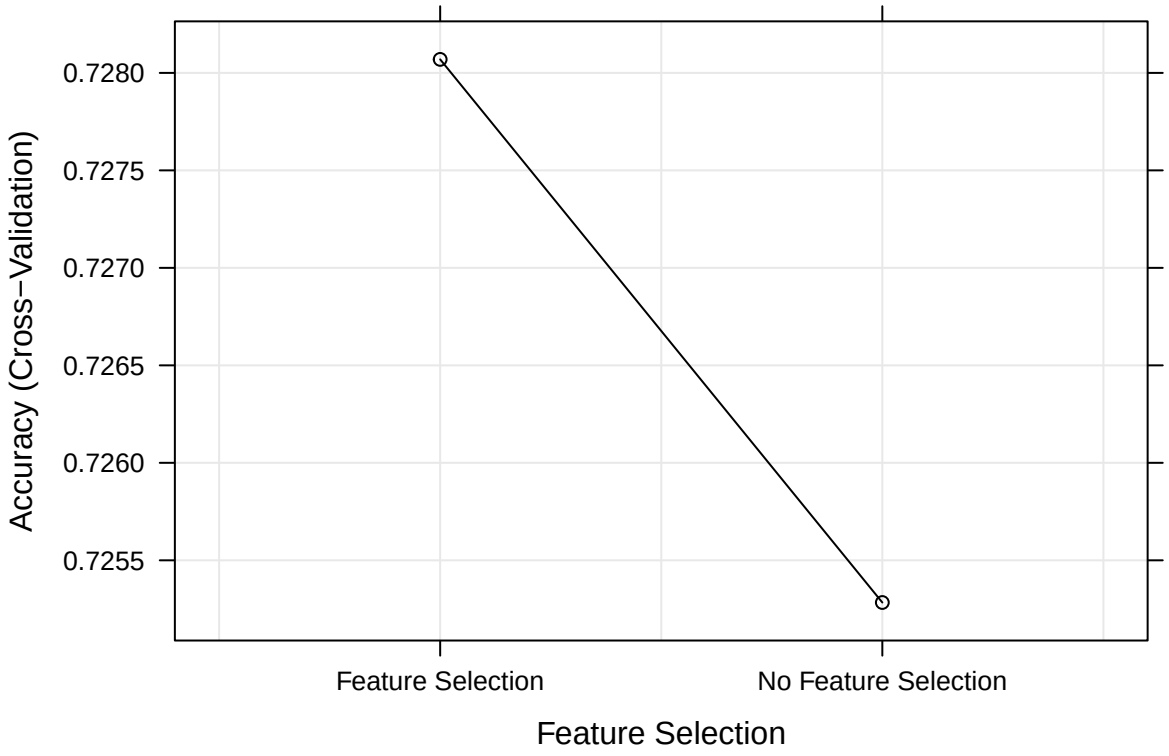
```
summary(model.gam)
```

```
##
## Family: binomial
## Link function: logit
##
## Formula:
## .outcome ~ gender1 + race3 + race4 + smoking1 + smoking2 + hypertension1 +
##     diabetes1 + vaccine1 + severity1 + study2 + study3 + s(age) +
##     s(SBP) + s(LDL) + s(bmi) + s(height) + s(weight)
##
## Parametric coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)    2.28480    0.17898  12.766  < 2e-16 ***
## gender1       -0.32518    0.09530  -3.412 0.000645 ***
## race3         -0.11553    0.11859  -0.974 0.329947
## race4         -0.05404    0.16682  -0.324 0.745978
## smoking1       0.42913    0.10732   3.998 6.38e-05 ***
## smoking2       0.54608    0.17350   3.147 0.001647 **
## hypertension1  0.32009    0.10199   3.138 0.001699 **
## diabetes1      0.04432    0.13201   0.336 0.737058
## vaccine1      -0.72124    0.10050  -7.177 7.14e-13 ***
## severity1      0.79673    0.18421   4.325 1.52e-05 ***
## study2        -1.56884    0.15154 -10.352  < 2e-16 ***
## study3        -0.37745    0.18229  -2.071 0.038390 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                edf Ref.df Chi.sq p-value
## s(age)    7.814e-01      9  3.572  0.0324 *
## s(SBP)    4.152e-05      9  0.000  0.5056
## s(LDL)    4.583e-05      9  0.000  0.4524
## s(bmi)    3.517e+00      9 61.167  <2e-16 ***
## s(height) 1.449e+00      9  1.721  0.2912
## s(weight) 9.245e-05      9  0.000  0.8029
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.146   Deviance explained = 13.1%
## UBRE = 0.066587  Scale est. = 1         n = 2512
```

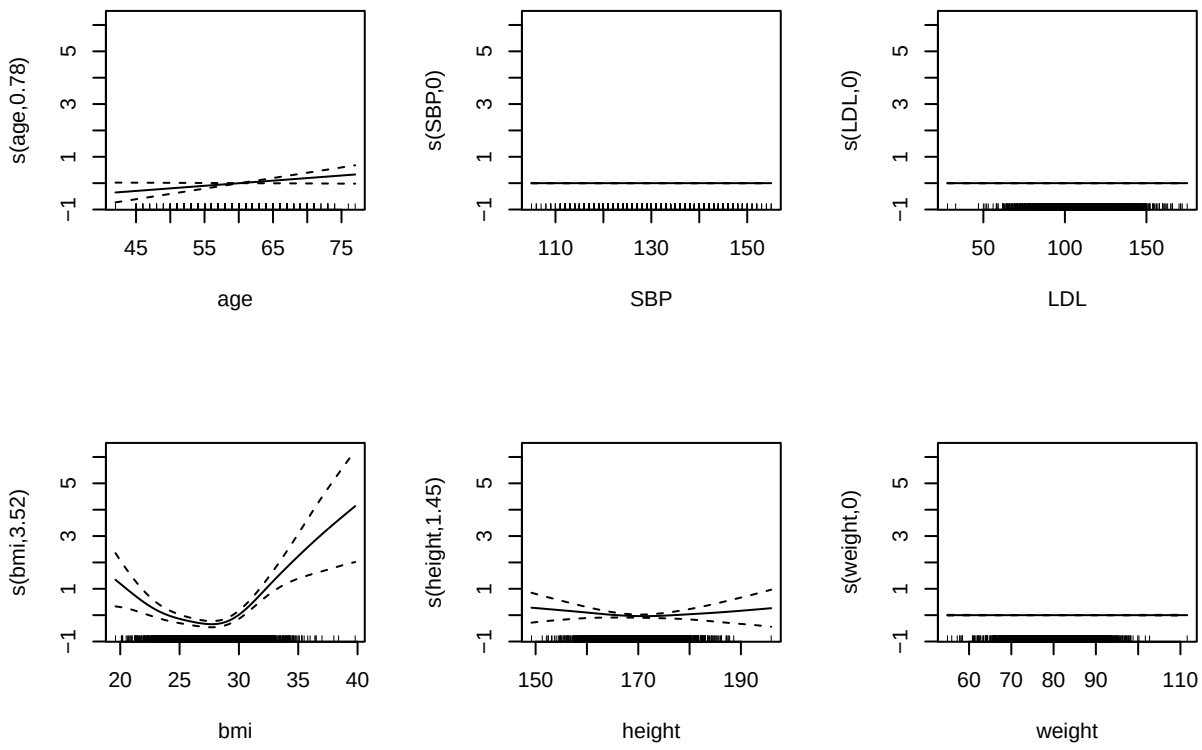
```r
# visualization
ggplot(model.gam) +
  labs(tital = "GAM Classification") +
  theme_bw()
```




```r
plot(model.gam)
```




```
par(mfrow = c(2,3))
plot(model.gam$finalModel)
```




```
par(mfrow = c(1,1))


# training error
pred.gam.train = predict(model.gam, newdata = covid_dat2[rowTrain,])
confusionMatrix(data = pred.gam.train, reference = covid_dat$recovery_time[rowTrain])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction long short
##      long   230   158
##      short  509  1615
##
##                Accuracy : 0.7345
##                  95% CI : (0.7167, 0.7517)
##     No Information Rate : 0.7058
##     P-Value [Acc > NIR] : 0.0007922
##
##                   Kappa : 0.2578
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.31123
##             Specificity : 0.91089
```

```
##          Pos Pred Value : 0.59278
##          Neg Pred Value : 0.76036
##              Prevalence : 0.29419
##          Detection Rate : 0.09156
##    Detection Prevalence : 0.15446
##       Balanced Accuracy : 0.61106
##
##        'Positive' Class : long
##
```

```
# test error
pred.gam.test = predict(model.gam, newdata = covid_dat2[-rowTrain,])
confusionMatrix(data = pred.gam.test, reference = covid_dat$recovery_time[-rowTrain])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction long short
##      long   104    82
##      short  212   677
##
##                Accuracy : 0.7265
##                  95% CI : (0.6988, 0.753)
##     No Information Rate : 0.706
##     P-Value [Acc > NIR] : 0.07429
##
##                   Kappa : 0.2512
##
##  Mcnemar's Test P-Value : 5.336e-14
##
##             Sensitivity : 0.32911
##             Specificity : 0.89196
##          Pos Pred Value : 0.55914
##          Neg Pred Value : 0.76153
##              Prevalence : 0.29395
##          Detection Rate : 0.09674
##    Detection Prevalence : 0.17302
##       Balanced Accuracy : 0.61054
##
##        'Positive' Class : long
##
```