ds2_final

Ayako, Sekiya, Maisie Sun, and Daisy Yan

2023-04-29

Merge data

```
data_2101 <-
  read_csv("./data/2101_data.csv") %>%
  janitor::clean_names() %>%
  na.omit()
```

```
## Rows: 2000 Columns: 15
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (1): study
## dbl (14): age, gender, race, smoking, height, weight, bmi, hypertension, dia...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
data_6360 <-
  read_csv("./data/6360_data.csv") %>%
  janitor::clean_names() %>%
  na.omit()
```

```
## Rows: 2000 Columns: 15
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (1): study
## dbl (14): age, gender, race, smoking, height, weight, bmi, hypertension, dia...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
data <- rbind(data_2101, data_6360) %>%
  unique() %>%
  mutate(gender = as.factor(gender),
         smoking = as.factor(smoking),
         race = as.factor(race),
         hypertension = as.factor(hypertension),
         diabetes = as.factor(diabetes),
         vaccine = as.factor(vaccine),
         severity = as.factor(severity),
         study = as.factor(study))
```

Data partition: training and testing datasets

```
set.seed(6360)
trRows <- createDataPartition(data$recovery_time,
                              p = .80,
                              list = F)

# training data
trainData <- data[trRows, ]
trainData_matrix <- model.matrix(recovery_time~.,data)[trRows, ]
train_x <- model.matrix(recovery_time~.,data)[trRows,-1]
train_y <- data$recovery_time[trRows]

# test data
testData <- data[-trRows, ]
testData_matrix <- model.matrix(recovery_time~.,data)[-trRows,]
test_x <- model.matrix(recovery_time~.,data)[-trRows,-1]
test_y <- data$recovery_time[-trRows]

ctrl <- trainControl(method = "cv", number = 10, repeats = 5, selectionFunction = "best", savePredictio

str(data)
```

```
## tibble [3,594 x 15] (S3: tbl_df/tbl/data.frame)
##  $ age          : num [1:3594] 67 63 65 62 61 68 61 59 53 58 ...
##  $ gender       : Factor w/ 2 levels "0","1": 2 2 2 1 1 2 1 2 2 1 ...
##  $ race         : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 2 1 1 4 1 3 ...
##  $ smoking      : Factor w/ 3 levels "0","1","2": 2 1 1 1 1 2 1 2 1 1 ...
##  $ height       : num [1:3594] 171 156 168 166 171 ...
##  $ weight       : num [1:3594] 89 85.9 74.8 77 72.5 82.9 91.6 82.8 74.4 83.6 ...
##  $ bmi          : num [1:3594] 30.5 35.5 26.7 27.8 24.8 26.7 32.1 26.5 23.6 26.1 ...
##  $ hypertension : Factor w/ 2 levels "0","1": 2 2 2 1 2 2 2 1 1 1 ...
##  $ diabetes     : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ sbp          : num [1:3594] 143 132 139 127 133 133 138 121 116 121 ...
##  $ ldl          : num [1:3594] 86 107 123 109 65 109 97 138 134 125 ...
##  $ vaccine      : Factor w/ 2 levels "0","1": 1 2 2 2 1 2 2 2 2 1 ...
##  $ severity     : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ study        : Factor w/ 3 levels "A","B","C": 3 2 1 2 2 1 3 3 3 2 ...
##  $ recovery_time: num [1:3594] 48 133 24 18 105 36 35 32 38 6 ...
```

Exploratory analysis and data visualization

Summary and NA's

Summary is used for outlier visualization and to see if there are any NAs in the data.

```
summary(trainData)
```

```
##       age          gender    race     smoking      height           weight
##  Min.   :42.00   0:1492   1:1838   0:1722   Min.   :149.7   Min.   : 55.90
##  1st Qu.:57.00   1:1385   2: 149   1: 863   1st Qu.:166.0   1st Qu.: 75.10
##  Median :60.00            3: 599   2: 292   Median :170.0   Median : 79.90
##  Mean   :60.11            4: 291            Mean   :170.0   Mean   : 79.96
##  3rd Qu.:63.00                              3rd Qu.:173.9   3rd Qu.: 84.70
##  Max.   :75.00                              Max.   :189.1   Max.   :105.70
##       bmi        hypertension diabetes      sbp             ldl          vaccine
##  Min.   :19.70   0:1470       0:2459   Min.   :102.0   Min.   : 33.0   0:1127
##  1st Qu.:25.80   1:1407       1: 418   1st Qu.:125.0   1st Qu.: 97.0   1:1750
##  Median :27.60                         Median :130.0   Median :110.0
##  Mean   :27.74                         Mean   :130.4   Mean   :110.1
##  3rd Qu.:29.40                         3rd Qu.:136.0   3rd Qu.:123.0
##  Max.   :39.80                         Max.   :158.0   Max.   :182.0
##  severity study    recovery_time
##  0:2597   A: 552   Min.   :  2.00
##  1: 280   B:1766   1st Qu.: 28.00
##           C: 559   Median : 38.00
##                    Mean   : 42.41
##                    3rd Qu.: 49.00
##                    Max.   :365.00
```

```
trainData %>%
  summarise_all(~ sum(is.na(.)))
```

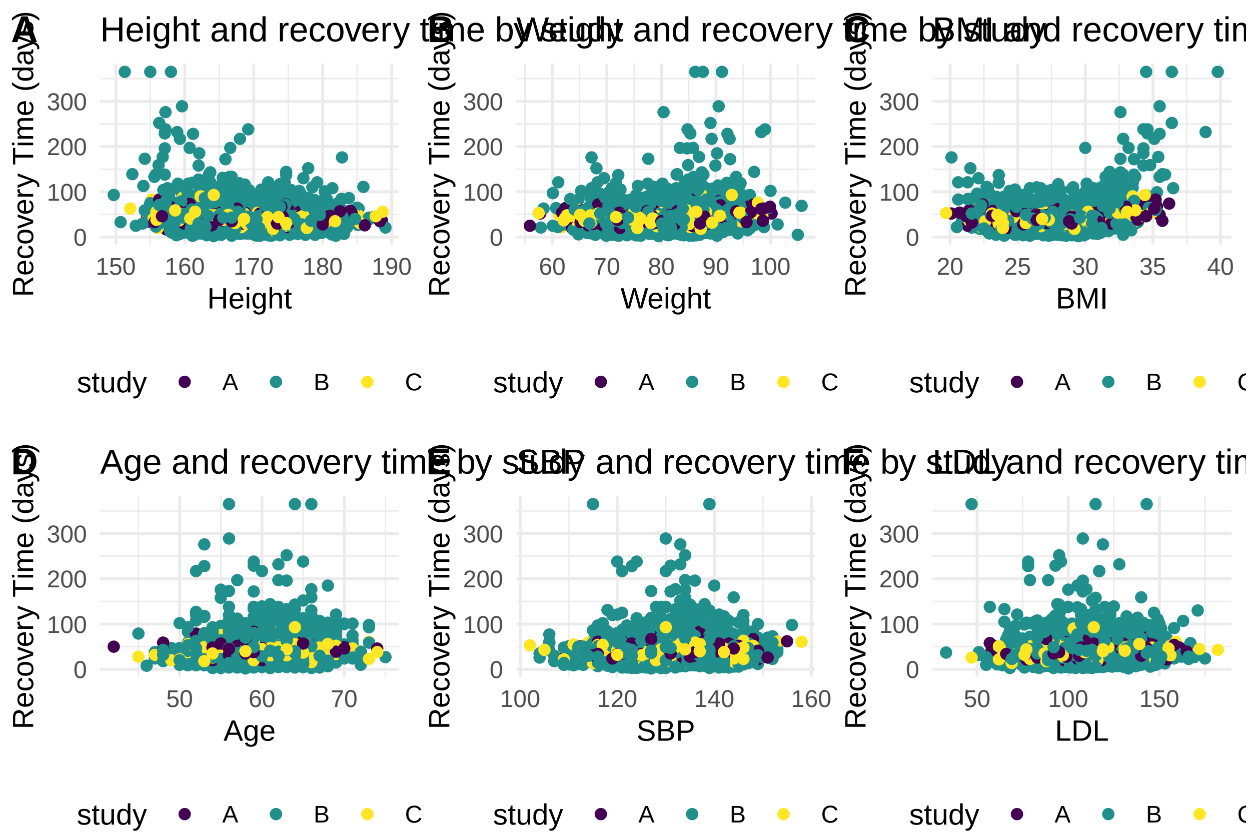
```
## # A tibble: 1 x 15
##     age gender  race smoking height weight   bmi hypertens~1 diabe~2   sbp   ldl
##   <int>  <int> <int>   <int>  <int>  <int> <int>       <int>   <int> <int> <int>
## 1     0      0     0       0      0      0     0           0       0     0     0
## # ... with 4 more variables: vaccine <int>, severity <int>, study <int>,
## #   recovery_time <int>, and abbreviated variable names 1: hypertension,
## #   2: diabetes
```

Relationship between predictor and outcome

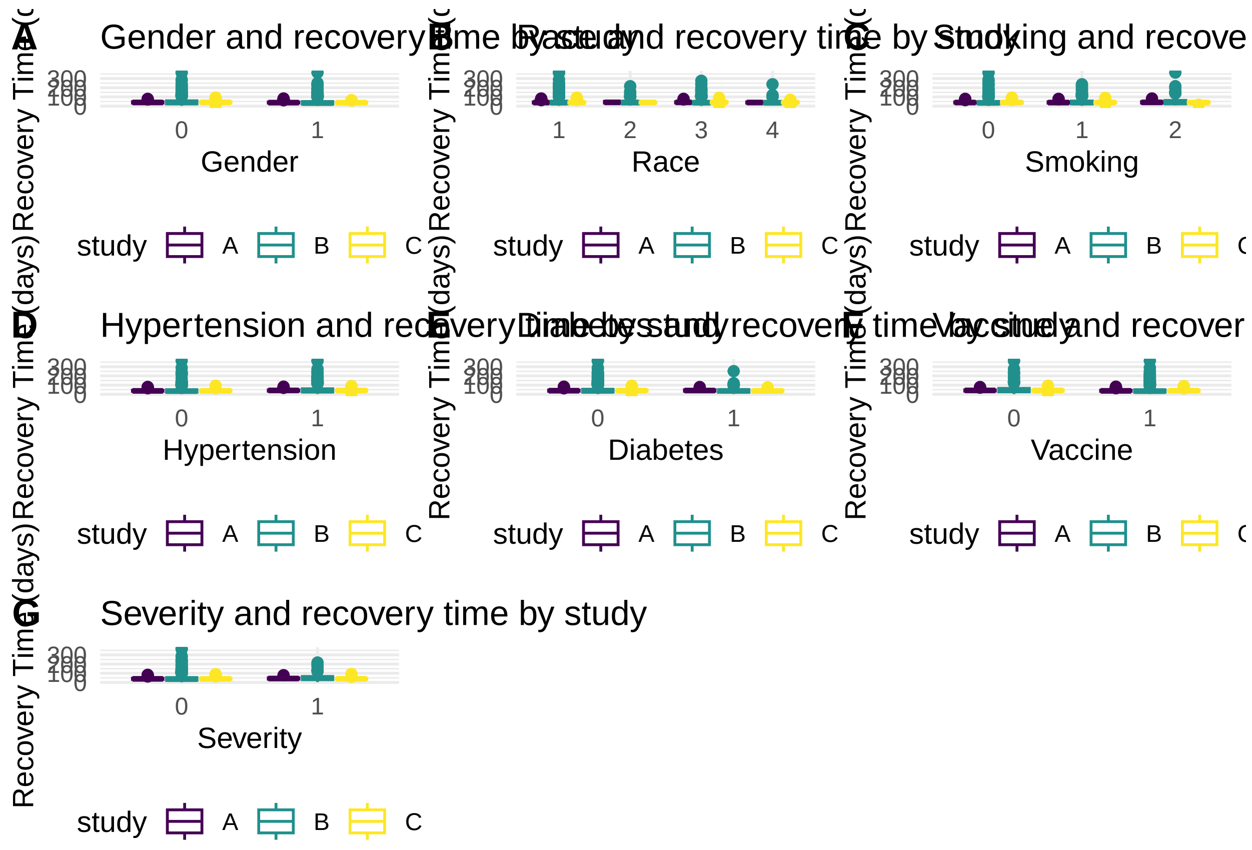
We want to see basic visualizations between predictor and outcome through a couple of different methods.

```
viz_point = function(name, title) {
  z = trainData %>%
    ggplot(aes(x = name, y = recovery_time, color = study)) +
    geom_point() +
    ggtitle(paste(title, "and recovery time by study")) +
    xlab(title) +
    ylab("Recovery Time (days)")
  z
}

age_recovery = viz_point(name = trainData$age, title = "Age")
height_recovery = viz_point(name = trainData$height, title = "Height")
weight_recovery = viz_point(name = trainData$weight, title = "Weight")
bmi_recovery = viz_point(name = trainData$bmi, title = "BMI")
sbp_recovery = viz_point(name = trainData$sbp, title = "SBP")
ldl_recovery = viz_point(name = trainData$ldl, title = "LDL")
```

```
viz_box = function(name, title) {
  z = trainData %>%
    ggplot(aes(x = name, y = recovery_time, color = study)) +
    geom_boxplot() +
    ggtitle(paste(title, "and recovery time by study")) +
    xlab(title) +
    ylab("Recovery Time (days)")
  z
}

gender_recovery = viz_box(name = trainData$gender, title = "Gender")
race_recovery = viz_box(name = trainData$race, title = "Race")
smoking_recovery = viz_box(name = trainData$smoking, title = "Smoking")
hypertension_recovery = viz_box(name = trainData$hypertension, title = "Hypertension")
diabetes_recovery = viz_box(name = trainData$diabetes, title = "Diabetes")
vaccine_recovery = viz_box(name = trainData$vaccine, title = "Vaccine")
severity_recovery = viz_box(name = trainData$severity, title = "Severity")

continuous_predictor_recovery = ggarrange(height_recovery, weight_recovery, bmi_recovery, age_recovery,
          labels = c("A", "B", "C", "D", "E", "F"),
          ncol = 3, nrow = 2)
continuous_predictor_recovery
```



```
categorical_predictor_recovery = ggarrange(gender_recovery, race_recovery, smoking_recovery, hypertensi
          labels = c("A", "B", "C", "D", "E", "F", "G"),
          ncol = 3, nrow = 3)
categorical_predictor_recovery
```



```
ggsave("plot/continuous_predictor_recovery.png", continuous_predictor_recovery, width = 10, height = 5,
ggsave("plot/categorical_predictor_recovery.png", categorical_predictor_recovery, width = 15, height =
```

Test multicolinearity

#Part I

Linear model training

We will now model train for linear methods. Each model has been tuned.

Linear regression

```
set.seed(6360)

# Using min rule
```

```r
fit.linear <- train(train_x, train_y,
                method = "lm",
                trControl = ctrl)

summary(fit.linear)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -67.939 -13.823  -1.638  10.275 261.794
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -2.735e+03  1.281e+02 -21.343  < 2e-16 ***
## age              5.568e-02  1.117e-01   0.499 0.618089
## gender1         -5.590e+00  8.825e-01  -6.335 2.75e-10 ***
## race2            1.708e-02  2.014e+00   0.008 0.993232
## race3           -3.230e-01  1.112e+00  -0.290 0.771468
## race4           -1.673e+00  1.490e+00  -1.122 0.261781
## smoking1         2.765e+00  9.884e-01   2.798 0.005182 **
## smoking2         6.590e+00  1.496e+00   4.405 1.09e-05 ***
## height           1.597e+01  7.511e-01  21.262  < 2e-16 ***
## weight          -1.716e+01  7.952e-01 -21.578  < 2e-16 ***
## bmi              5.170e+01  2.276e+00  22.709  < 2e-16 ***
## hypertension1    5.398e+00  1.456e+00   3.707 0.000214 ***
## diabetes1       -2.207e+00  1.251e+00  -1.764 0.077856 .
## sbp              2.572e-02  9.625e-02   0.267 0.789325
## ldl             -4.364e-02  2.327e-02  -1.875 0.060883 .
## vaccine1        -7.168e+00  9.043e-01  -7.927 3.19e-15 ***
## severity1        7.332e+00  1.488e+00   4.926 8.86e-07 ***
## studyB           3.111e+00  1.154e+00   2.695 0.007090 **
## studyC           3.417e-01  1.422e+00   0.240 0.810129
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 23.59 on 2858 degrees of freedom
## Multiple R-squared:  0.2598, Adjusted R-squared:  0.2551
## F-statistic: 55.73 on 18 and 2858 DF,  p-value: < 2.2e-16
```

```r
# Predict with test data
pred.linear <- predict(fit.linear, newdata = testData_matrix)

# Test error (MSE)
mean((pred.linear - test_y)^2)
```

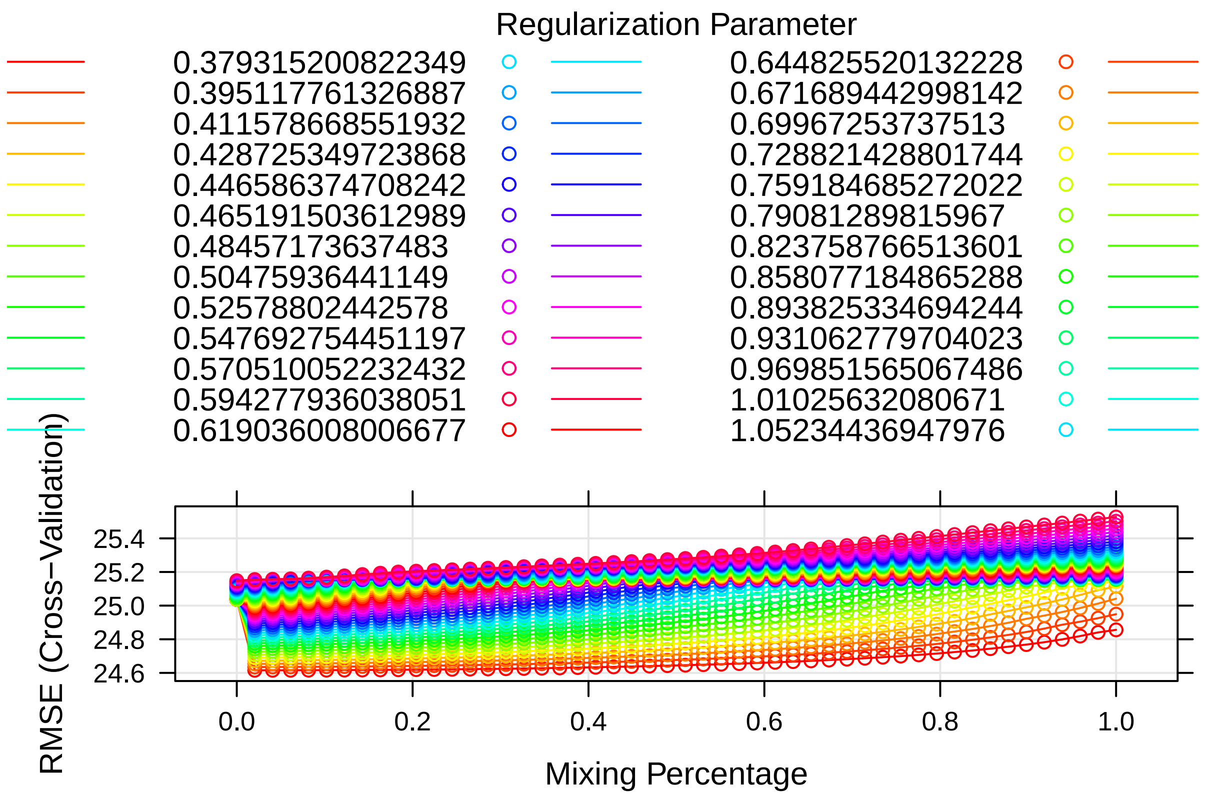
```
## [1] 531.1784
```

```r
# Final model
fit.linear$finalModel
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Coefficients:
##   (Intercept)            age        gender1          race2          race3
##    -2.735e+03      5.568e-02     -5.590e+00      1.708e-02     -3.230e-01
##         race4       smoking1       smoking2         height         weight
##    -1.673e+00      2.765e+00      6.590e+00      1.597e+01     -1.716e+01
##           bmi  hypertension1      diabetes1            sbp            ldl
##     5.170e+01      5.398e+00     -2.207e+00      2.572e-02     -4.364e-02
##      vaccine1      severity1         studyB         studyC
##    -7.168e+00      7.332e+00      3.111e+00      3.417e-01
```

Elastic net

```
set.seed(6360)

# Using min rule
fit.enet <- train(train_x, train_y,
                   method = "glmnet",
                   tuneGrid = expand.grid(alpha = seq(0, 1, length = 50),
                                          lambda = exp(seq(0.5, -1.5, length = 50))),
                  trControl = ctrl)

colors <- rainbow(25)
parings <- list(superpose.symbol = list(col = colors),
                    superpose.line = list(col = colors))

plot_enet <- plot(fit.enet, par.settings = parings)
plot_enet
```



```
# Find selected tuning parameter
fit.enet$bestTune
```

```
##         alpha    lambda
## 51 0.02040816 0.2231302
```

```
# Predict with test data
pred.enet <- predict(fit.enet, newdata = testData_matrix)

# Test error (MSE)
mean((pred.enet - test_y)^2)
```

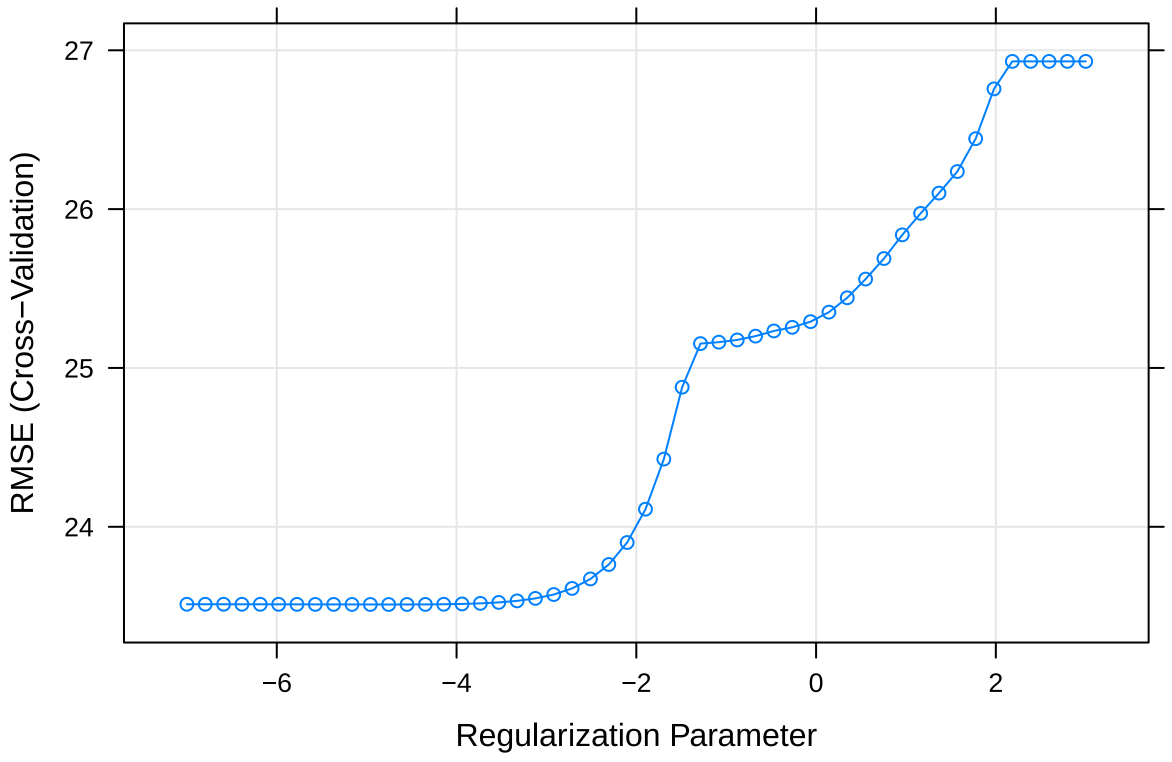
```
## [1] 606.0981
```

```
# Coefficients in the final model
coef(fit.enet$finalModel, fit.enet$bestTune$lambda)
```

```
## 19 x 1 sparse Matrix of class "dgCMatrix"
##                         s1
## (Intercept)   -5.115214e+02
## age            4.288675e-02
## gender1       -5.268363e+00
## race2          6.271261e-03
## race3         -2.636419e-01
## race4          -1.606718e+00
## smoking1        1.811935e+00
## smoking2        5.900617e+00
## height          2.860611e+00
## weight         -3.256120e+00
## bmi             1.186422e+01
## hypertension1   5.626020e+00
## diabetes1      -2.192995e+00
## sbp             1.724391e-02
## ldl            -4.586607e-02
## vaccine1       -7.559391e+00
## severity1       7.166227e+00
## studyB          3.677699e+00
## studyC          4.140095e-02
```

Lasso

```
set.seed(6360)

# Using min rule
fit.lasso <- train(train_x, train_y,
                   method = "glmnet",
                   tuneGrid = expand.grid(alpha = 1,
                                          lambda = exp(seq(3, -7, length = 50))),
                   trControl = ctrl)

plot_lasso <- plot(fit.lasso, xTrans = log)
plot_lasso
```



```r
# Find selected tuning parameter
fit.lasso$bestTune
```

```
##    alpha      lambda
## 12     1 0.008607666
```

```r
# Predict with test data
pred.lasso <- predict(fit.lasso, newdata = testData_matrix)

# Test error (MSE)
mean((pred.lasso - test_y)^2)
```

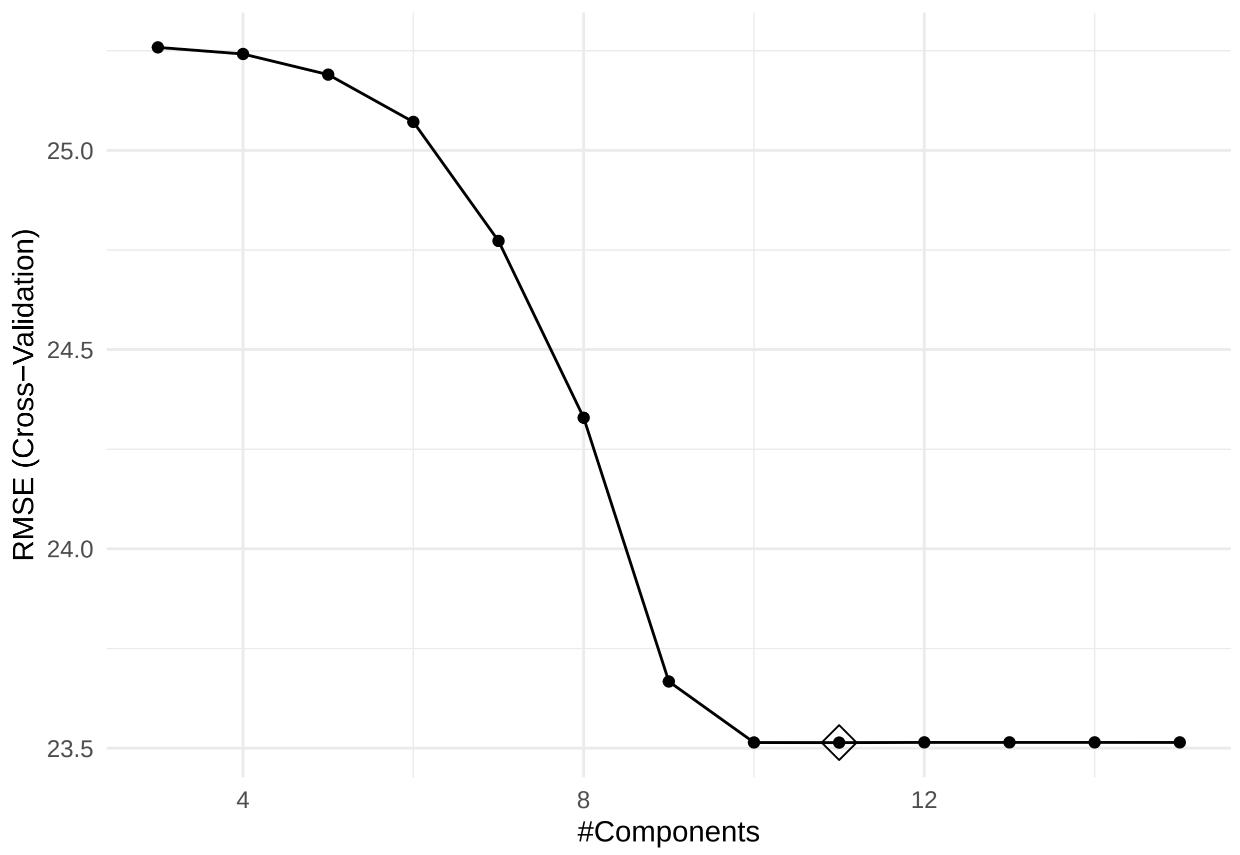
```
## [1] 532.1557
```

```r
# Coefficients in the final model
coef(fit.lasso$finalModel, fit.lasso$bestTune$lambda)
```

```
## 19 x 1 sparse Matrix of class "dgCMatrix"
##                          s1
## (Intercept)   -2.606921e+03
## age            5.365789e-02
## gender1       -5.557572e+00
## race2          .
## race3         -2.937586e-01
## race4          -1.635036e+00
## smoking1        2.688727e+00
## smoking2        6.518678e+00
## height          1.521479e+01
## weight         -1.635853e+01
## bmi             4.940613e+01
## hypertension1   5.408701e+00
## diabetes1      -2.185695e+00
## sbp             2.414711e-02
## ldl            -4.317507e-02
## vaccine1       -7.176565e+00
## severity1       7.293779e+00
## studyB          3.100969e+00
## studyC          2.708000e-01
```

PLS

```
set.seed(6360)

# Using min rule
fit.pls <- train(train_x, train_y,
                 method = "pls",
                 tuneGrid = data.frame(ncomp = 3:15),
                 trControl = ctrl,
                 preProcess = c("center", "scale"))

plot_pls <- ggplot(fit.pls, highlight = TRUE)
plot_pls
```



```r
# Find selected tuning parameter
fit.pls$bestTune
```

```
##   ncomp
## 9    11
```

```r
# Predict with test data
pred.pls <- predict(fit.pls, newdata = testData_matrix)

# Test error (MSE)
mean((pred.pls - test_y)^2)
```

```
## [1] 531.2747
```

```r
# Coefficients in the final model
coef(fit.pls$finalModel, fit.pls$bestTune$ncomp)
```

```
## , , 11 comps
##
##                     .outcome
## age               0.20206296
## gender1          -2.80274587
## race2             0.01429745
## race3            -0.07944635
## race4            -0.49910278
## smoking1          1.25818295
## smoking2          1.97224387
## height           95.70137908
## weight         -120.49014508
## bmi             142.12031382
## hypertension1     2.72328308
## diabetes1        -0.76394087
## sbp               0.21789882
## ldl              -0.86169721
## vaccine1         -3.50285549
## severity1         2.16436219
## studyB            1.51086488
## studyC            0.13287445
```

Non-linear models

We will now model train for non-linear methods.Each model has been tuned.

Generalized Additive Model(GAM)

```r
set.seed(6360)

# Using min rule
fit.gam <- train(train_x, train_y,
                 method = "gam",
                 tuneGrid = data.frame(method = "GCV.Cp", select = c(TRUE,FALSE)),
                 trControl = ctrl)

# Find selected tuning parameter
fit.gam$bestTune
```

```
##   select method
## 1  FALSE GCV.Cp
```

```r
# Predict with test data
pred.gam <- predict(fit.gam, newdata = testData_matrix)

# Test error (MSE)
mean((pred.gam - test_y)^2)
```

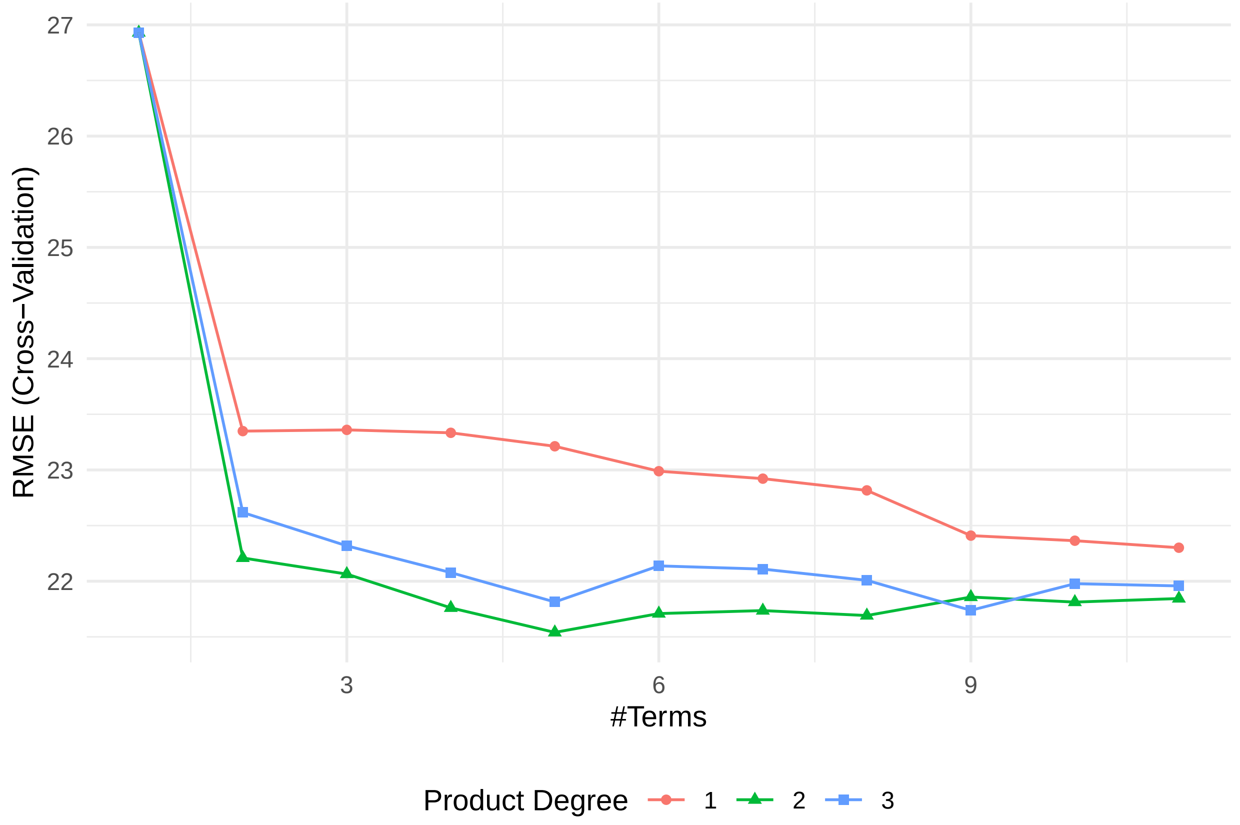
```
## [1] 489.3413
```

```r
# Coefficients in the final model
fit.gam$finalModel
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## .outcome ~ gender1 + race2 + race3 + race4 + smoking1 + smoking2 +
##     hypertension1 + diabetes1 + vaccine1 + severity1 + studyB +
##     studyC + s(age) + s(sbp) + s(ldl) + s(bmi) + s(height) +
##     s(weight)
##
## Estimated degrees of freedom:
## 2.17 2.04 1.14 7.84 6.35 3.43  total = 35.96
##
## GCV score: 479.4991
```

Multivariate Adaptive Regression Splines (MARS)

```r
set.seed(6360)

# Using min rule
fit.mars <- train(train_x, train_y,
                  method = "earth",
                  tuneGrid = expand.grid(degree = 1:3,
                                         nprune = 1:11),
                  trControl = ctrl)

plot_mars <- ggplot(fit.mars)
plot_mars
```



```
# Find selected tuning parameter
fit.mars$bestTune
```

```
##    nprune degree
## 16      5      2
```

```
# Predict with test data
pred.mars <- predict(fit.mars, newdata = testData_matrix)

# Test error (MSE)
mean((pred.mars - test_y)^2)
```

```
## [1] 490.8068
```

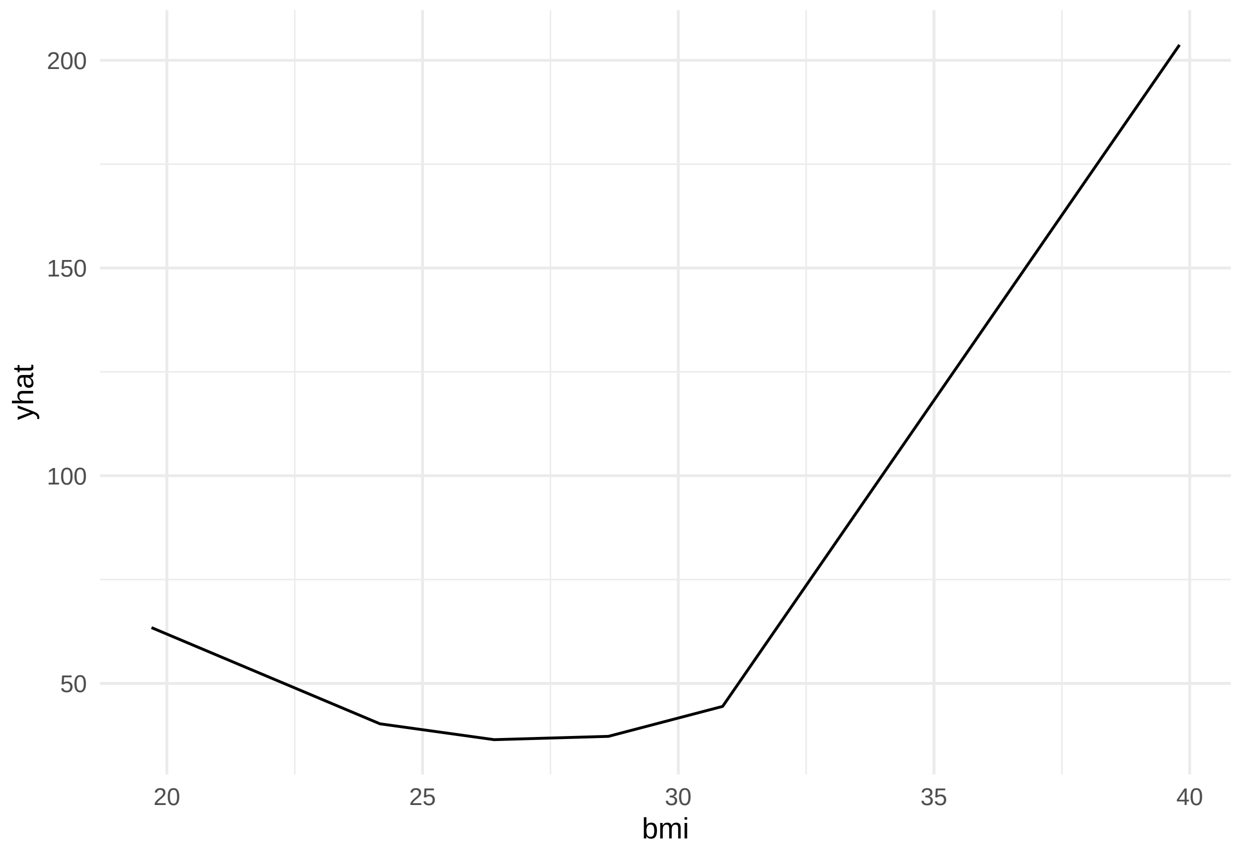
```
# Coefficients for final model
summary(fit.mars) %>% .$coefficients
```

```
##                                y
## (Intercept)          10.011086
## h(30.5-bmi)           5.183861
## h(bmi-30.5)*studyB   20.014232
## h(bmi-25)             5.541442
## vaccine1*studyB      -6.781929
```

```
fit.mars$finalModel
```

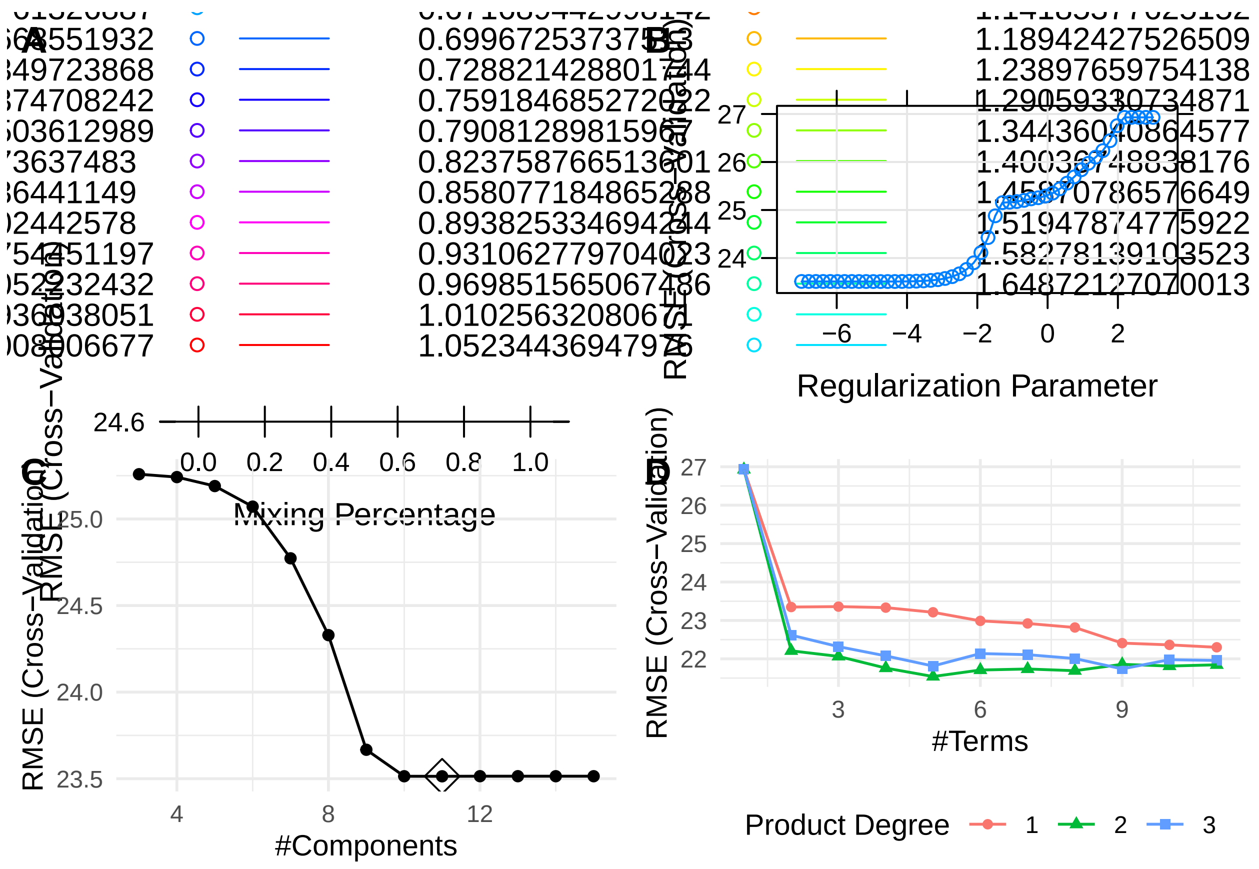
```
## Selected 5 of 26 terms, and 3 of 18 predictors (nprune=5)
## Termination condition: Reached nk 37
## Importance: bmi, studyB, vaccine1, age-unused, gender1-unused, ...
## Number of terms at each degree of interaction: 1 2 2
## GCV 465.9737    RSS 1330375    GRSq 0.3767223    RSq 0.3810491
```

```
# Partial dependence plot of BMI on recovery time from final model
continuous_pdp <- pdp::partial(fit.mars, pred.var = c("bmi"), grid.resolution = 10) %>% autoplot()
continuous_pdp
```



Figures for cross-validation for enet, lasso, pls, and mars.

```
ggarrange(plot_enet, plot_lasso, plot_pls, plot_mars,
          labels = c("A", "B", "C", "D"),
          ncol = 2, nrow = 2)
```



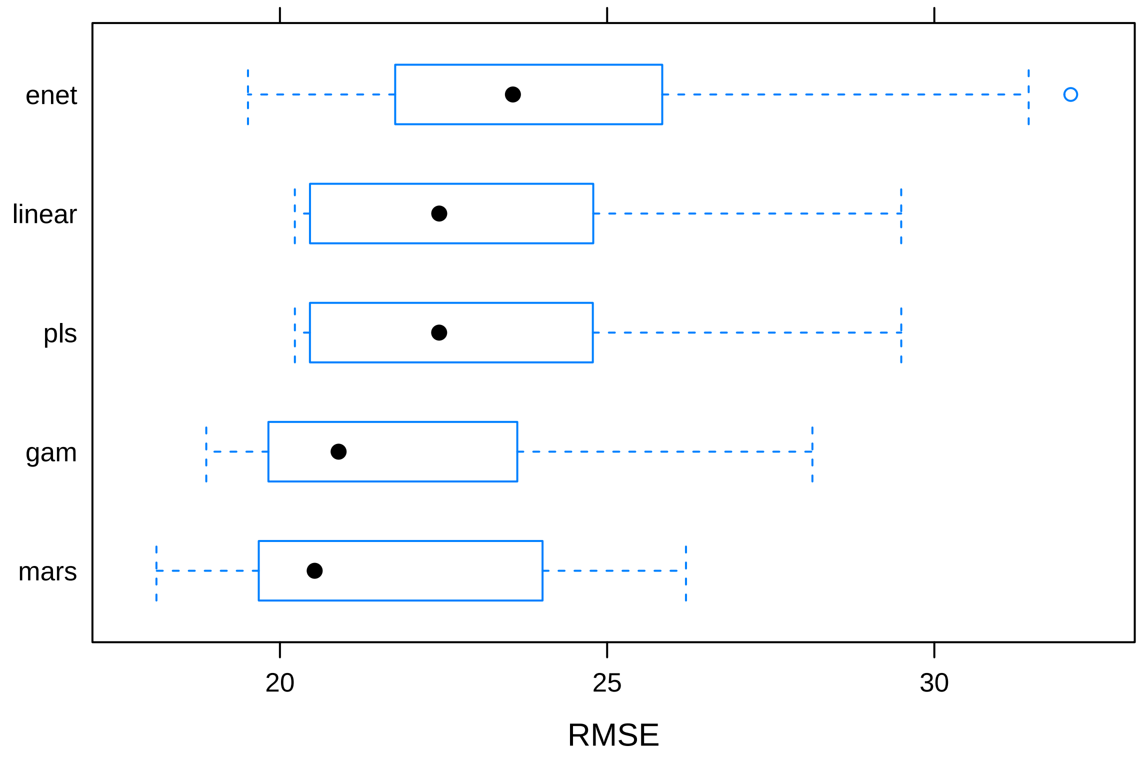
Comparing models for continuous recovery_time

We will now use resample to see which model produces the best predictive model for COVID-19 recovery time.

```
resample <- resamples(list(linear = fit.linear,
                           enet = fit.enet,
                           pls = fit.pls,
                           gam = fit.gam,
                           mars = fit.mars))
summary(resample)
```

```
##
## Call:
## summary.resamples(object = resample)
##
## Models: linear, enet, pls, gam, mars
## Number of resamples: 10
##
## MAE
##            Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## linear 14.54800 15.27895 15.99057 16.08786 16.96477 17.75319    0
## enet   14.19700 15.04929 15.94603 15.93757 16.91950 17.28806    0
## pls    14.54485 15.27935 15.98791 16.08754 16.96830 17.75511    0
## gam    13.69287 14.17302 15.04664 15.12169 16.03107 16.81801    0
## mars   13.35219 13.96394 14.39934 14.68957 15.47525 16.45813    0
##
## RMSE
##            Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## linear 20.22762 20.72375 22.43437 23.51446 24.69423 29.49377    0
## enet   19.51165 21.96415 23.56001 24.61546 25.66720 32.08463    0
## pls    20.22865 20.72183 22.43309 23.51395 24.69150 29.49455    0
## gam    18.87428 19.89204 20.89648 22.01722 23.50900 28.13665    0
## mars   18.11274 19.70780 20.53094 21.54015 23.46258 26.20489    0
##
## Rsquared
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## linear 0.1292126 0.2172170 0.2412512 0.2469056 0.2748234 0.3616900    0
## enet   0.0967574 0.1500365 0.1807555 0.1758062 0.2028757 0.2782463    0
## pls    0.1291943 0.2170408 0.2411909 0.2469396 0.2751816 0.3615804    0
## gam    0.1799479 0.3122485 0.3287883 0.3407728 0.4064722 0.4755038    0
## mars   0.1329910 0.2725725 0.3911641 0.3633788 0.4652147 0.5089855    0
```

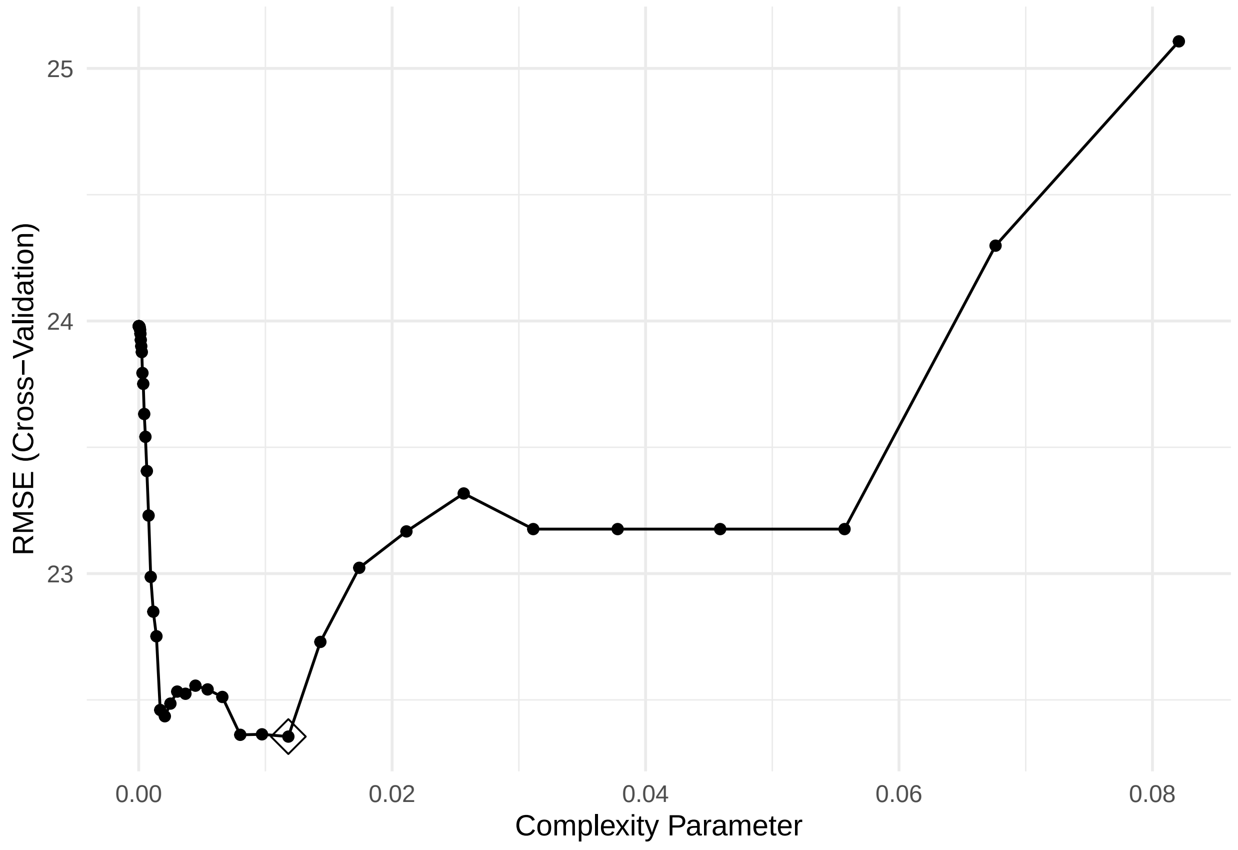
```
bwplot(resample, metric = "RMSE")
```



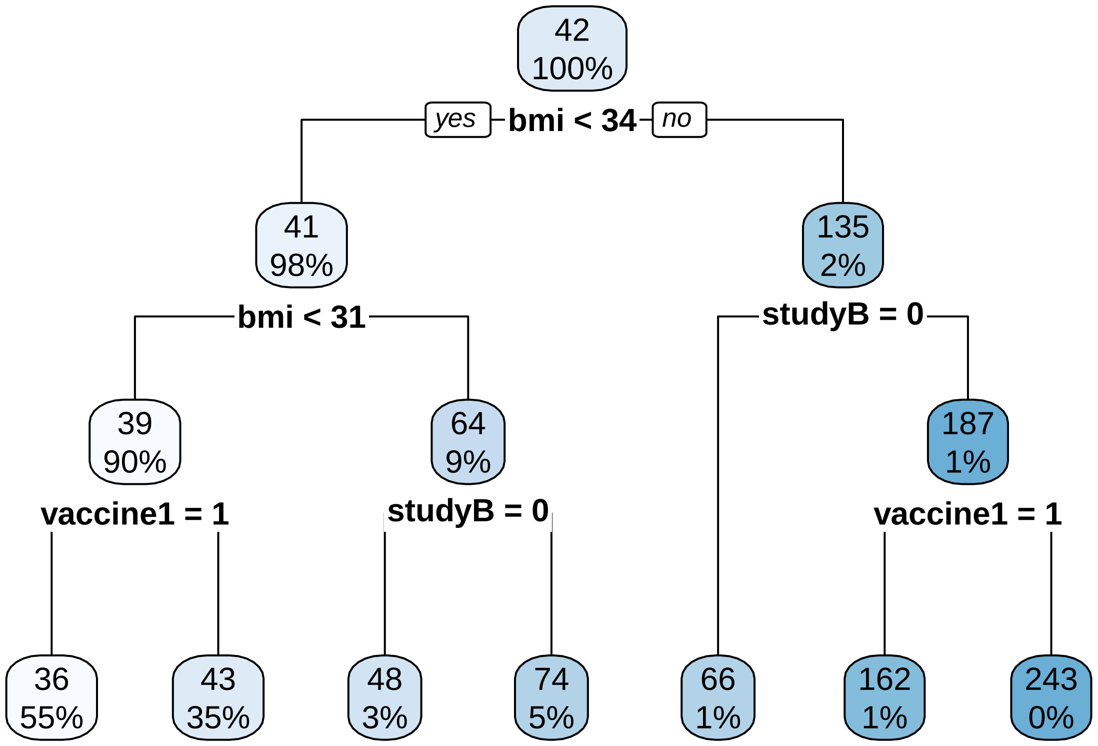
Regression tree

Using rpart

```
set.seed(6360)
rpart.fit <- train(recovery_time ~ . ,
                   trainData,
                   method = "rpart",
                   tuneGrid = data.frame(cp = exp(seq(-12,-2.5, length = 50))),
                   trControl = ctrl)

rpart_plot <- ggplot(rpart.fit, highlight = TRUE)
rpart_plot
```

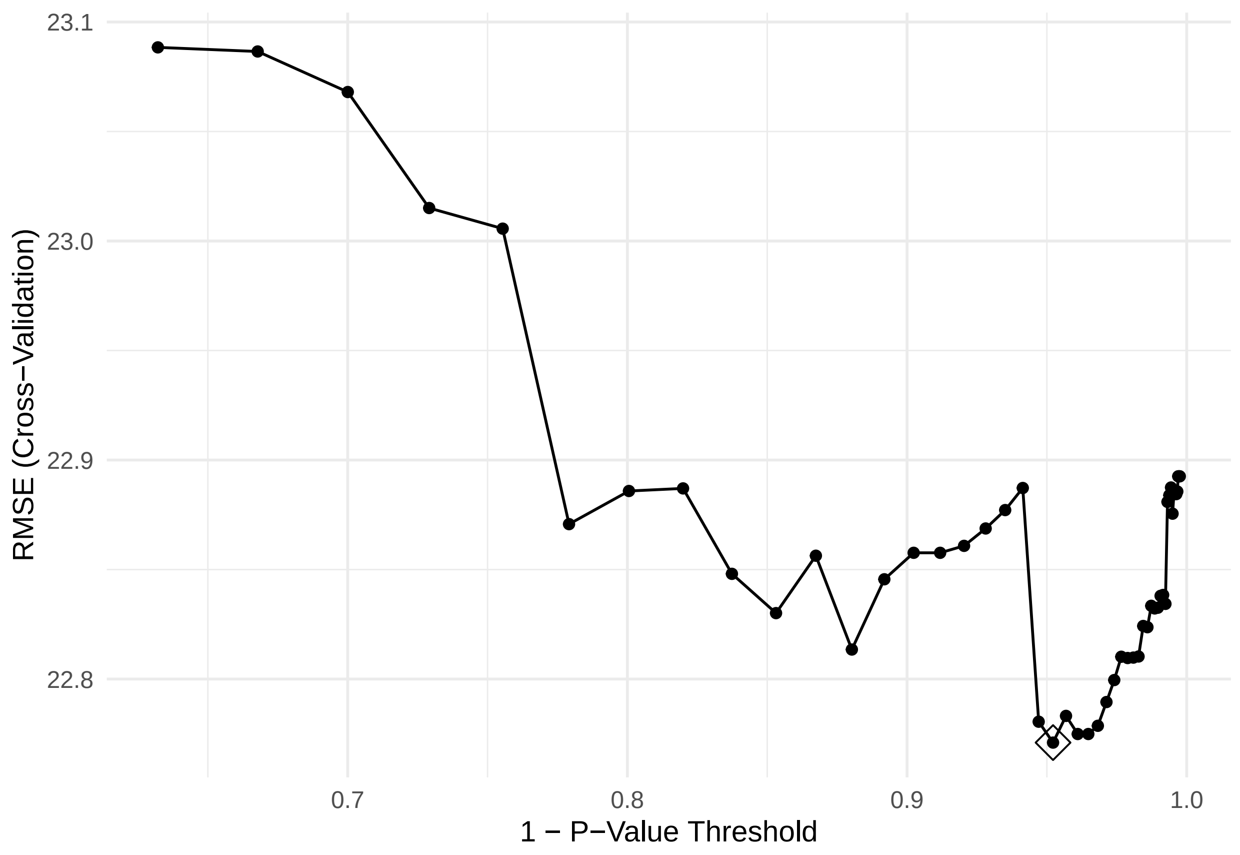


```
rpart.plot(rpart.fit$finalModel)
```

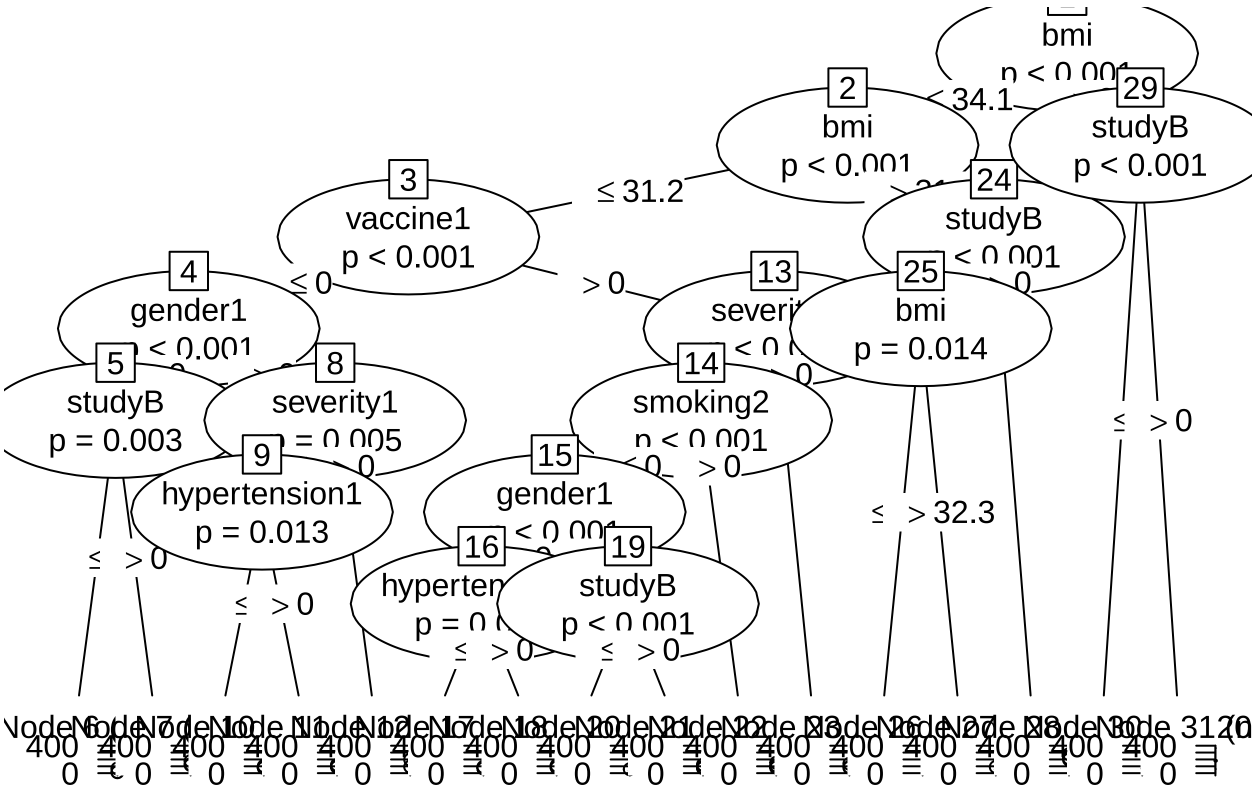


Using ctree

```
set.seed(6360)
ctree.fit <- train(recovery_time ~ . ,
                   trainData,
                   method = "ctree",
                   tuneGrid = data.frame(mincriterion = 1-exp(seq(-1, -6, length = 50))),
                   trControl = ctrl)

ctree_plot <- ggplot(ctree.fit, highlight = TRUE)
ctree_plot
```



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

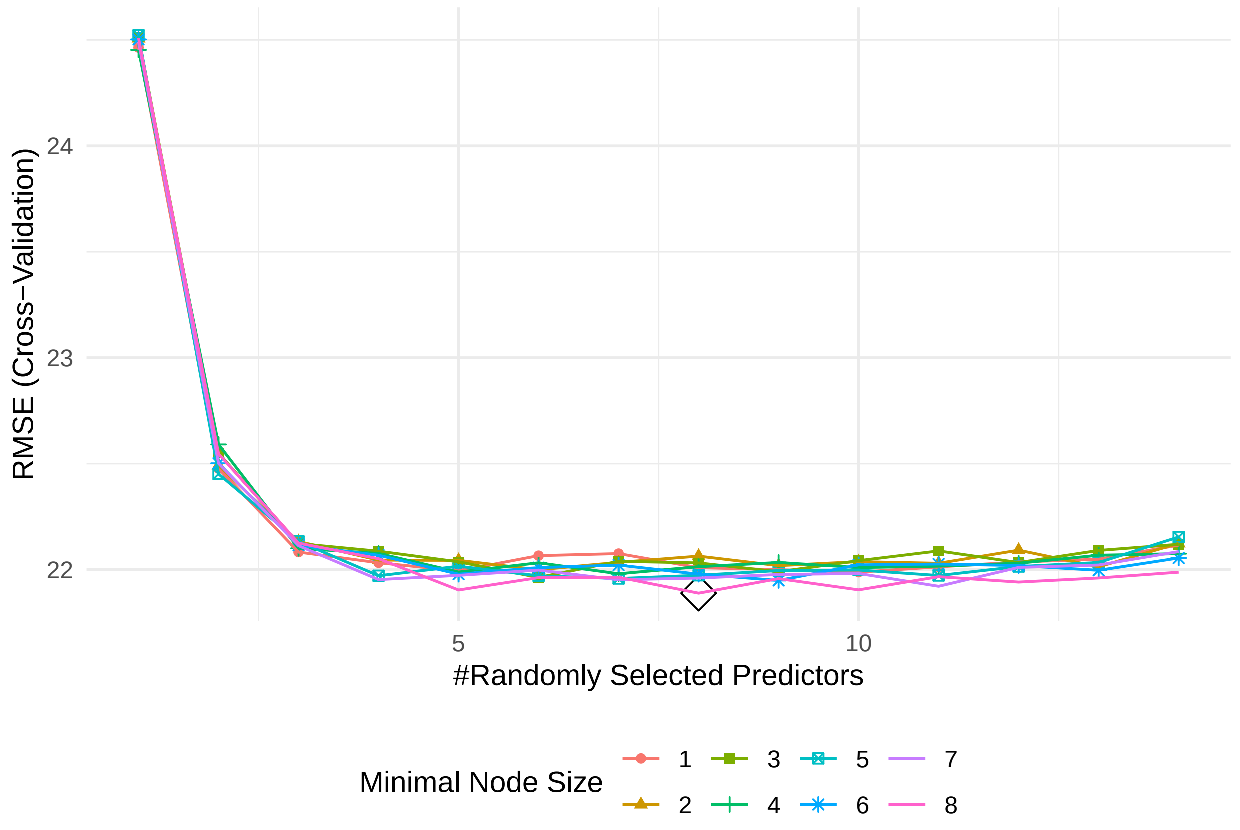


Random Forest

```
rf.grid <- expand.grid(mtry = 1:14,
                       splitrule = "variance",
                       min.node.size = 1:8)
set.seed(6360)
rf.fit <- train(recovery_time ~ . ,
                trainData,
                method = "ranger",
                tuneGrid = rf.grid,
                trControl = ctrl)

rf_plot <- ggplot(rf.fit, highlight = TRUE)
rf_plot
```

```
## Warning: The shape palette can deal with a maximum of 6 discrete values because
## more than 6 becomes difficult to discriminate; you have 8. Consider
## specifying shapes manually if you must have them.
```

```
## Warning: Removed 28 rows containing missing values (`geom_point()`).
```

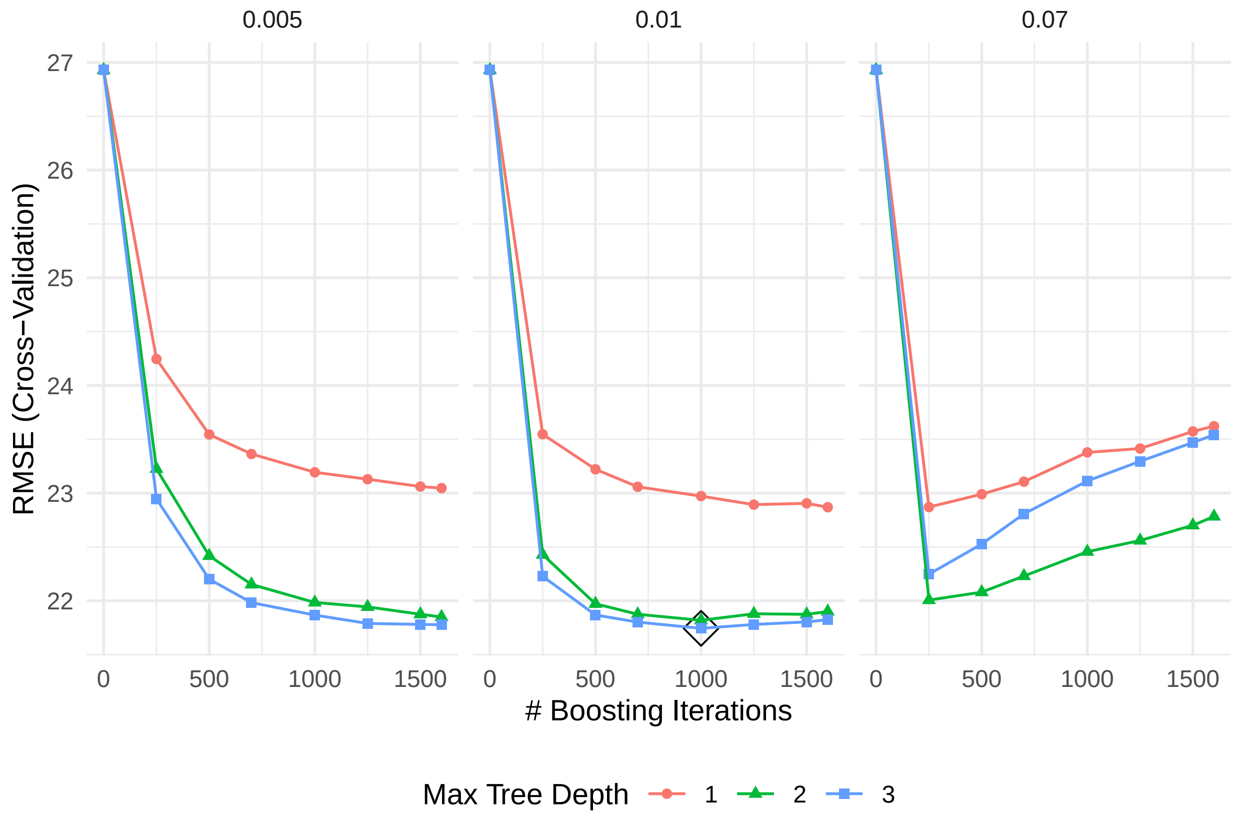


Boosting

```
set.seed(6360)
gbm.grid <- expand.grid(n.trees = c(0, 250, 500, 700, 1000, 1250, 1500, 1600),
                        interaction.depth = 1:3,
                        shrinkage = c(0.005,0.01, 0.07),
                        n.minobsinnode = c(1))
set.seed(6360)
gbm.fit <- train(recovery_time ~ . ,
                 trainData,
                 method = "gbm",
                 tuneGrid = gbm.grid,
                 trControl = ctrl,
                 verbose = FALSE)
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo,
## : There were missing values in resampled performance measures.
```

```
gbm_plot <- ggplot(gbm.fit, highlight = TRUE)
gbm_plot
```

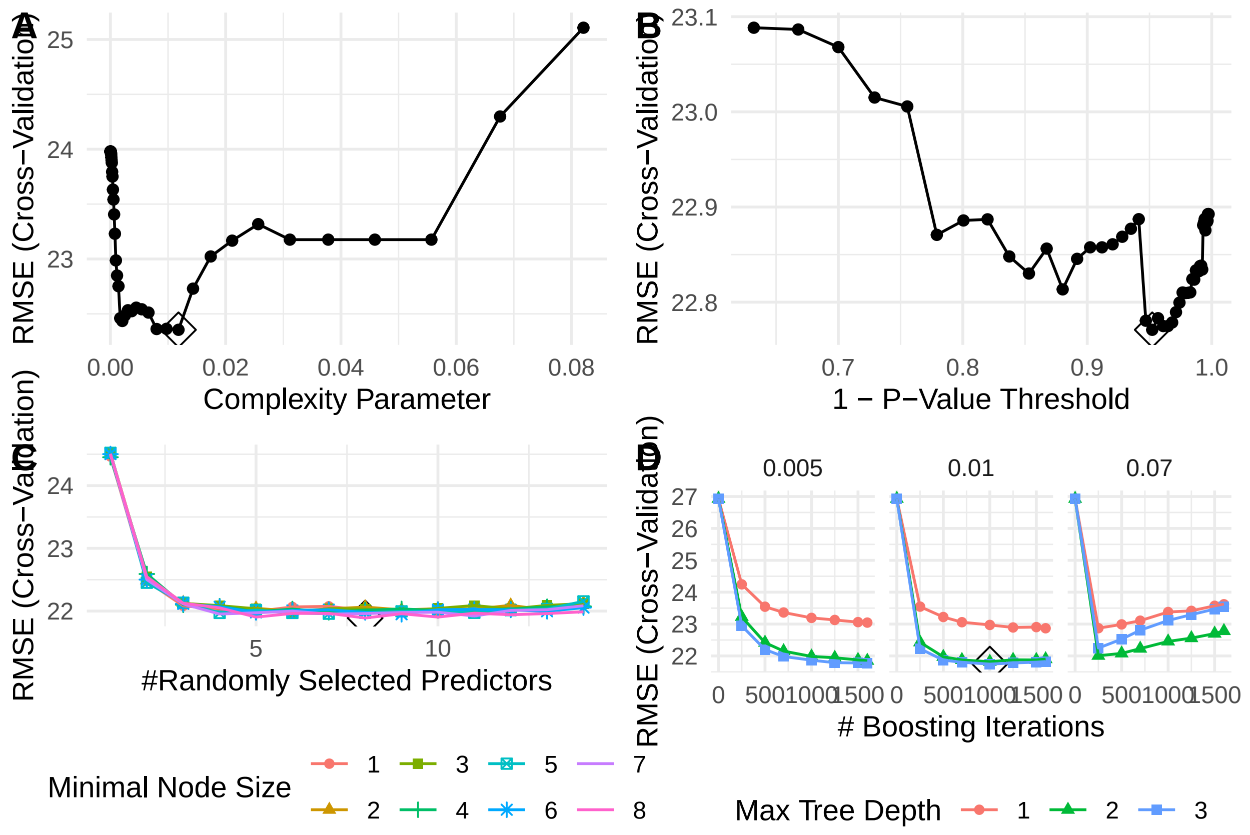


Figures for cross-validation for regression trees, random forest, and boosting.

```
ggarrange(rpart_plot, ctree_plot, rf_plot, gbm_plot,
          labels = c("A", "B", "C", "D"),
          ncol = 2, nrow = 2)
```

```
## Warning: The shape palette can deal with a maximum of 6 discrete values because
## more than 6 becomes difficult to discriminate; you have 8. Consider
## specifying shapes manually if you must have them.
```

```
## Warning: Removed 28 rows containing missing values (‘geom_point()‘).
```



Comparing all models

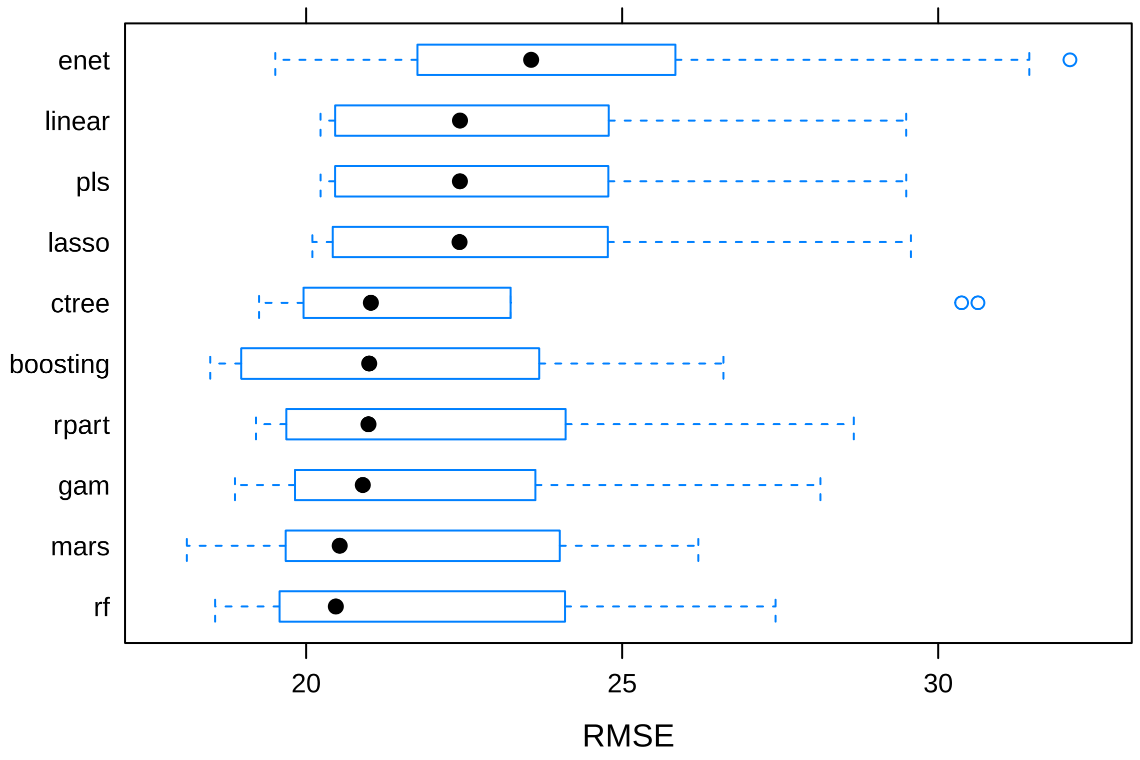
We will now compare all the models created to each other to find the best predictive model for COVID-19 recovery time.

```r
resample <- resamples(list(linear = fit.linear,
                           enet = fit.enet,
                           lasso = fit.lasso,
                           pls = fit.pls,
                           gam = fit.gam,
                           mars = fit.mars,
                           rpart = rpart.fit,
                           ctree = ctree.fit,
                           rf = rf.fit,
                           boosting = gbm.fit))
summary(resample)
```

```
##
## Call:
## summary.resamples(object = resample)
##
## Models: linear, enet, lasso, pls, gam, mars, rpart, ctree, rf, boosting
## Number of resamples: 10
##
```

```
## MAE
##              Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## linear   14.54800 15.27895 15.99057 16.08786 16.96477 17.75319    0
## enet     14.19700 15.04929 15.94603 15.93757 16.91950 17.28806    0
## lasso    14.49939 15.24152 15.94365 16.02539 16.92511 17.67081    0
## pls      14.54485 15.27935 15.98791 16.08754 16.96830 17.75511    0
## gam      13.69287 14.17302 15.04664 15.12169 16.03107 16.81801    0
## mars     13.35219 13.96394 14.39934 14.68957 15.47525 16.45813    0
## rpart    13.62218 14.12238 14.58765 14.98879 15.81284 17.12404    0
## ctree    13.42657 14.08968 14.58170 15.06997 15.65463 17.97128    0
## rf       13.27396 13.68775 14.40748 14.65368 15.61409 16.59782    0
## boosting 12.96487 13.75398 14.19358 14.49087 15.27066 16.53840    0
##
## RMSE
##              Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## linear   20.22762 20.72375 22.43437 23.51446 24.69423 29.49377    0
## enet     19.51165 21.96415 23.56001 24.61546 25.66720 32.08463    0
## lasso    20.09852 20.69110 22.42737 23.51069 24.67624 29.56878    0
## pls      20.22865 20.72183 22.43309 23.51395 24.69150 29.49455    0
## gam      18.87428 19.89204 20.89648 22.01722 23.50900 28.13665    0
## mars     18.11274 19.70780 20.53094 21.54015 23.46258 26.20489    0
## rpart    19.20685 19.87449 20.98629 22.35476 23.93906 28.66489    0
## ctree    19.25532 20.00246 21.02435 22.77098 22.97023 30.62911    0
## rf       18.55995 19.65328 20.47009 21.88895 23.63233 27.42705    0
## boosting 18.48297 19.18882 20.99771 21.74404 23.46361 26.60257    0
##
## Rsquared
##               Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## linear   0.1292126 0.2172170 0.2412512 0.2469056 0.2748234 0.3616900    0
## enet     0.0967574 0.1500365 0.1807555 0.1758062 0.2028757 0.2782463    0
## lasso    0.1291980 0.2157163 0.2425265 0.2468054 0.2754042 0.3625204    0
## pls      0.1291943 0.2170408 0.2411909 0.2469396 0.2751816 0.3615804    0
## gam      0.1799479 0.3122485 0.3287883 0.3407728 0.4064722 0.4755038    0
## mars     0.1329910 0.2725725 0.3911641 0.3633788 0.4652147 0.5089855    0
## rpart    0.1045308 0.2268274 0.3183282 0.3070220 0.3833201 0.4831504    0
## ctree    0.1290495 0.2247600 0.3110447 0.2918609 0.3726941 0.4046159    0
## rf       0.1758763 0.2936861 0.3668386 0.3427526 0.4072172 0.4631973    0
## boosting 0.1608023 0.3245875 0.3717188 0.3653391 0.4402029 0.5343460    0
```

```r
bwplot(resample, metric = "RMSE")
```



Part II

Create Data with Binary Outcome

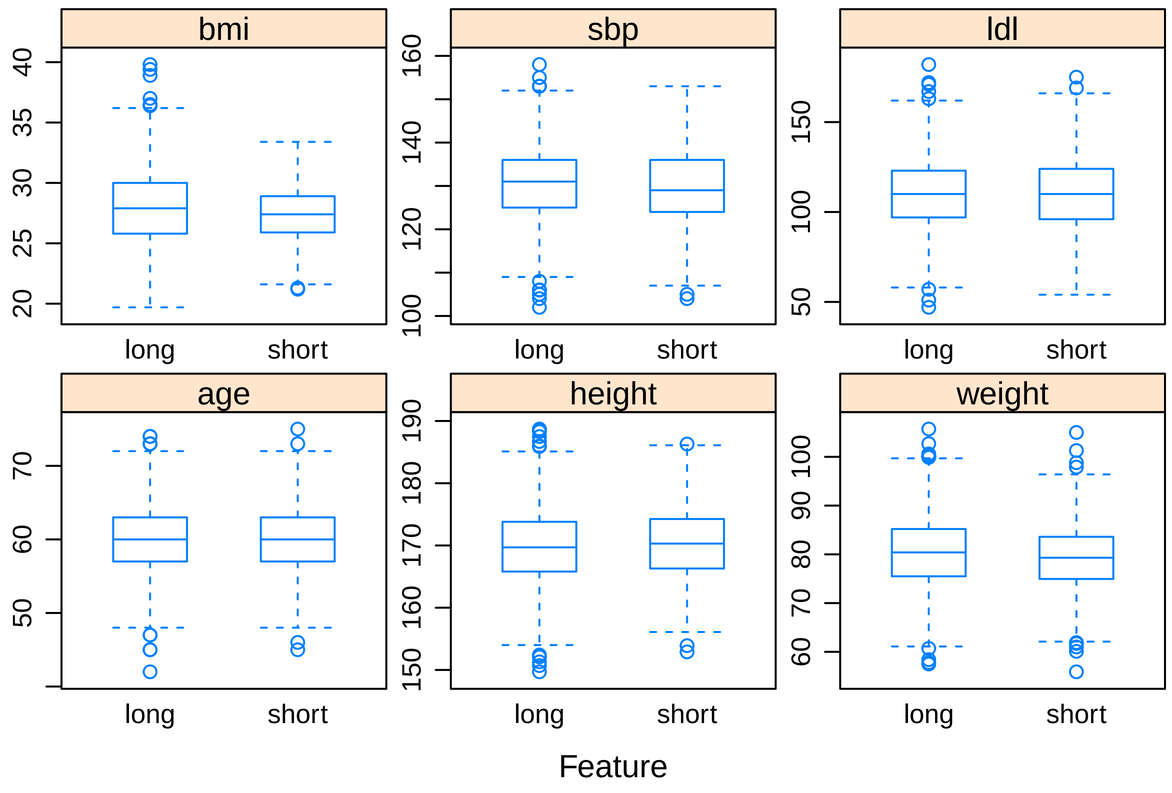
Instead of having recovery time as continuous, we will now binarize the outcome to short and long recover time using 30 as the cut-off point. Coefficients will only be outputted for the best model to allow for greater running speed.

```r
data_binary <-
  data %>%
  mutate(recovery_time = case_when(recovery_time > 30 ~ "long", TRUE ~ "short"),
         recovery_time = as.factor(recovery_time))

set.seed(6360)
trRows_binary <- createDataPartition(data_binary$recovery_time,
                                     p = .80,
                                     list = F)
# training data
trainData_binary <- data_binary[trRows_binary, ]
trainData_matrix_binary <- model.matrix(recovery_time~.,data_binary)[trRows_binary, ]
train_x_binary <- model.matrix(recovery_time~.,data_binary)[trRows_binary,-1]
train_y_binary <- data_binary$recovery_time[trRows_binary]
```

```
# test data
testData_binary <- data_binary[-trRows_binary, ]
testData_matrix_binary <- model.matrix(recovery_time~.,data_binary)[-trRows_binary,]
test_x_binary <- model.matrix(recovery_time~.,data_binary)[-trRows_binary,-1]
test_y_binary <- data_binary$recovery_time[-trRows_binary]

ctrl_binary <- trainControl(method = "cv",
                            summaryFunction = twoClassSummary,
                            classProbs = TRUE)
```

Visualization

```
featurePlot(x = trainData_binary[,c(1, 5:7, 10:11)],
            y = trainData_binary$recovery_time,
            scales = list(x = list(relation = "free"),
                          y = list(relation = "free")),
            plot = "box", pch = "|",
            auto.key = list(columns = 1))
```



Linear Models

We will now model train for linear methods. Each model has been tuned.

Logistic Regression

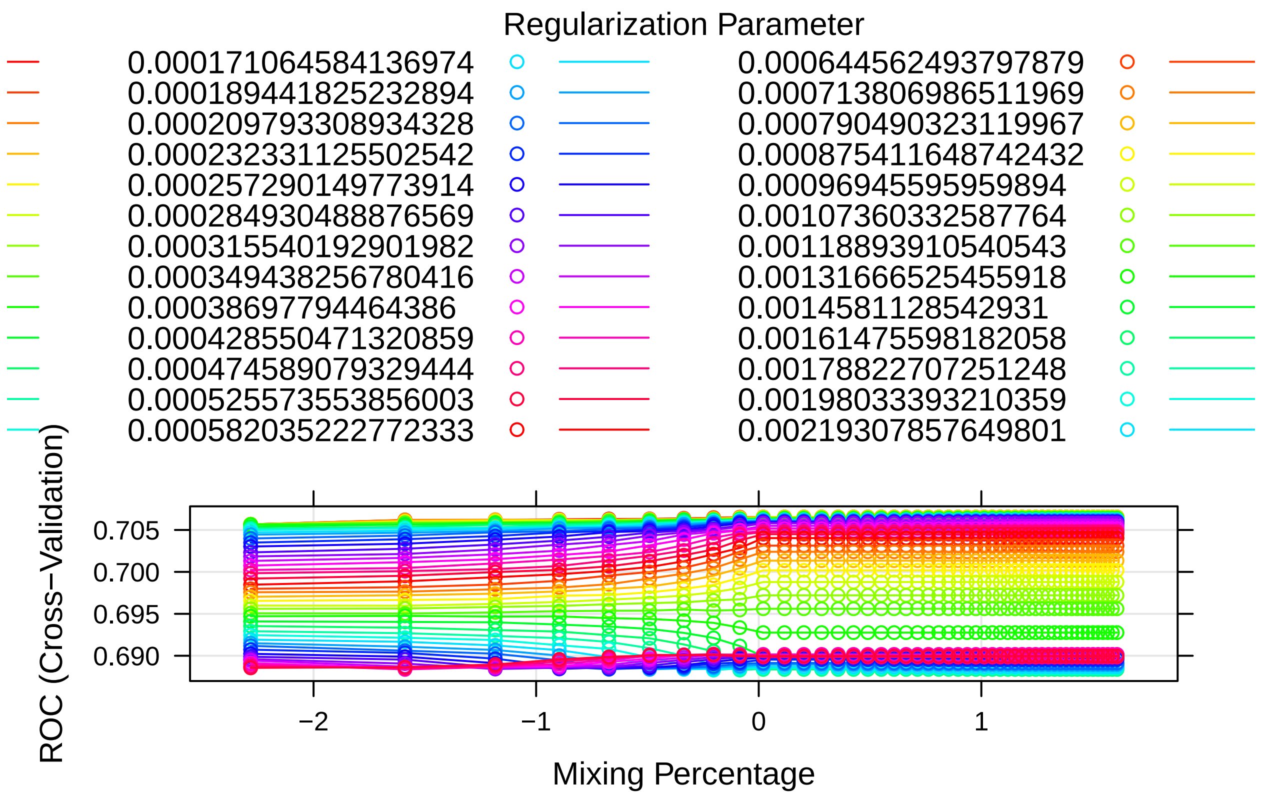
```
model.glm.binary <- train(train_x_binary,
                          train_y_binary,
                          method = "glm",
                          metric = "ROC",
                          trControl = ctrl_binary)
set.seed(6360)
pred_binary <- predict(model.glm.binary, newdata = test_x_binary,
                       type = "prob")[,1]
```

Penalized Logistic Regression

```
glmnGrid <- expand.grid(.alpha = seq(0, 5, length = 50),
                        .lambda = exp(seq(-10, -5, length = 50)))
set.seed(6360)
model.glmn.penalized <- train(x = train_x_binary,
                    y = train_y_binary,
                    method = "glmnet",
                    tuneGrid = glmnGrid,
                    metric = "ROC",
                    trControl = ctrl_binary)

model.glmn.penalized$bestTune
```

```
##        alpha      lambda
## 503 1.020408 5.56784e-05
```

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

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



GAM for Binary Outcome

```
set.seed(6360)
model.gam.binary <- train(train_x_binary,
                   train_y_binary,
                   method = "gam",
                   metric = "ROC",
                   trControl = ctrl_binary)

model.gam.binary$bestTune
```

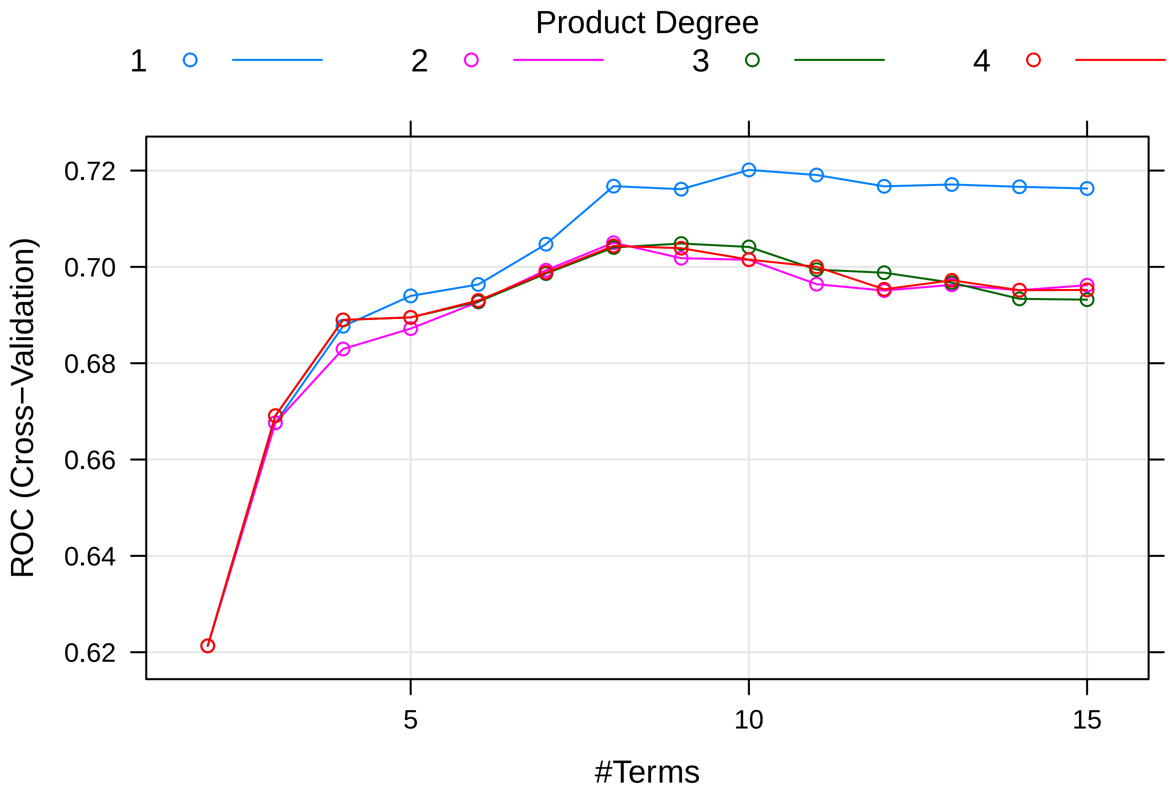
```
##   select method
## 2   TRUE GCV.Cp
```

```
model.gam.binary$finalModel
```

```
##
## Family: binomial
## Link function: logit
##
## Formula:
## .outcome ~ gender1 + race2 + race3 + race4 + smoking1 + smoking2 +
##     hypertension1 + diabetes1 + vaccine1 + severity1 + studyB +
##     studyC + s(age) + s(sbp) + s(ldl) + s(bmi) + s(height) +
##     s(weight)
##
## Estimated degrees of freedom:
## 0.4772 3.4094 0.0006 4.0037 0.0005 1.4770  total = 22.37
##
## UBRE score: 0.09831472
```

MARS for Binary Outcome

```
set.seed(6360)
model.mars.binary <- train(train_x_binary,
                           train_y_binary,
                           method = "earth",
                           tuneGrid = expand.grid(degree = 1:4,
                                                  nprune = 2:15),
                           metric = "ROC",
                           trControl = ctrl_binary)
mars_b_plot <- plot(model.mars.binary)
mars_b_plot
```



```
model.mars.binary$bestTune
```

```
##   nprune degree
## 9     10      1
```

```
test.mars <- predict(model.mars.binary, newdata = testData_matrix_binary, type = "prob")[,2]
test.mars_binary <- rep("long", length(test.mars))
test.mars_binary[test.mars > 0.5] <- "short"
mean(test.mars_binary != test_y_binary)
```

```
## [1] 0.2994429
```

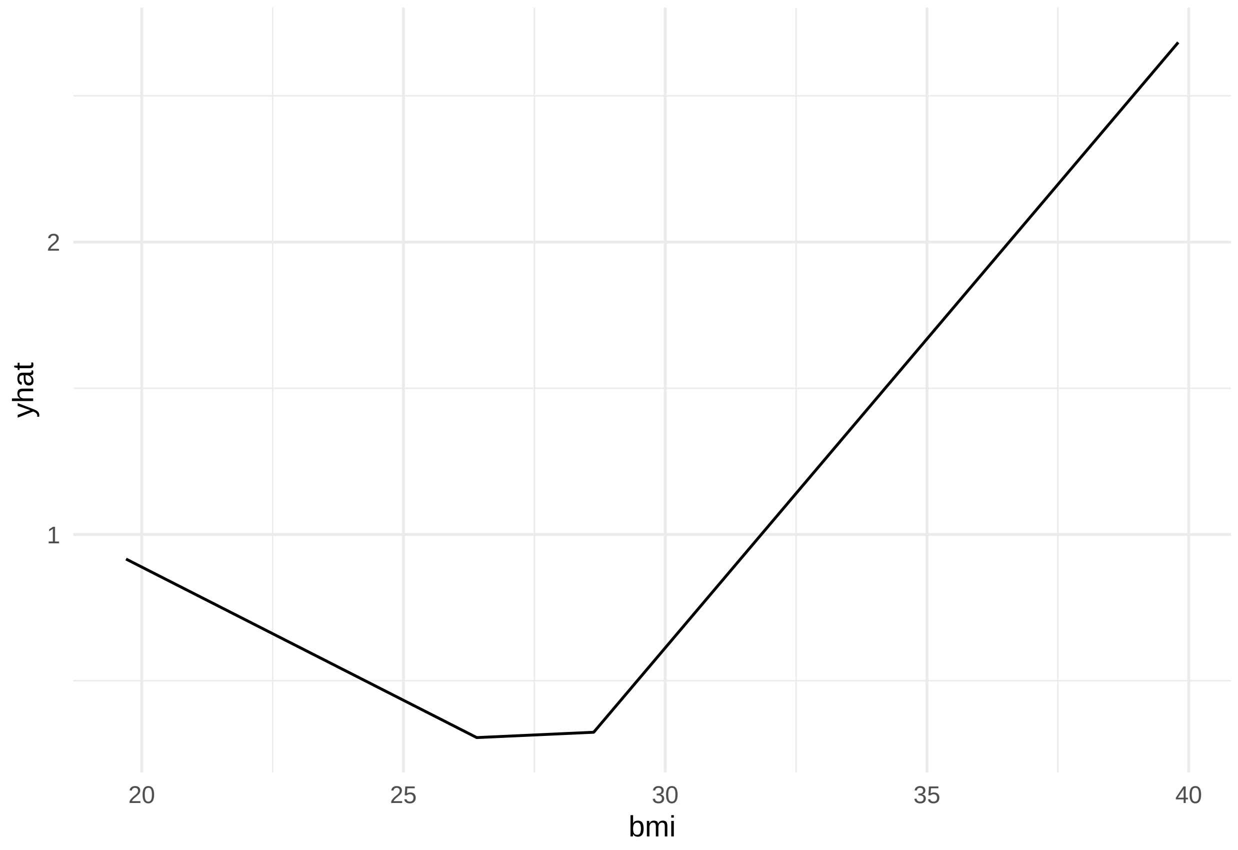
```
# Coefficients for final model
summary(model.mars.binary) %>% .$coefficients
```

```
##                      short
## (Intercept)     0.24907240
## studyB          0.21274991
## h(bmi-27.9)    -0.06183408
## h(27.9-bmi)    -0.03077706
## vaccine1        0.10416560
## severity1      -0.13608455
## gender1         0.07264080
## hypertension1 -0.06937501
## smoking2       -0.08566066
## smoking1       -0.05533681
```

```
model.mars.binary$finalModel
```

```
## GLM (family binomial, link logit):
##  nulldev   df       dev   df   devratio     AIC iters converged
##  3547.25 2875   3140.43 2866      0.115    3160     4         1
##
## Earth selected 10 of 19 terms, and 8 of 18 predictors (nprune=10)
## Termination condition: RSq changed by less than 0.001 at 19 terms
## Importance: studyB, bmi, vaccine1, severity1, gender1, hypertension1, ...
## Number of terms at each degree of interaction: 1 9 (additive model)
## Earth GCV 0.1887688    RSS 535.7496    GRSq 0.1133792    RSq 0.1244464
```

```
# Partial dependence plot of BMI on recovery time from final model
binary_pdp <- pdp::partial(model.mars.binary, pred.var = c("bmi"), grid.resolution = 10) %>% autoplot()
binary_pdp
```



LDA

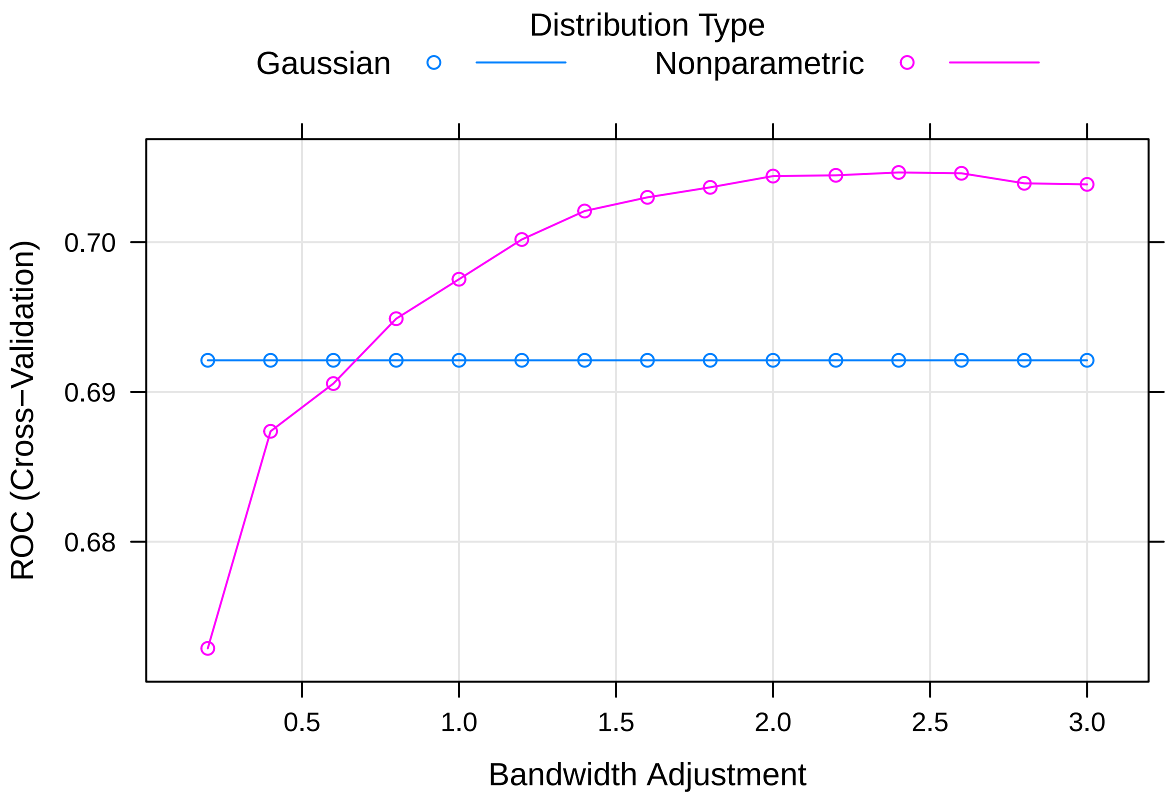
```r
set.seed(6360)
model.lda.binary <- train(train_x_binary,
                          train_y_binary,
                          method = "lda",
                          metric = "ROC",
                          trControl = ctrl_binary)
```

QDA

```r
set.seed(6360)
model.qda.binary <- train(train_x_binary,
                          train_y_binary,
                          method = "qda",
                          metric = "ROC",
                          trControl = ctrl_binary)
```

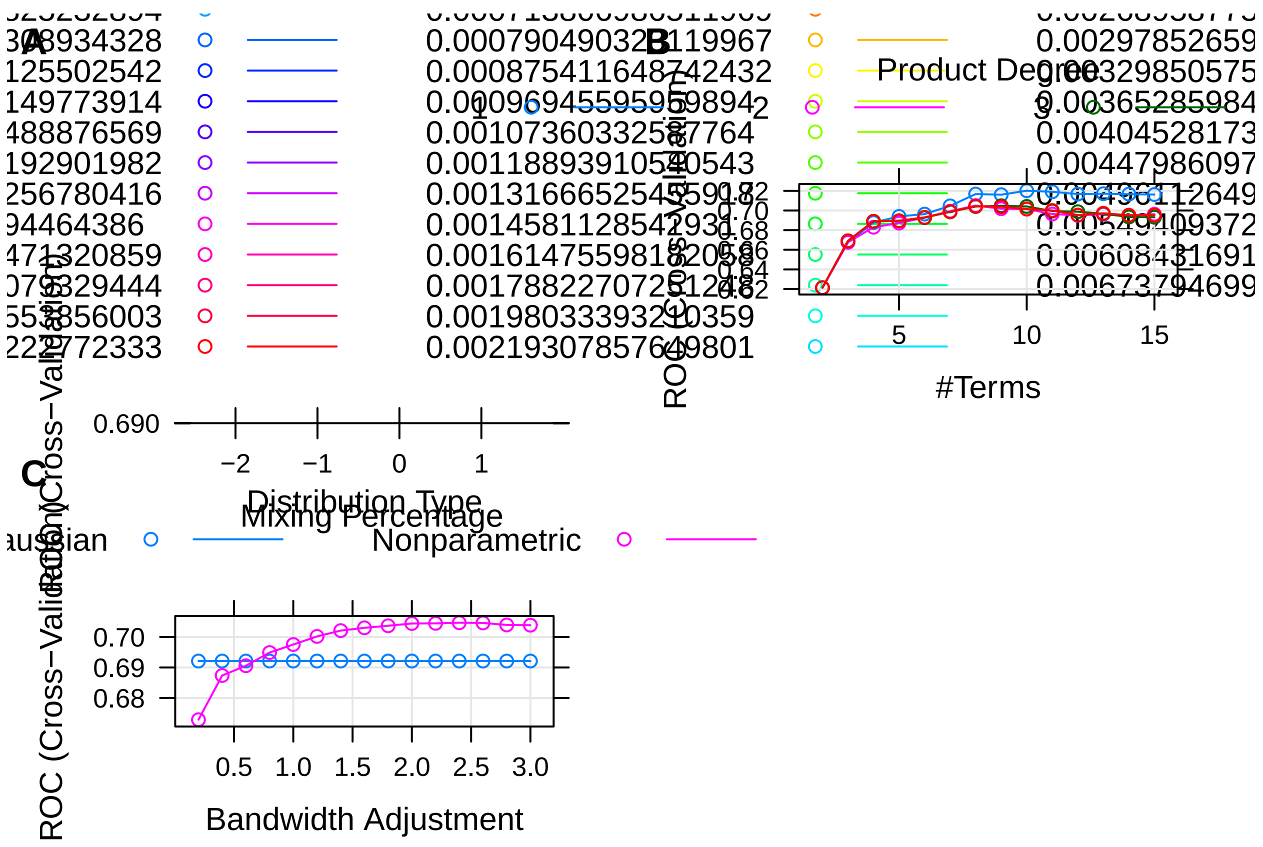
Naive Bayes

```
nbGrid <- expand.grid(usekernel = c(FALSE,TRUE),
                      fL = 1,
                      adjust = seq(.2, 3, by = .2))
set.seed(6360)
model.nb.binary <- train(train_x_binary,
                         train_y_binary,
                         method = "nb",
                         tuneGrid = nbGrid,
                         metric = "ROC",
                         trControl = ctrl_binary)
nb_plot <- plot(model.nb.binary)
nb_plot
```



Figures for cross-validation for enet, lasso, pls, and mars.

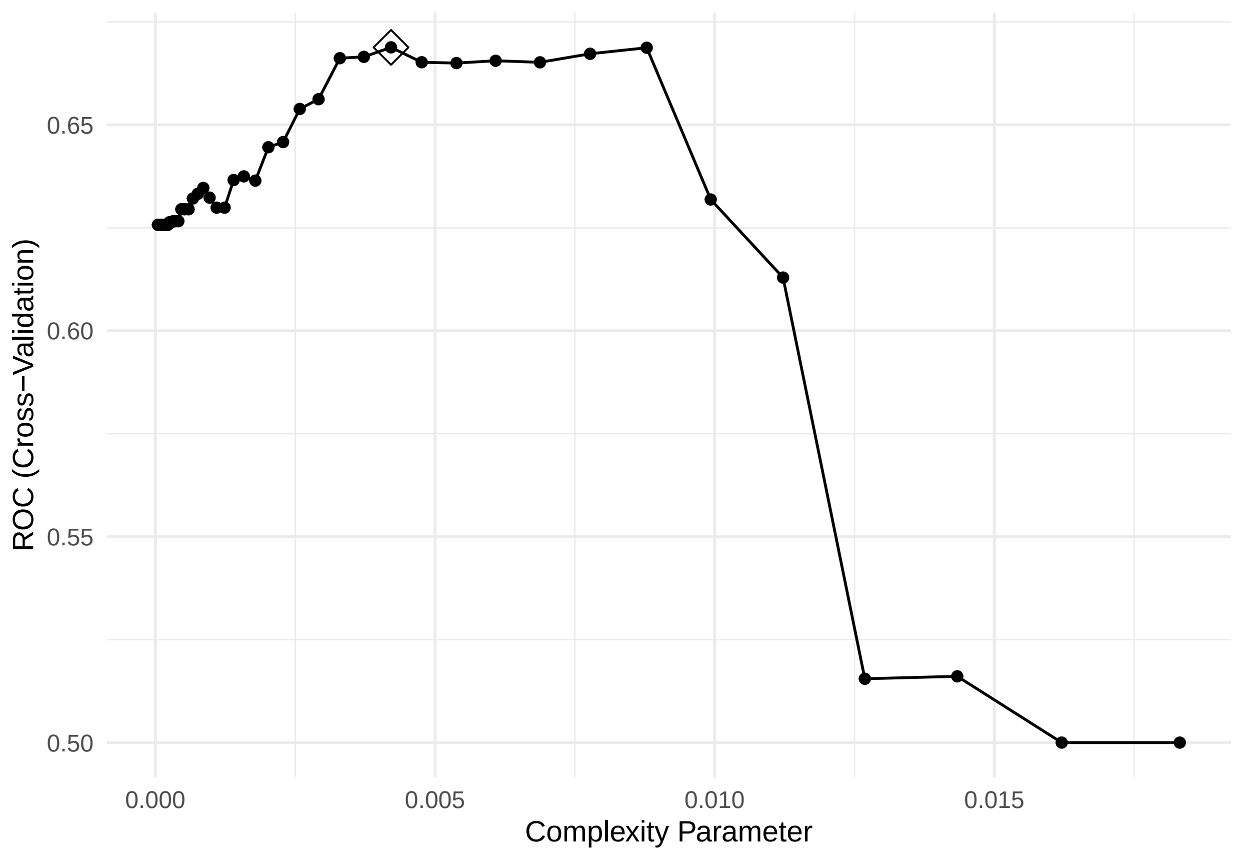
```
ggarrange(penalized_plot, mars_b_plot, nb_plot,
          labels = c("A", "B", "C"),
          ncol = 2, nrow = 2)
```



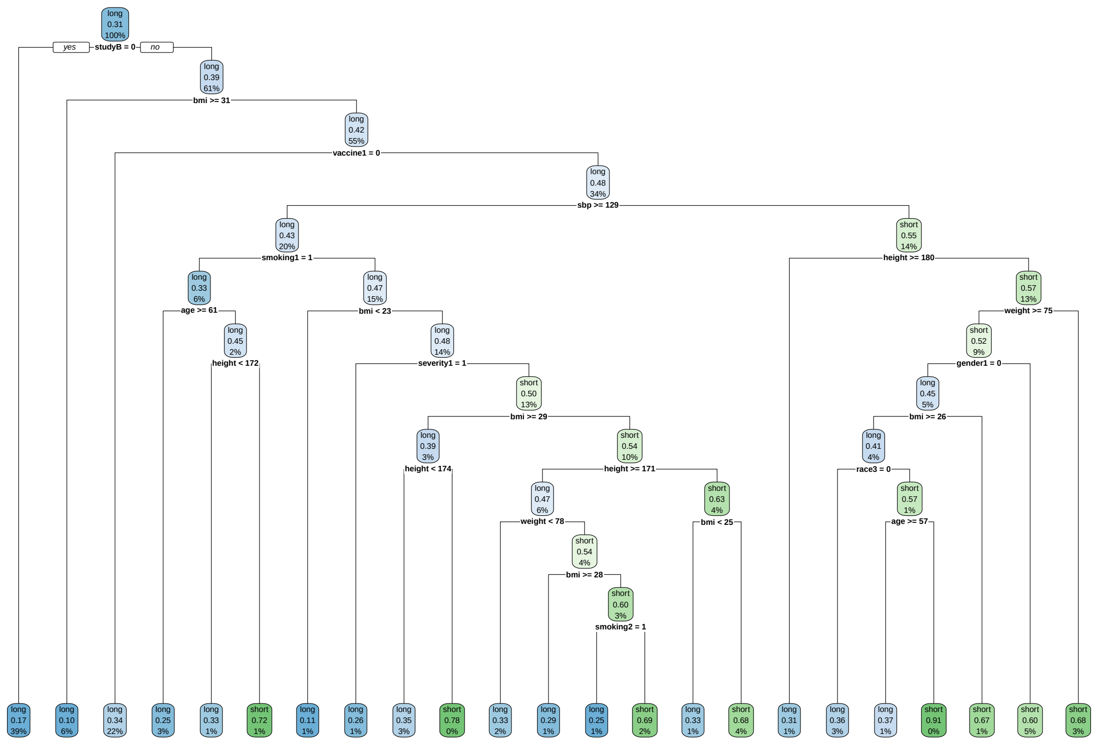
Classification Trees

Using rpart

```r
set.seed(6360)
rpart.fit.binary <- train(recovery_time ~ . ,
                          data_binary,
                          subset = trRows_binary,
                          method = "rpart",
                          tuneGrid = data.frame(
                            cp = exp(seq(-10,-4, len = 50))),
                          trControl = ctrl_binary,
                          metric = "ROC")
binary_rpart <- ggplot(rpart.fit.binary, highlight = TRUE)
binary_rpart
```

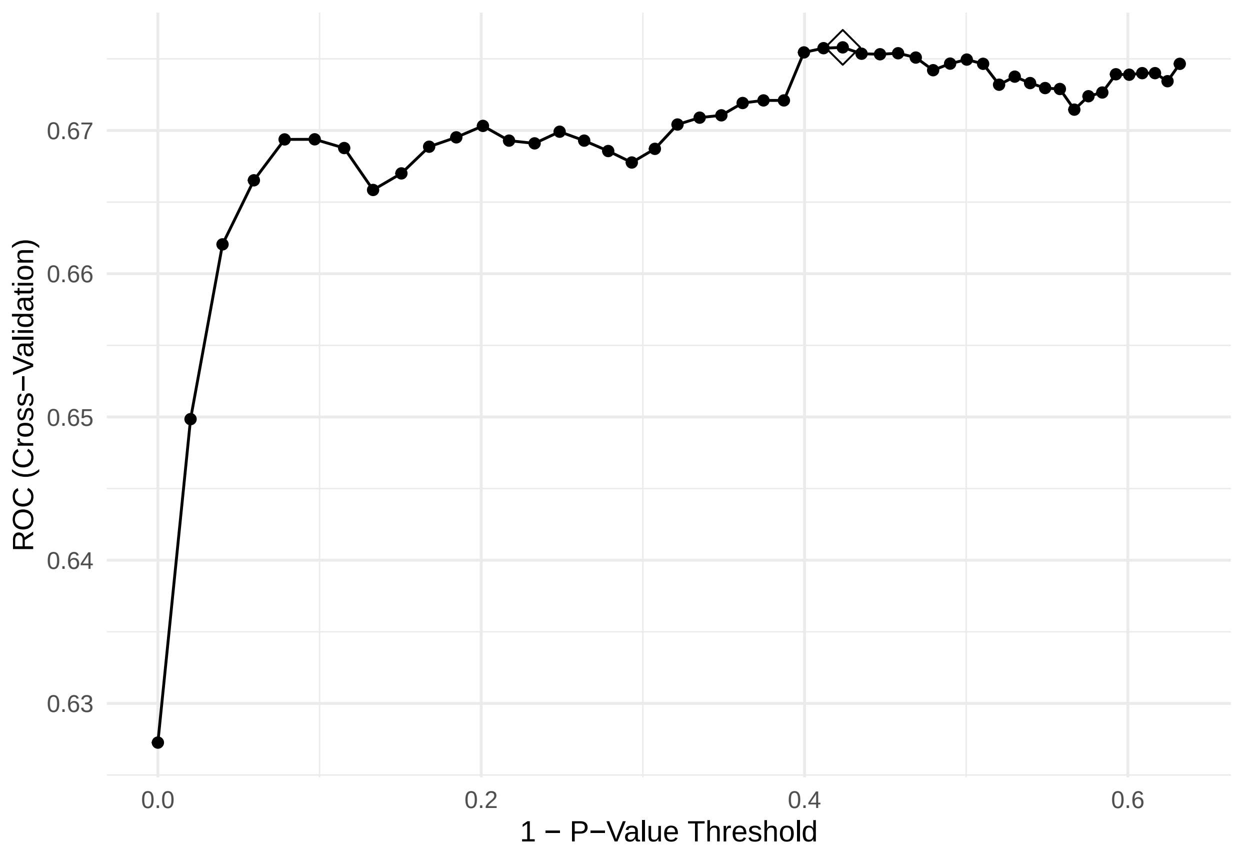


```
rpart.plot::rpart.plot(rpart.fit.binary$finalModel)
```

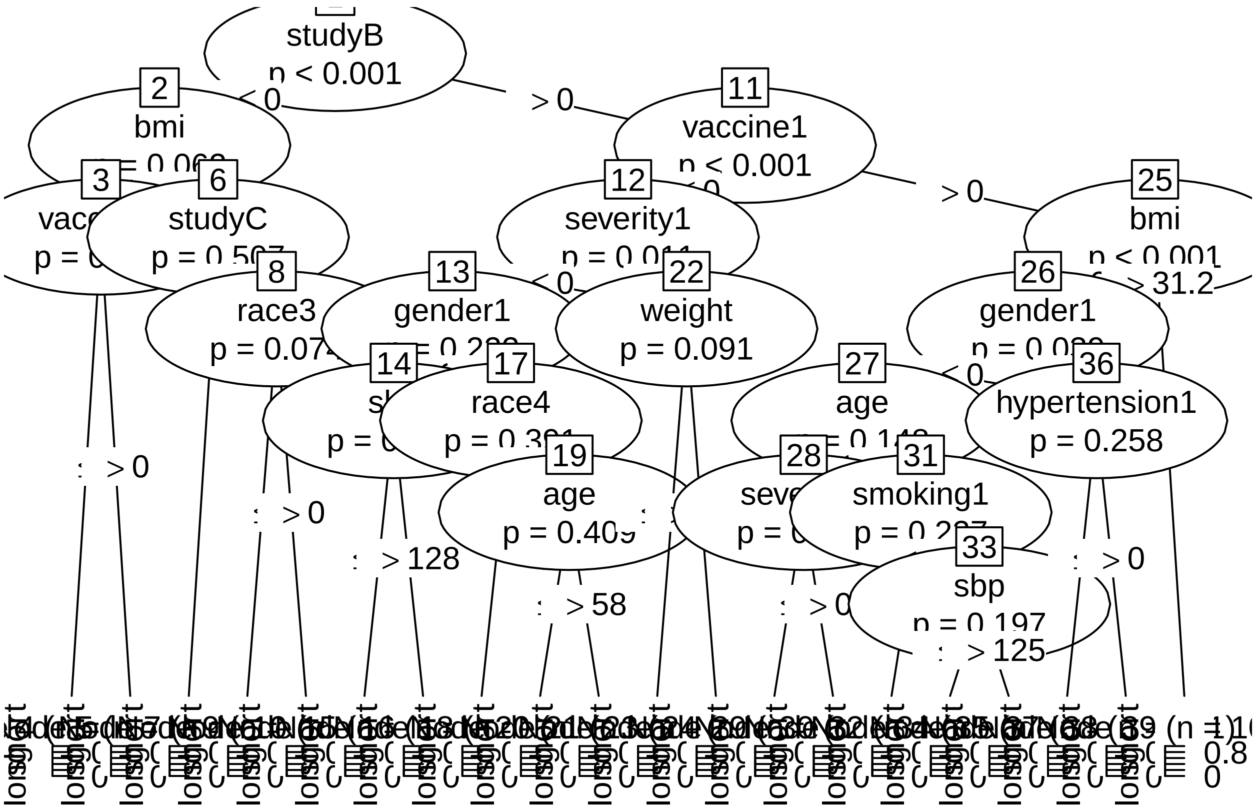


Using ctree

```
set.seed(6360)
ctree.fit.binary <- train(recovery_time ~ . ,
                          data_binary,
                          subset = trRows_binary,
                          method = "ctree",
                          tuneGrid = data.frame(
                            mincriterion = 1-exp(seq(-1, 0, length = 50))),
                          metric = "ROC",
                          trControl = ctrl_binary)
binary_ctree <- ggplot(ctree.fit.binary, highlight = TRUE)
binary_ctree
```



```
plot(ctree.fit.binary$finalModel)
```



Random forests

```
# Using caret
rf.grid.binary <- expand.grid(mtry = 1:14,
                              splitrule = "gini",
                              min.node.size = seq(from = -2, to = 8, by = 2))
set.seed(6360)
rf.fit.binary <- train(recovery_time ~ . ,
                       data_binary,
                       subset = trRows_binary,
                       method = "ranger",
                       tuneGrid = rf.grid.binary,
                       metric = "ROC",
                       trControl = ctrl_binary)
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo,
## : There were missing values in resampled performance measures.
```

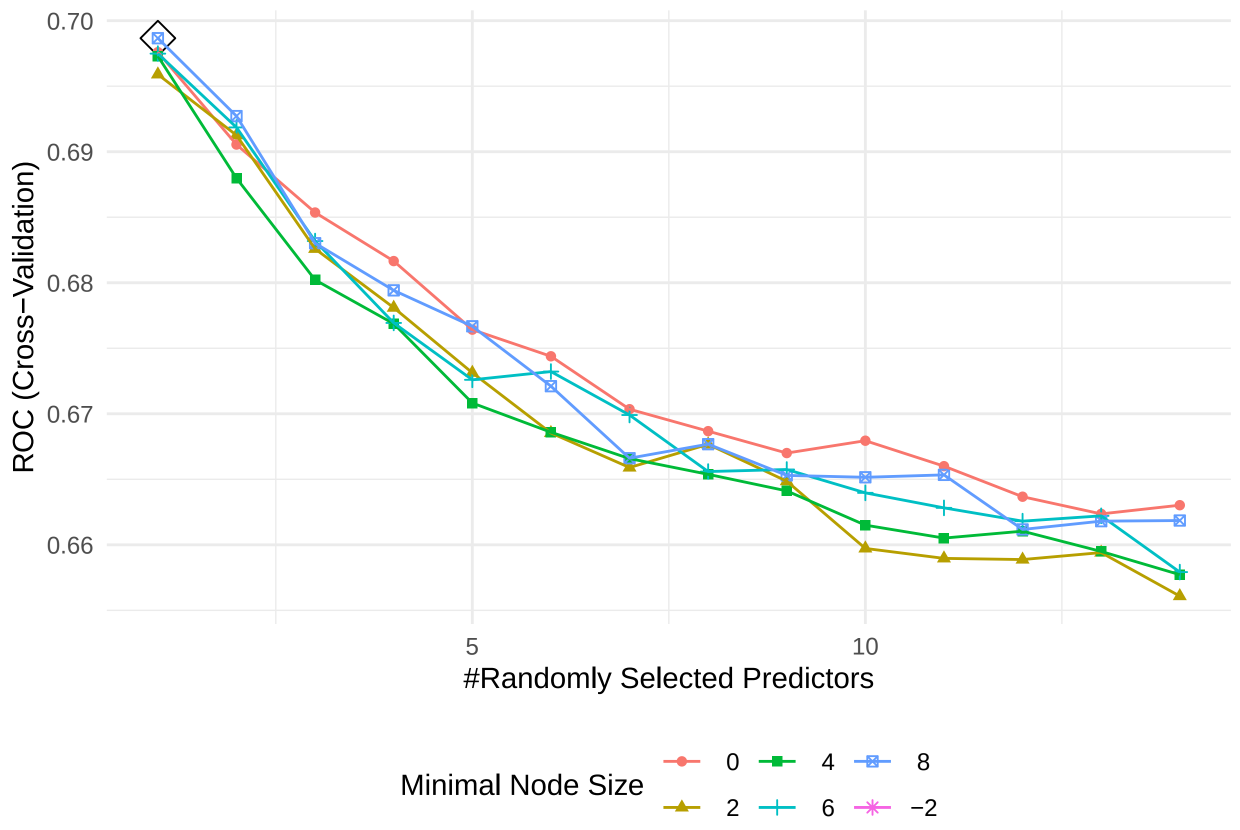
```
## Warning in train.default(x, y, weights = w, ...): missing values found in
## aggregated results
```

```
rf.fit.binary$bestTune
```

```
##   mtry splitrule min.node.size
## 6    1      gini             8
```

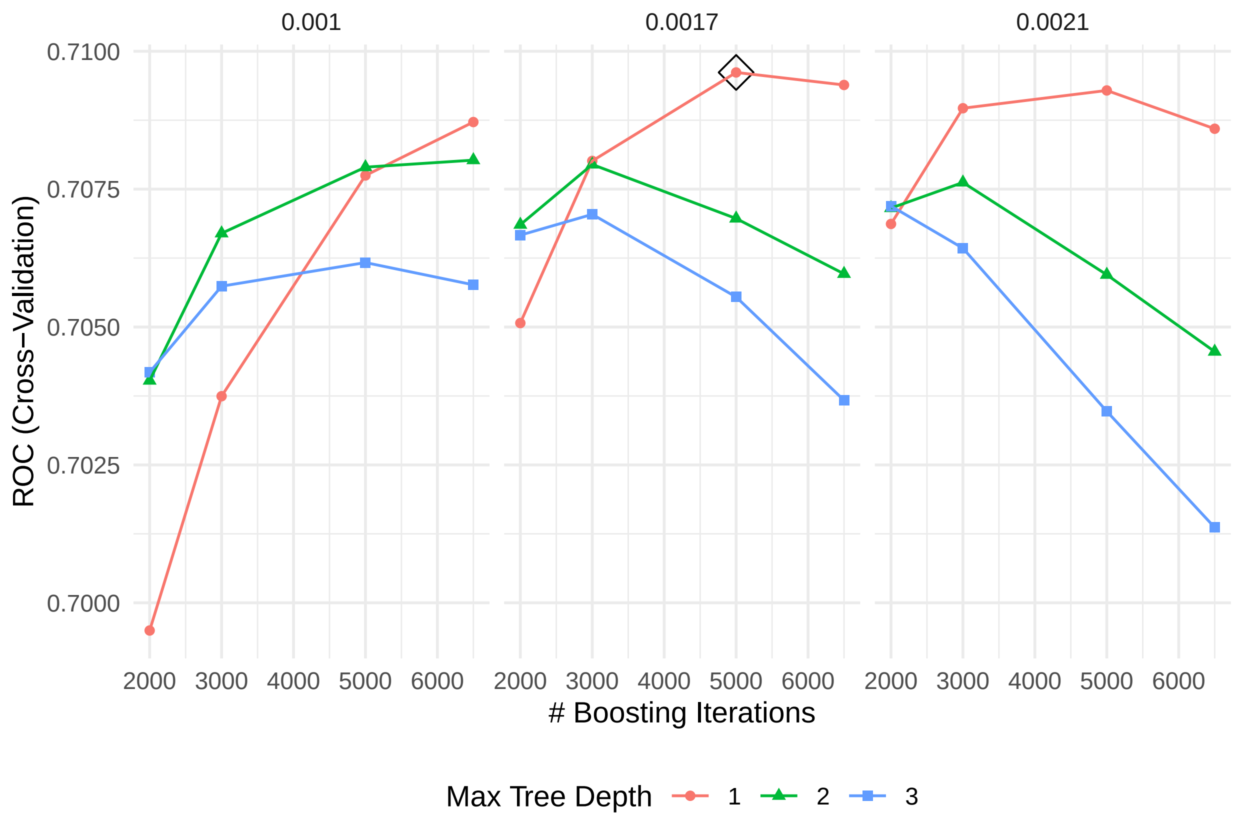
```
binary_rf <- ggplot(rf.fit.binary, highlight = TRUE)
binary_rf
```

```
## Warning: Removed 14 rows containing missing values (`geom_point()`).

## Warning: Removed 14 rows containing missing values (`geom_line()`).
```



Boosting

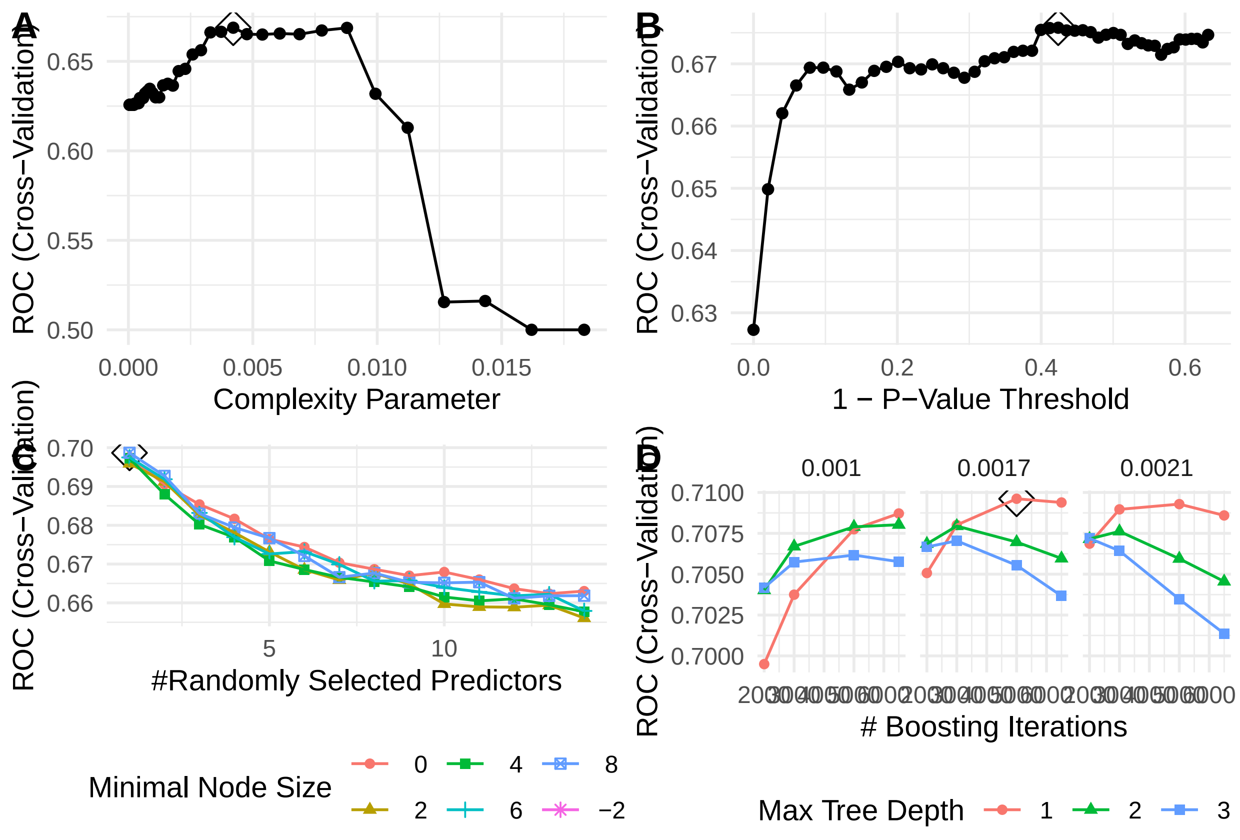
```
gbmA.grid.binary <- expand.grid(n.trees = c(2000,3000,6500,5000),
                                interaction.depth = 1:3,
                                shrinkage = c(0.001,0.0017,0.0021),
                                n.minobsinnode = 1)
set.seed(6360)
gbmA.fit.binary <- train(recovery_time ~ . ,
                         data_binary,
                         subset = trRows_binary,
                         tuneGrid = gbmA.grid.binary,
                         trControl = ctrl_binary,
                         method = "gbm",
                         distribution = "adaboost",
                         metric = "ROC",
                         verbose = FALSE)

binary_boosting <- ggplot(gbmA.fit.binary, highlight = TRUE)
binary_boosting
```



Figures for cross-validation for rpart, ctree, rf, and boosting

```
ggarrange(binary_rpart, binary_ctree, binary_rf, binary_boosting,
          labels = c("A", "B", "C", "D"),
          ncol = 2, nrow = 2)
```

```
## Warning: Removed 14 rows containing missing values (‘geom_point()‘).

## Warning: Removed 14 rows containing missing values (‘geom_line()‘).
```



Comparing all binary models

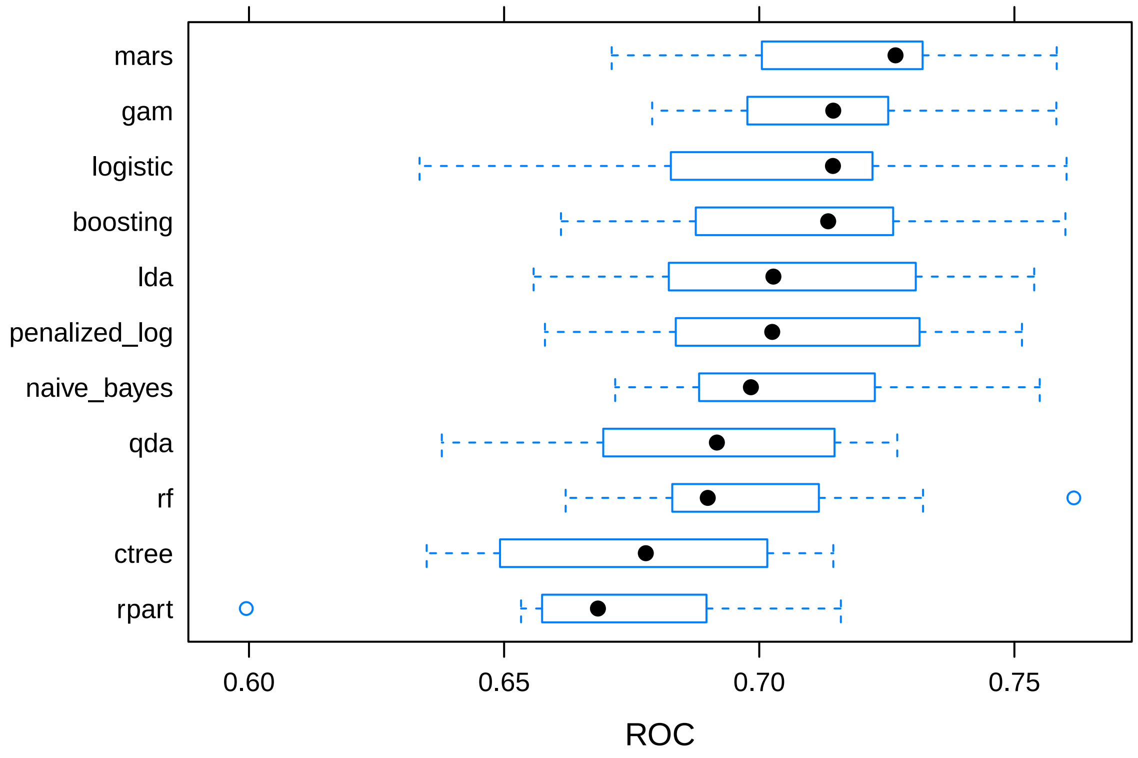
We will now compare all the models created to each other to find the best predictive model for COVID-19 recovery time which will be represented as a binarized outcome.

```
resample <- resamples(list(logistic = model.glm.binary,
                           penalized_log = model.glmn.penalized,
                           lda = model.lda.binary,
                           qda = model.qda.binary,
                           naive_bayes = model.nb.binary,
                           gam = model.gam.binary,
                           mars = model.mars.binary,
                           rpart = rpart.fit.binary,
                           ctree = ctree.fit.binary,
                           rf = rf.fit.binary,
                           boosting = gbmA.fit.binary))
summary(resample)
```

```
##
## Call:
## summary.resamples(object = resample)
##
## Models: logistic, penalized_log, lda, qda, naive_bayes, gam, mars, rpart, ctree, rf, boosting
## Number of resamples: 10
```

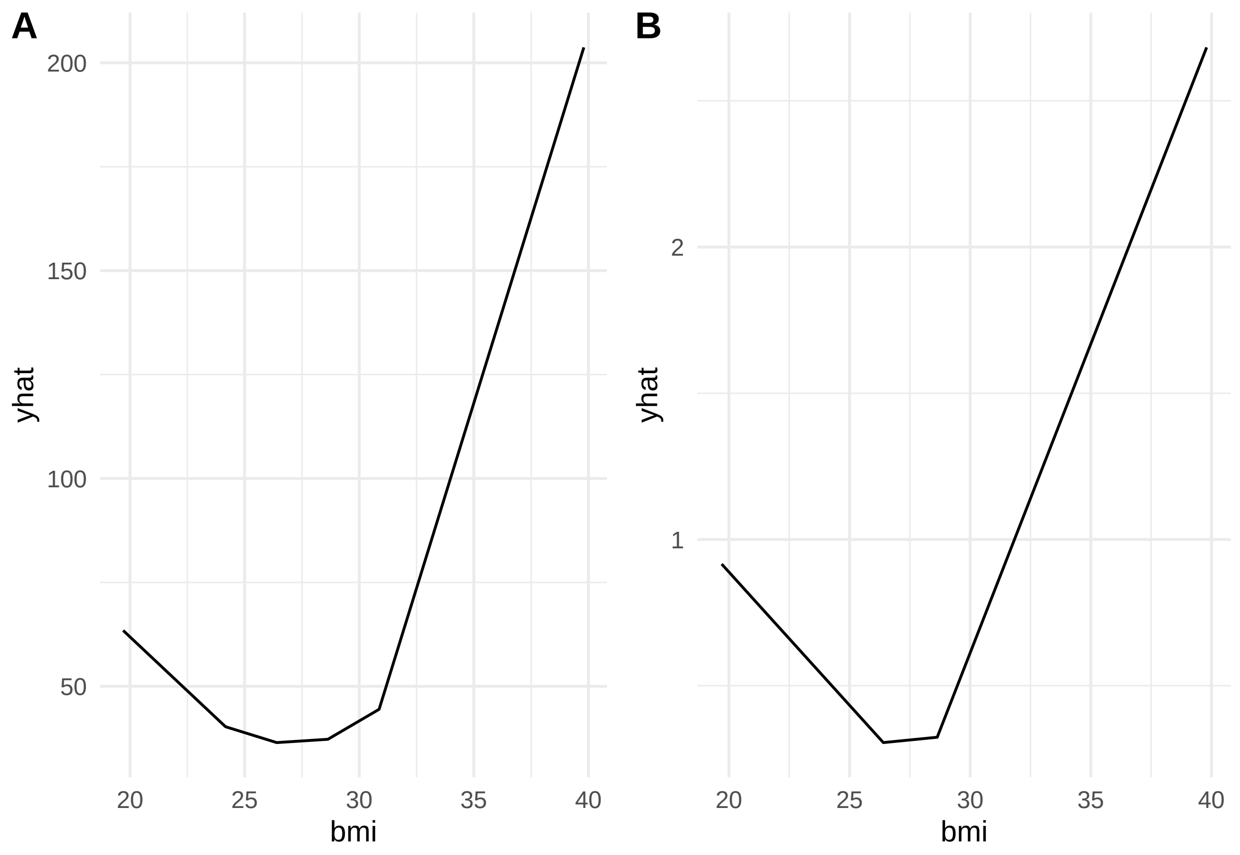
```
##
## ROC
##                   Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## logistic      0.6334270 0.6881678 0.7144460 0.7061498 0.7215150 0.7602216    0
## penalized_log 0.6580059 0.6874492 0.7025468 0.7065405 0.7292129 0.7514847    0
## lda           0.6557789 0.6853678 0.7027752 0.7055242 0.7292683 0.7538831    0
## qda           0.6377912 0.6726178 0.6916958 0.6904075 0.7145856 0.7270455    0
## naive_bayes   0.6717679 0.6886297 0.6983646 0.7046511 0.7201062 0.7549680    0
## gam           0.6790201 0.6999509 0.7145024 0.7164725 0.7252027 0.7582229    0
## mars          0.6710827 0.7026620 0.7267055 0.7201370 0.7319041 0.7582955    0
## rpart         0.5994746 0.6597818 0.6683892 0.6688046 0.6854076 0.7159947    0
## ctree         0.6348218 0.6516219 0.6777691 0.6758037 0.7008908 0.7145101    0
## rf            0.6620603 0.6841571 0.6899077 0.6986612 0.7070866 0.7616492    0
## boosting      0.6611466 0.6880351 0.7135098 0.7096156 0.7247554 0.7599931    0
##
## Sens
##                   Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## logistic      0.8844221 0.9057789 0.9097739 0.9142010 0.9309045 0.9396985    0
## penalized_log 0.8894472 0.9007538 0.9072613 0.9156784 0.9311495 0.9550000    0
## lda           0.9045226 0.9145729 0.9223116 0.9272236 0.9411621 0.9600000    0
## qda           0.6532663 0.6858606 0.6967462 0.6979271 0.7035176 0.7450000    0
## naive_bayes   0.9800000 0.9949749 0.9949749 0.9944849 0.9987500 1.0000000    0
## gam           0.8643216 0.8907035 0.9022739 0.9026432 0.9184485 0.9300000    0
## mars          0.8643216 0.8923744 0.9045226 0.9021533 0.9137500 0.9246231    0
## rpart         0.8341709 0.8644912 0.8793970 0.8765653 0.8875000 0.9195980    0
## ctree         0.8542714 0.8657412 0.8793970 0.8890754 0.9195980 0.9400000    0
## rf            1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000    0
## boosting      0.9547739 0.9610992 0.9723618 0.9729095 0.9849246 0.9900000    0
##
## Spec
##                     Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
## logistic      0.17045455 0.23946629 0.27119510 0.26272983 0.29012385 0.31818182
## penalized_log 0.17045455 0.20454545 0.26136364 0.24795710 0.29012385 0.30337079
## lda           0.14772727 0.19602273 0.25989530 0.23891726 0.27272727 0.30337079
## qda           0.44318182 0.59375000 0.61018897 0.59442033 0.62215909 0.66292135
## naive_bayes   0.01123596 0.01136364 0.01136364 0.01699438 0.02266343 0.03409091
## gam           0.15909091 0.22443182 0.29934883 0.28538049 0.34646323 0.38636364
## mars          0.18181818 0.25284091 0.29366701 0.29664198 0.34659091 0.40449438
## rpart         0.18181818 0.23103933 0.26136364 0.26160623 0.28651685 0.35227273
## ctree         0.08988764 0.14488636 0.24840398 0.22197395 0.29462462 0.30681818
## rf            0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
## boosting      0.05681818 0.07887513 0.09595250 0.09624617 0.11331716 0.13636364
##               NA's
## logistic         0
## penalized_log    0
## lda              0
## qda              0
## naive_bayes      0
## gam              0
## mars             0
## rpart            0
## ctree            0
## rf               0
## boosting         0
```

```r
bwplot(resample, metric = "ROC")
```



partial dependence plots

```r
ggarrange(continuous_pdp, binary_pdp,
          labels = c("A", "B"),
          ncol = 2)
```


A
B
yhat
bmi
200
150
100
50
2
1
20
25
30
35
40