analysis_I

Yijin Wang

2023-05-06

```
library(janitor)
library(tidyverse)
library(AppliedPredictiveModeling)
library(lattice)
library(caret)
library(corrplot)
library(GGally)
library(miscset)
library(ggpubr)
library(knitr)
library(rpart)
library(rpart.plot)
library(ranger)
```

1. Load Data

```
load("final_data.RData")
```

2. Train/test split

```
set.seed(2)
training_rows <- createDataPartition(final_data$recovery_time,
                                     p = 0.8,
                                     list = F)
```

```
train_x <- model.matrix(recovery_time~., final_data %>% select(-id, -binary_recovery_time))[training_rou
train_y <- final_data$recovery_time[training_rows]
test_x <- model.matrix(recovery_time~., final_data %>% select(-id, -binary_recovery_time))[-training_rou
test_y <- final_data$recovery_time[-training_rows]
```

```
training_set <- final_data[training_rows,]
```

3.EDA

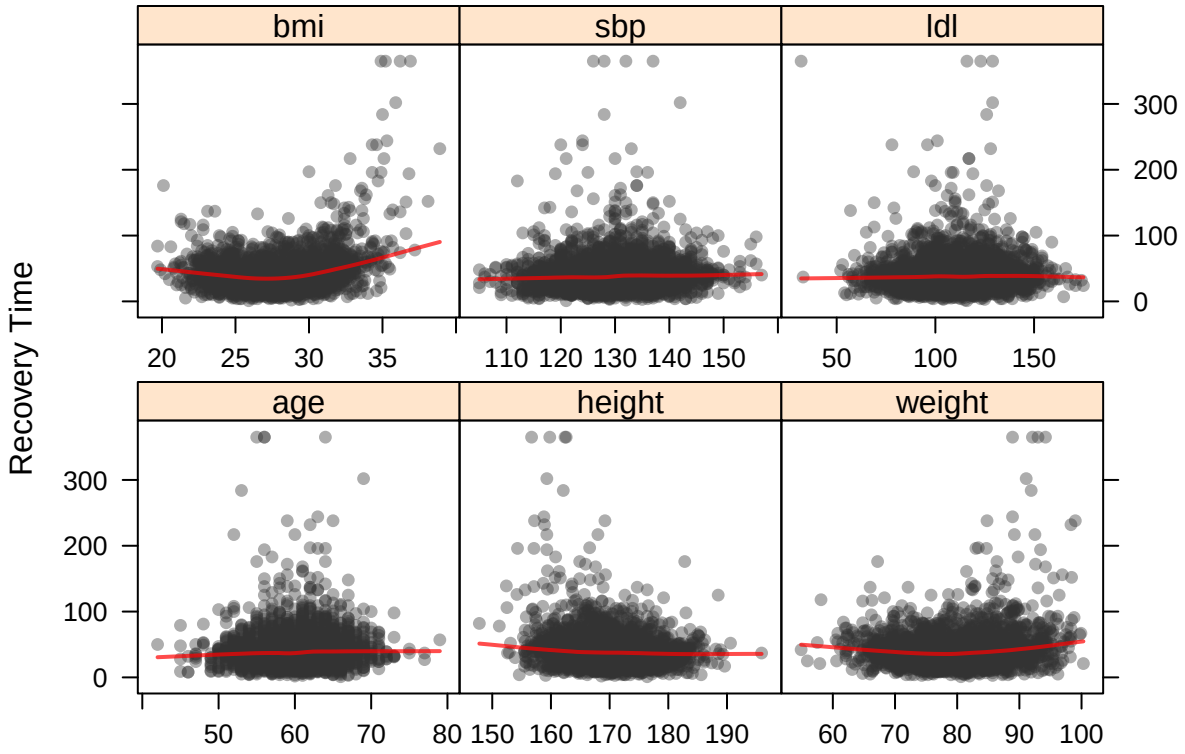
basic summary statistics

```
summary(training_set)
```

```
##        id              age        gender   race     smoking      height
##  Min.   :    5   Min.   :42.00   0:1474   1:1842   0:1791   Min.   :147.8
##  1st Qu.: 2454   1st Qu.:57.00   1:1416   2: 146   1: 832   1st Qu.:166.1
##  Median : 4804   Median :60.00            3: 614   2: 267   Median :170.1
##  Mean   : 4904   Mean   :60.14            4: 288            Mean   :170.0
##  3rd Qu.: 7339   3rd Qu.:63.00                              3rd Qu.:173.8
##  Max.   :10000   Max.   :79.00                              Max.   :195.9
##      weight           bmi        hypertension diabetes      sbp
##  Min.   : 54.90   Min.   :19.70   0:1518       0:2434   Min.   :105
##  1st Qu.: 75.40   1st Qu.:25.90   1:1372       1: 456   1st Qu.:125
##  Median : 80.10   Median :27.70                         Median :130
##  Mean   : 80.02   Mean   :27.76                         Mean   :130
##  3rd Qu.: 84.90   3rd Qu.:29.50                         3rd Qu.:135
##  Max.   :100.30   Max.   :38.90                         Max.   :157
##       ldl      vaccine  severity study    recovery_time    binary_recovery_time
##  Min.   : 32   0:1174   0:2615   A: 586   Min.   :  1.00   Min.   :0.000
##  1st Qu.: 97   1:1716   1: 275   B:1754   1st Qu.: 28.00   1st Qu.:0.000
##  Median :110                     C: 550   Median : 39.00   Median :1.000
##  Mean   :110                              Mean   : 42.58   Mean   :0.691
##  3rd Qu.:123                              3rd Qu.: 50.00   3rd Qu.:1.000
##  Max.   :175                              Max.   :365.00   Max.   :1.000
```

quantative variables

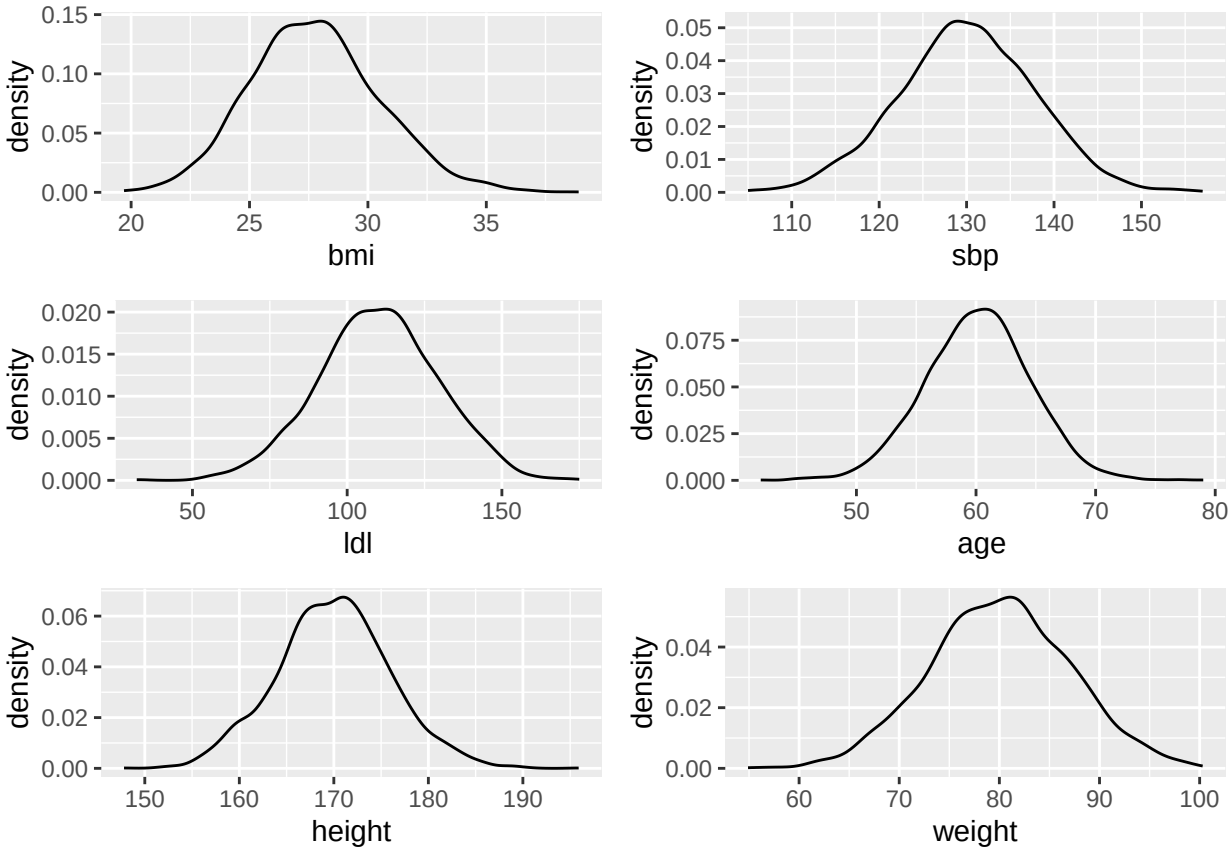
```
theme <- transparentTheme(trans = 0.4)
theme$plot.symbol$col = rgb(.2, .2, .2, .4)
theme$plot.symbol$pch = 16
theme$plot.line$col = rgb(1, 0, 0, .7)
theme$plot.line$lwd <- 2
trellis.par.set(theme)
featurePlot(x = training_set %>% dplyr::select(-recovery_time, -id, -gender, -race,-smoking,-hypertensi
            y = training_set$recovery_time,
            plot = "scatter",
            type = c("p","smooth"),
            span = .5,
            auto.key = list(columns = 2),
            labels = c("", "Recovery Time"))
```




```
ggplotGrid(ncol = 2,
  lapply(c("bmi", "sbp", "ldl","age", "height", "weight"),
    function(col) {
        ggplot(training_set, aes_string(col)) + geom_density(aes(y = ..density..))
    }))
```

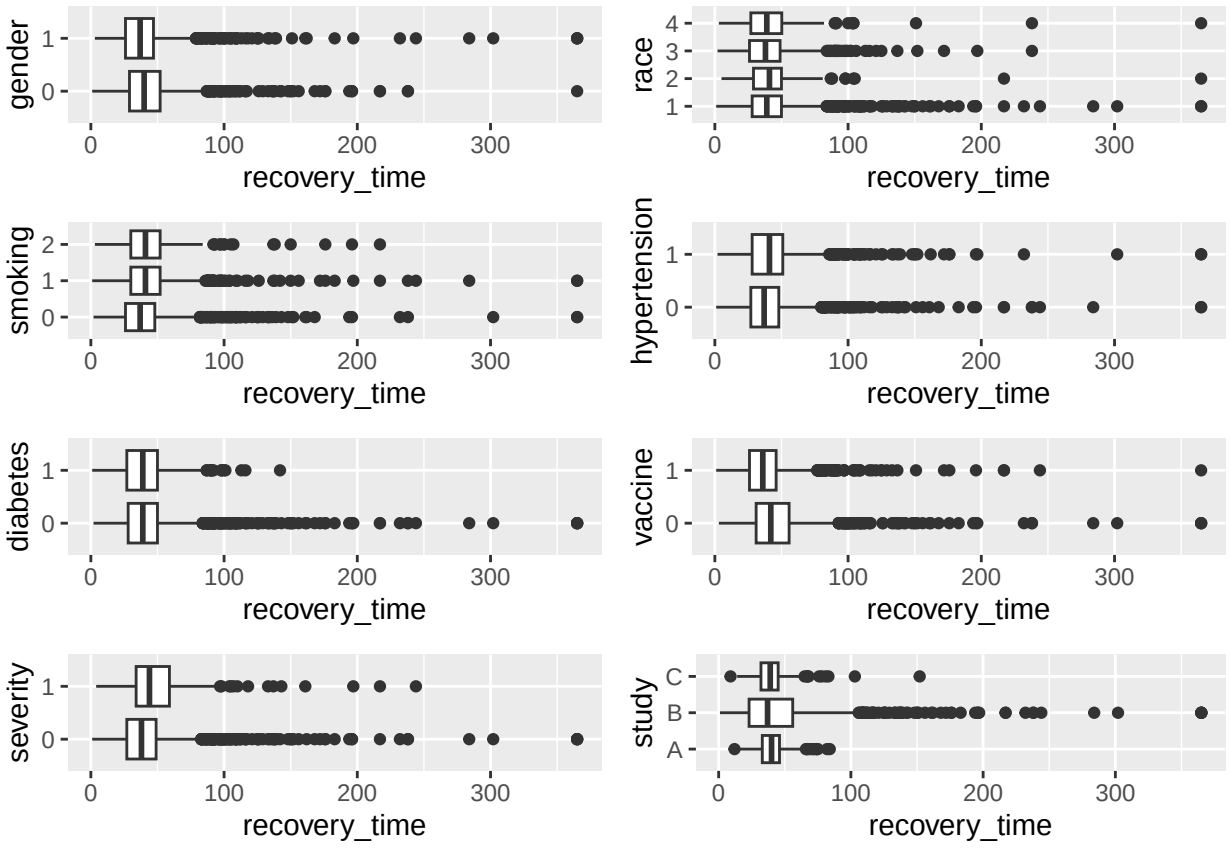
```
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## i Please use tidy evaluation idioms with `aes()`.
## i See also `vignette("ggplot2-in-packages")` for more information.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## Warning: The dot-dot notation (`..density..`) was deprecated in ggplot2 3.4.0.
## i Please use `after_stat(density)` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



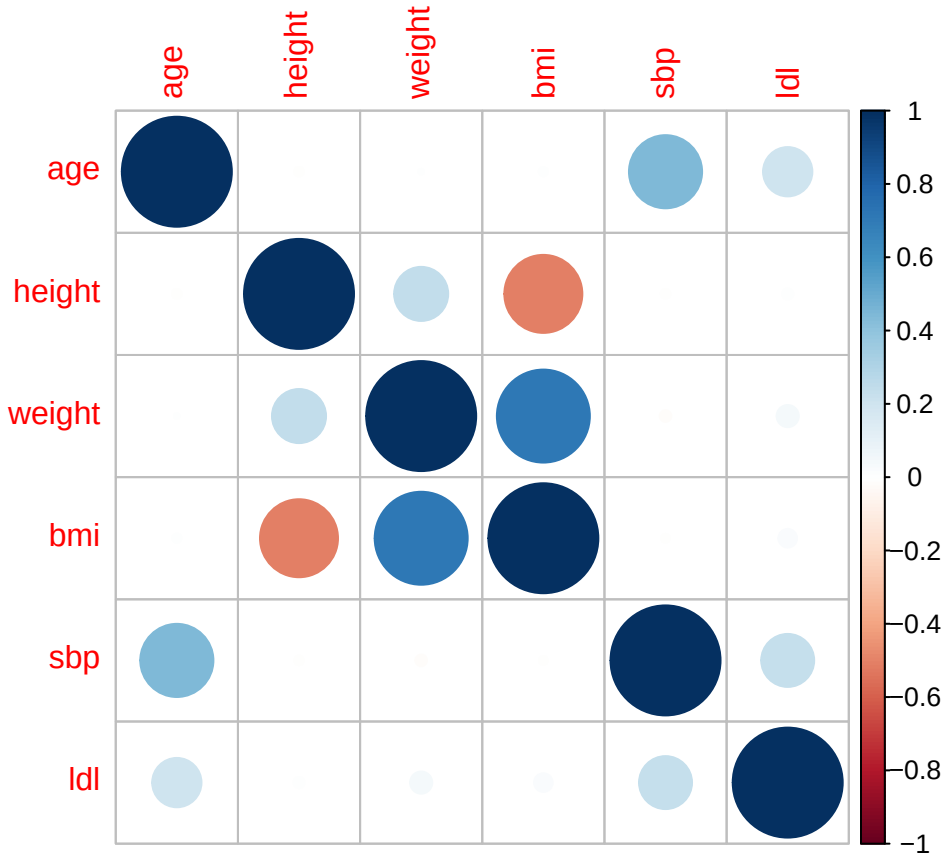
categorical variables

```
ggplotGrid(ncol = 2,
  lapply(c("gender", "race", "smoking","hypertension", "diabetes", "vaccine", "severity", "study"),
    function(col) {
        ggplot(training_set, aes_string(col)) + geom_boxplot(aes(y = recovery_time)) + coord_flip()
    }))
```



Correlation plot for quantative variables

```
corrplot(cor(training_set %>% dplyr::select(-recovery_time, -id, -gender, -race,-smoking,-hypertension,
```



4. set up contrl

Create control for 10-fold cross validation

```
set.seed(2)
ctrl1 <- trainControl(method = "cv", number = 10)
```

5. Model training

a. Linear

```
set.seed(2)
linear_fit <- train(train_x, train_y,method = "lm")
```

```
coef(linear_fit$finalModel)%>%
  as.matrix() %>%
  as.data.frame() %>%
  rename(value = V1) %>%
  kable(caption = "Linear Regression Parameter Coefficients")
```

Table 1: Linear Regression Parameter Coefficients

	value
(Intercept)	-2300.4502049
age	0.1521655
gender1	-3.8307356
race2	2.6808765
race3	-3.0255432
race4	-0.7740480
smoking1	4.6864852
smoking2	6.4648973
height	13.3641497
weight	-14.4018371
bmi	43.6870557
hypertension1	2.9232874
diabetes1	-1.9286525
sbp	0.0416723
ldl	-0.0140814
vaccine1	-9.7604602
severity1	9.4368078
studyB	4.1323356
studyC	-0.1725793

Test set performance

```
linear_fit_prediction <- predict(linear_fit, newdata = test_x)
linear_ts_mse <- mean((linear_fit_prediction - test_y)^2)
linear_ts_mse
```

```
## [1] 762.7512
```

b. Elastic Net

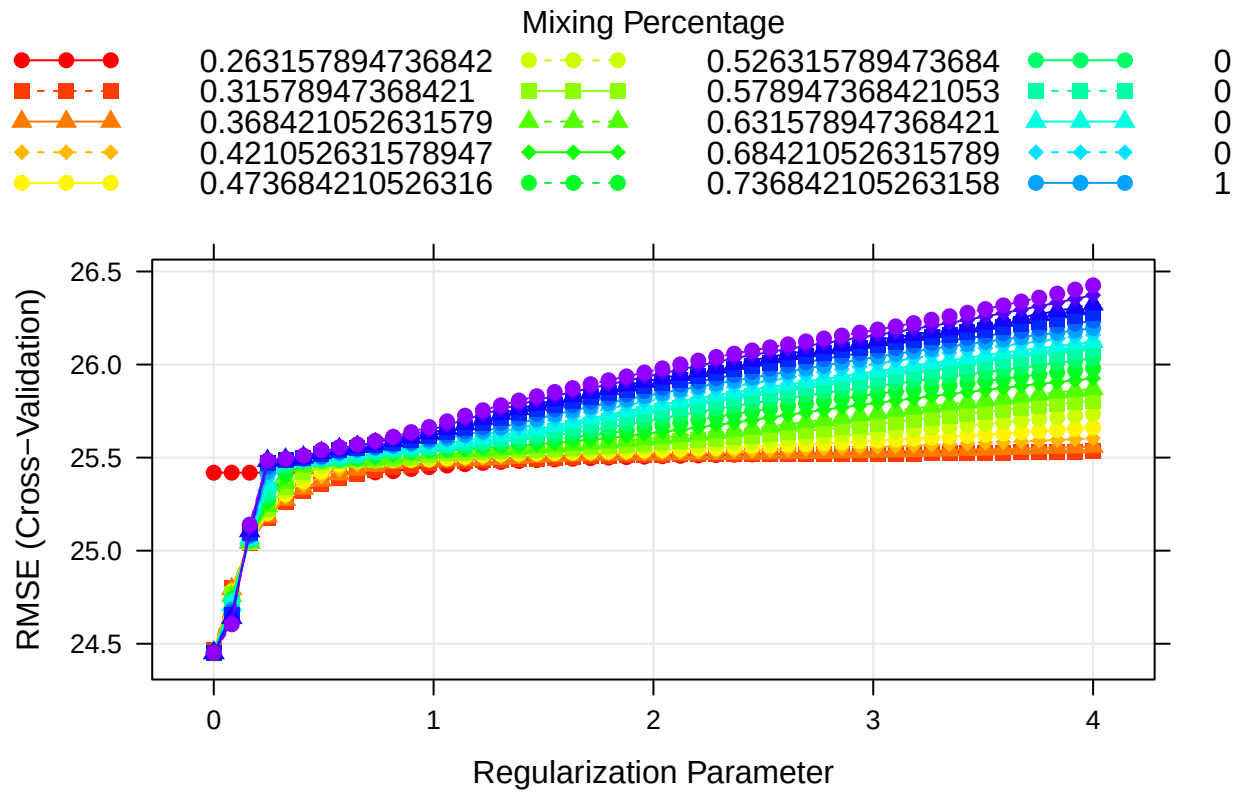
```
set.seed(2)
enet_fit <- train(train_x, train_y,
                  method = "glmnet",
                  tuneGrid = expand.grid(alpha = seq(0, 1, length = 20),
                                         lambda = seq(0,4,length = 50)),

                  trControl = ctrl1)
```

```
enet_fit$bestTune
```

```
##         alpha lambda
## 151 0.1578947      0
```

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


Mixing Percentage
0.263157894736842
0.31578947368421
0.368421052631579
0.421052631578947
0.473684210526316
0.526315789473684
0.578947368421053
0.631578947368421
0.684210526315789
0.736842105263158
0
0
0
0
1
RMSE (Cross−Validation)
26.5
26.0
25.5
25.0
24.5
0
1
2
3
4
Regularization Parameter

```
coef(enet_fit$finalModel, enet_fit$bestTune$lambda) %>%
  as.matrix() %>%
  as.data.frame() %>%
  rename(value = s1) %>%
  kable(caption = "Elastic-Net Parameter Coefficients")
```

Table 2: Elastic-Net Parameter Coefficients

	value
(Intercept)	-2100.9504791
age	0.1513531
gender1	-3.8019675
race2	2.6292444
race3	-3.0475788
race4	-0.7578348
smoking1	4.6452444
smoking2	6.4047249
height	12.1860090
weight	-13.1523205
bmi	40.1166777
hypertension1	2.9004753
diabetes1	-1.9252898
sbp	0.0411921
ldl	-0.0140378
vaccine1	-9.7839039
severity1	9.3904789
studyB	4.1647248
studyC	-0.1543670

Test set performance

```
enet_fit_prediction <- predict(enet_fit, newdata = test_x)
enet_ts_mse <- mean((enet_fit_prediction - test_y)^2)
enet_ts_mse
```

```
## [1] 770.718
```

c. GAM

```
set.seed(2)
gam_fit <- train(train_x,
                 train_y,
                 method = "gam",
                 trControl = ctrl1)
```

```
gam_fit$bestTune
```

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

```
gam_fit$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:
## 0.000 0.000 8.794 8.310 0.651 4.266  total = 35.02
##
## GCV score: 507.3009
```

age and sbp are not in the model

```
s <-summary(gam_fit$finalMode)
s$p.coeff%>%
  as.matrix() %>%
  as.data.frame()%>%
  rename(value = V1) %>%
  kable(caption = "GAM Parameter Coefficients")
```

Table 3: GAM Parameter Coefficients

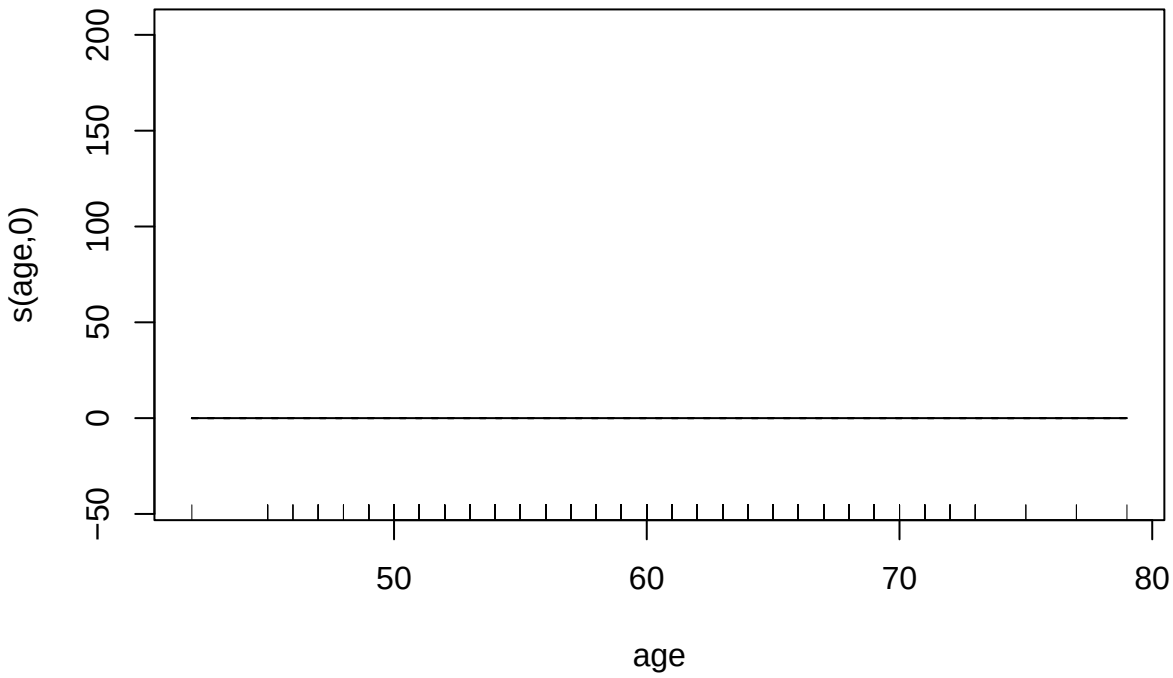
	value
(Intercept)	43.7326430
gender1	-3.3768343
race2	1.1757834
race3	-2.0871968
race4	-1.9148878
smoking1	5.4396984
smoking2	6.8982529
hypertension1	3.5038158
diabetes1	-1.0100817
vaccine1	-9.6389980
severity1	9.2355146
studyB	3.8150580
studyC	-0.5173404

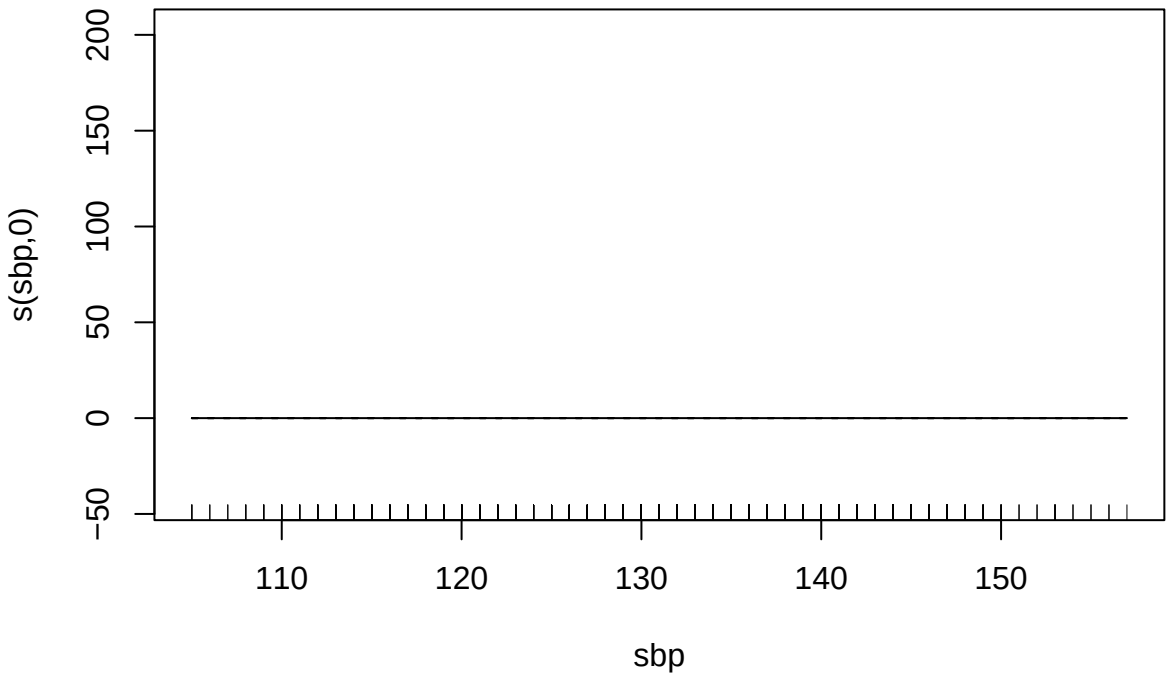
```
s$chi.sq%>%
  as.matrix() %>%
  as.data.frame() %>%
  rename(value = V1) %>%
  kable(caption = "GAM EDF")
```

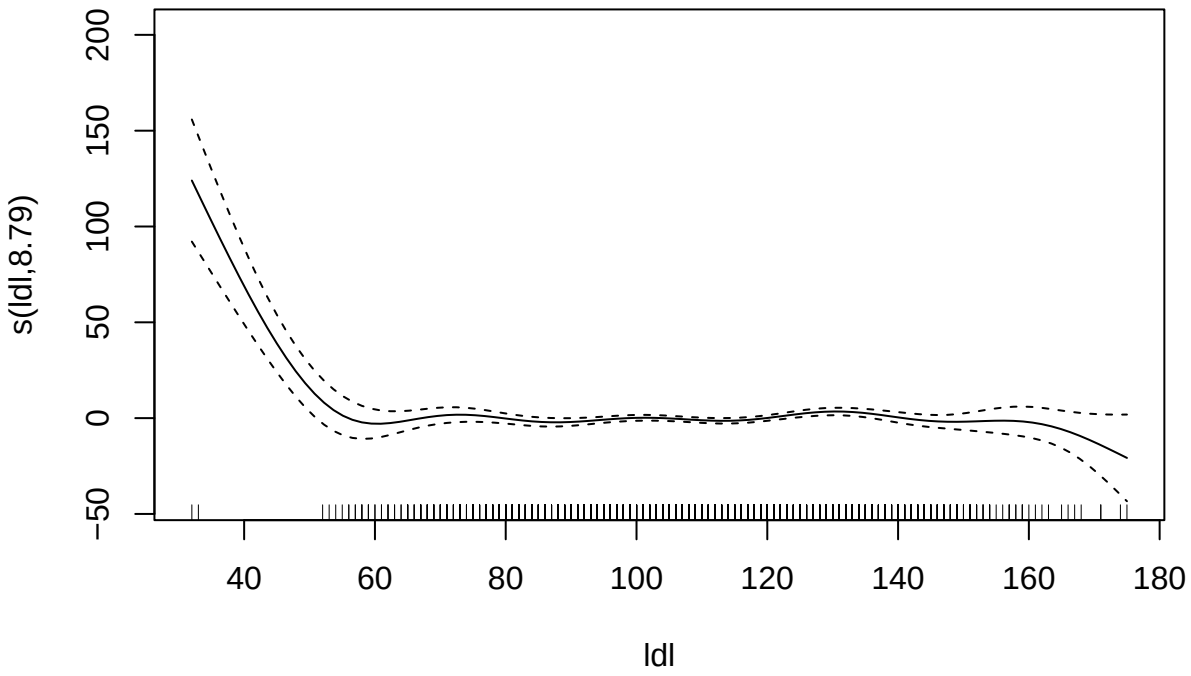
Table 4: GAM EDF

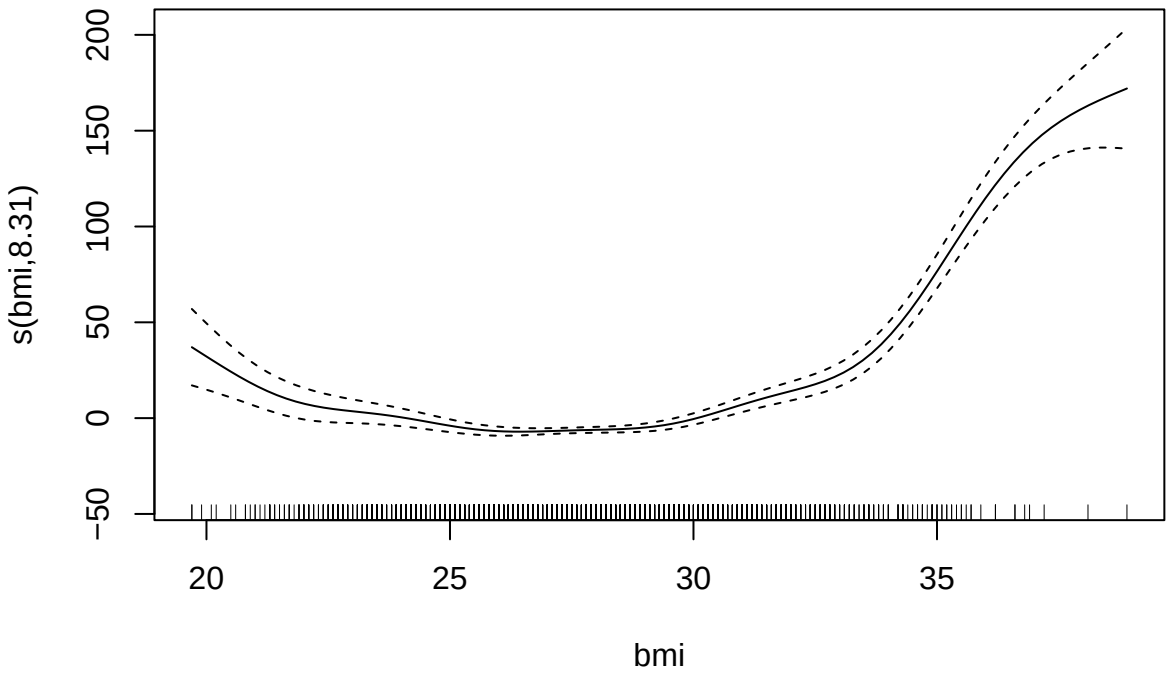
	value
s(age)	0.0000011
s(sbp)	0.0000004
s(ldl)	76.1404212
s(bmi)	776.2559450
s(height)	1.0131011
s(weight)	5.2909981

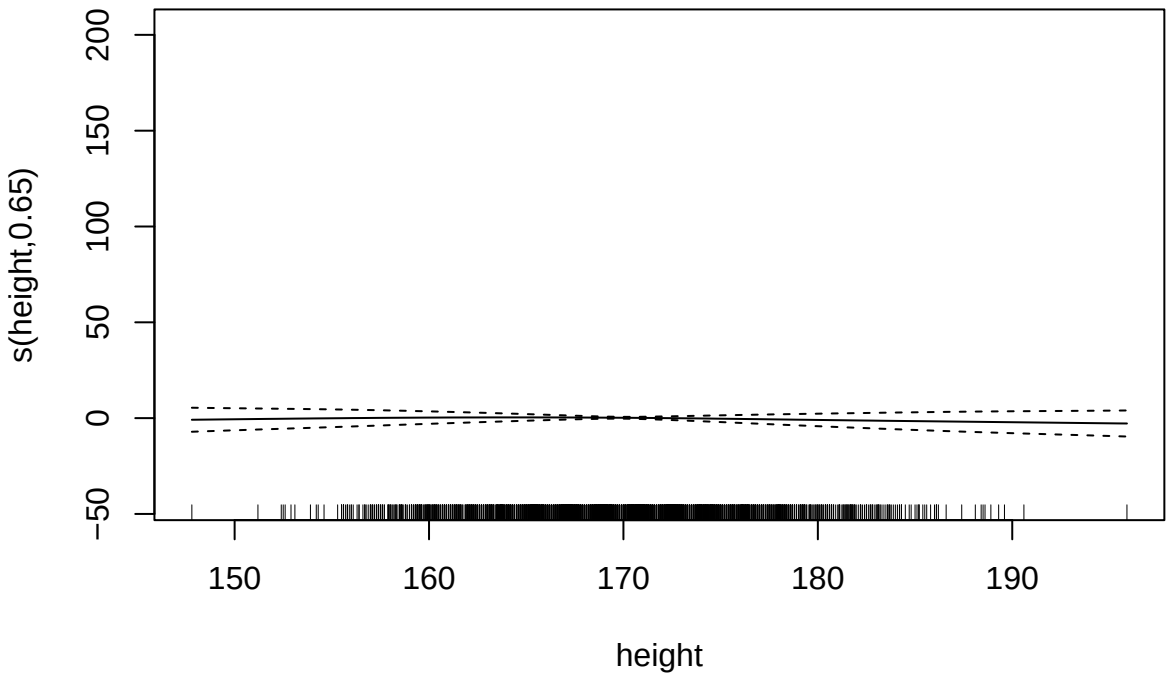
```
plot(gam_fit$finalModel)
```

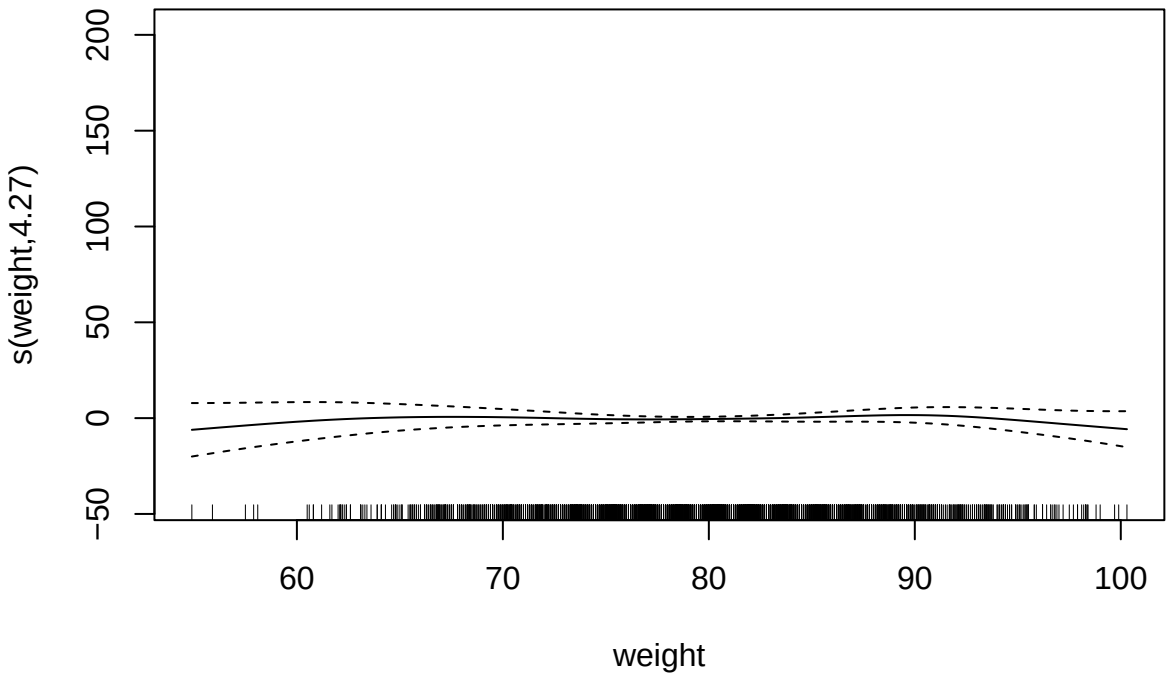

s(age,0)
−50
0
50
100
150
200
50
60
70
80
age


s(sbp,0)
200
150
100
50
0
−50
110
120
130
140
150
sbp


s(ldl,8.79)
200
150
100
50
0
−50
40
60
80
100
120
140
160
180
ldl


s(bmi,8.31)
200
150
100
50
0
−50
20
25
30
35
bmi


s(height,0.65)
200
150
100
50
0
−50
150
160
170
180
190
height



Test set performance

```
gam_fit_prediction <- predict(gam_fit, newdata = test_x)
gam_ts_mse <- mean((gam_fit_prediction - test_y)^2)
gam_ts_mse
```

```
## [1] 655.4724
```

d. MARS

```
mars_grid <- expand.grid(
# could be product of 4 hinge functions
degree = 1:4,
# upper bound of terms in the model. We have 16 predictors. upper bound could go higher
nprune = 2:18)
```

```
set.seed(2)
mars_fit <- train(train_x,
                  train_y,
                  method = "earth",
                  tuneGrid = mars_grid,
                  trControl = ctrl1)
```

```
## Loading required package: earth
```

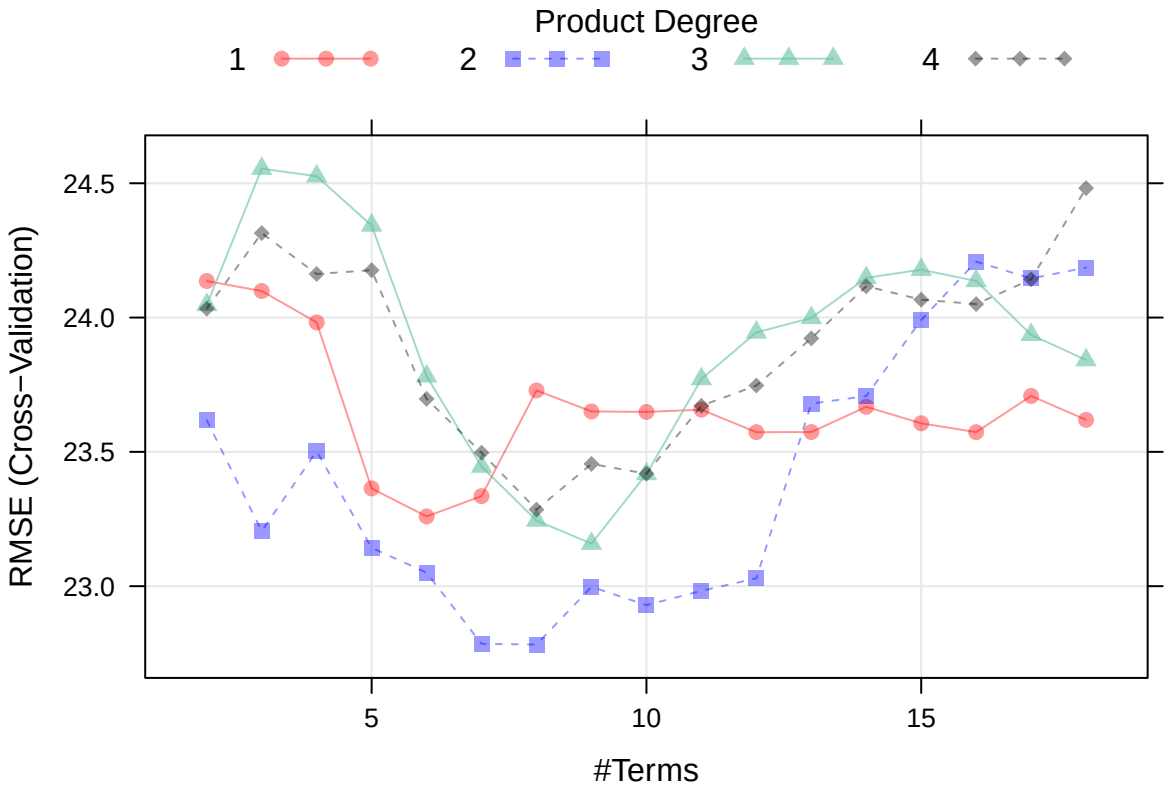
```
## Loading required package: Formula

## Loading required package: plotmo

## Loading required package: plotrix

## Loading required package: TeachingDemos
```

```
plot(mars_fit)
```




```
mars_fit$bestTune
```

```
##    nprune degree
## 24      8      2
```

```
coef(mars_fit$finalModel) %>%
  as.matrix() %>%
  as.data.frame() %>%
  rename(value = V1) %>%
  kable(caption = "MARS Parameter Coefficients")
```

Table 5: MARS Parameter Coefficients

	value
(Intercept)	25.139126
h(30.6-bmi)	3.293443
h(bmi-30.6) * studyB	13.250267
vaccine1	-8.691923
h(bmi-26.4)	4.809880
smoking1 * h(bmi-33.6)	51.325423
h(bmi-33.6) * h(90-ldl)	2.845967
severity1 * studyB	13.735474

Test set performance

```
mars_fit_prediction <- predict(mars_fit, newdata = test_x)
mars_ts_mse <- mean((mars_fit_prediction - test_y)^2)
mars_ts_mse
```

```
## [1] 621.5237
```

e. Random Forest with boosting

```
set.seed(2)
gbm_grid <- expand.grid(
  # upper bound for number of trees
  n.trees = c(200, 500, 1000, 2000, 3000),
  # similar to number of splits in the tree
  # number of layers in the tree
  interaction.depth = 1:5,
  shrinkage = c(0.005,0.01,0.015),
  # min obs allowed in a node
  n.minobsinnode=c(1,5))

library(doParallel)
```

```
## Loading required package: foreach
```

```
##
## Attaching package: 'foreach'
```

```
## The following objects are masked from 'package:purrr':
##
##     accumulate, when
```

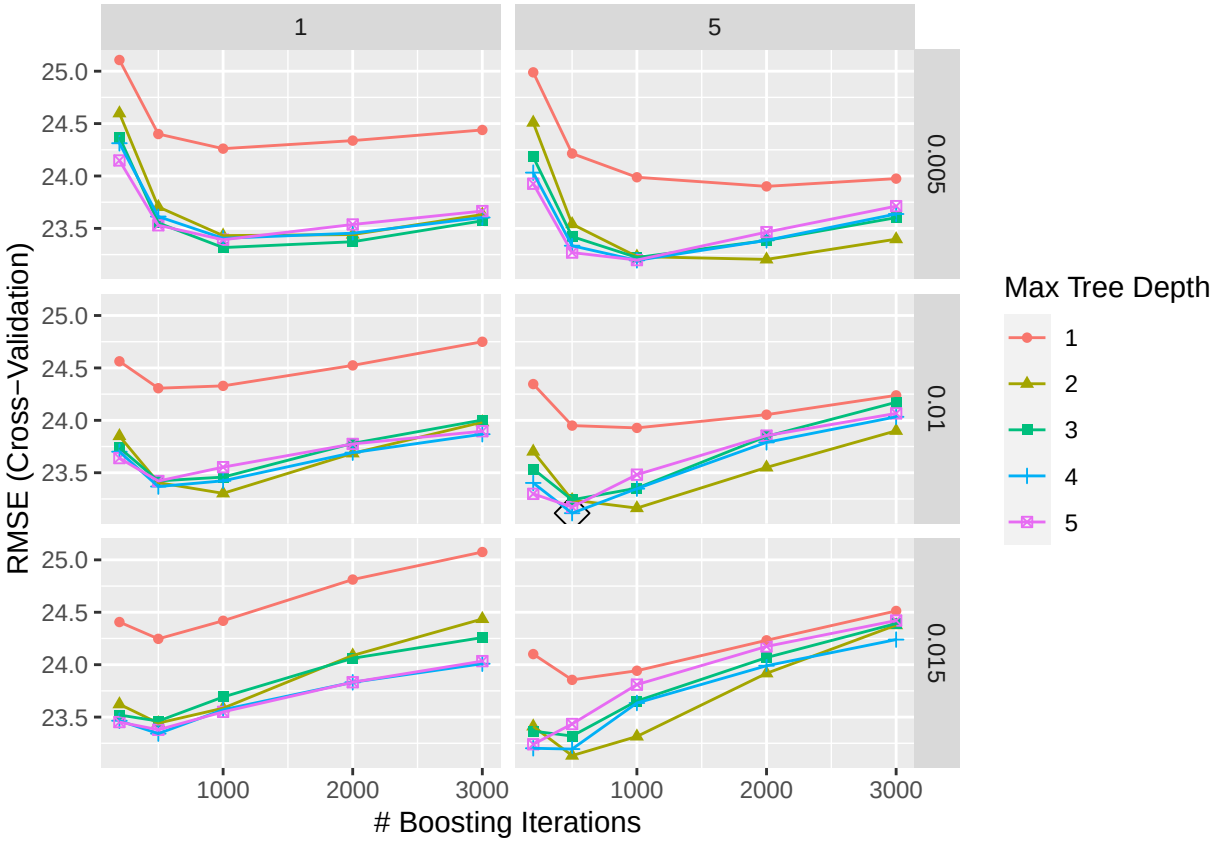
```
## Loading required package: iterators
```

```
## Loading required package: parallel
```

```
Mycluster = makeCluster(detectCores()-2)
registerDoParallel(Mycluster)

boost_fit <- train(train_x,
                   train_y,
                   method = "gbm",
                   tuneGrid = gbm_grid,
                   trControl = ctrl1,
                   verbose = FALSE)
stopCluster(Mycluster)
registerDoSEQ()
```

```
ggplot(boost_fit, highlight = TRUE)
```

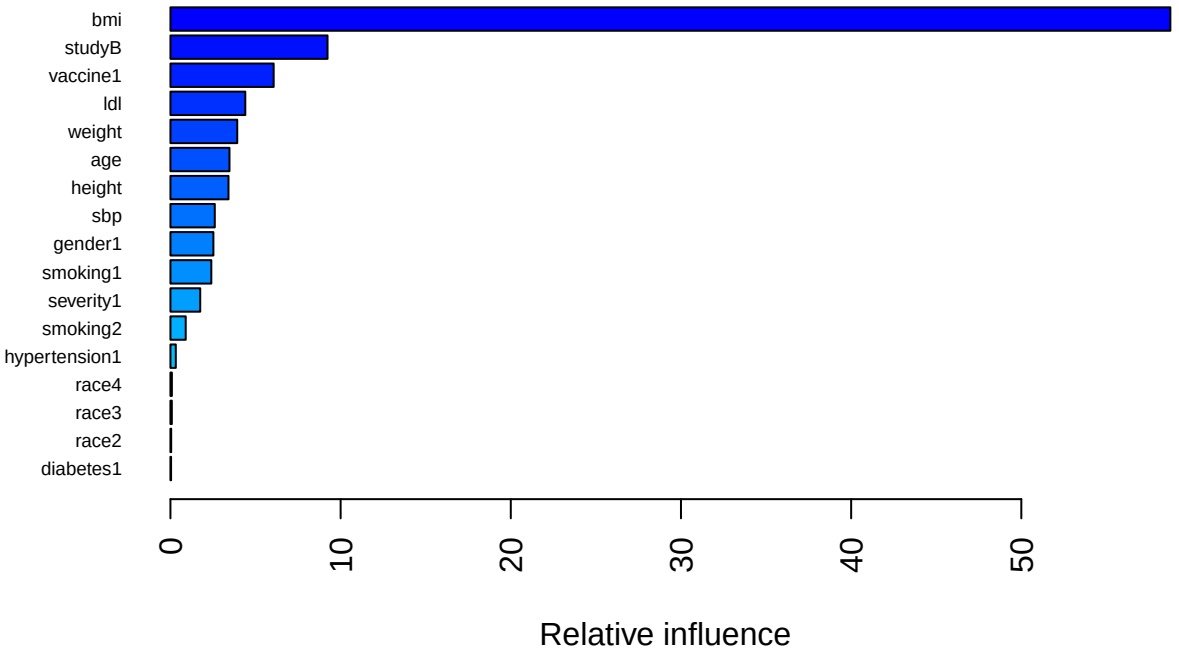



```
boost_fit$bestTune
```

```
##    n.trees interaction.depth shrinkage n.minobsinnode
## 87     500                 4      0.01              5
```

variable importance

```
summary(boost_fit$finalModel,las = 2, cBars = 17, cex.names = 0.6)
```




```
##                          var     rel.inf
## bmi                      bmi 58.75712341
## studyB                studyB  9.23051025
## vaccine1            vaccine1  6.06556957
## ldl                      ldl  4.39834204
## weight                weight  3.92516037
## age                      age  3.46871063
## height                height  3.41129619
## sbp                      sbp  2.60623391
## gender1              gender1  2.52408320
## smoking1            smoking1  2.40173881
## severity1          severity1  1.74972041
## smoking2            smoking2  0.89828855
## hypertension1 hypertension1  0.31563513
## race4                  race4  0.08419079
## race3                  race3  0.08167044
## race2                  race2  0.04356978
## diabetes1          diabetes1  0.03815652
## studyC                studyC  0.00000000
```

test error

```
boost_prediction <- predict(boost_fit$finalModel, newdata = test_x)
```

```
## Using 500 trees...
```

```
boost_ts_mse <- mean((boost_prediction - test_y)^2)
boost_ts_mse
```

```
## [1] 606.3513
```

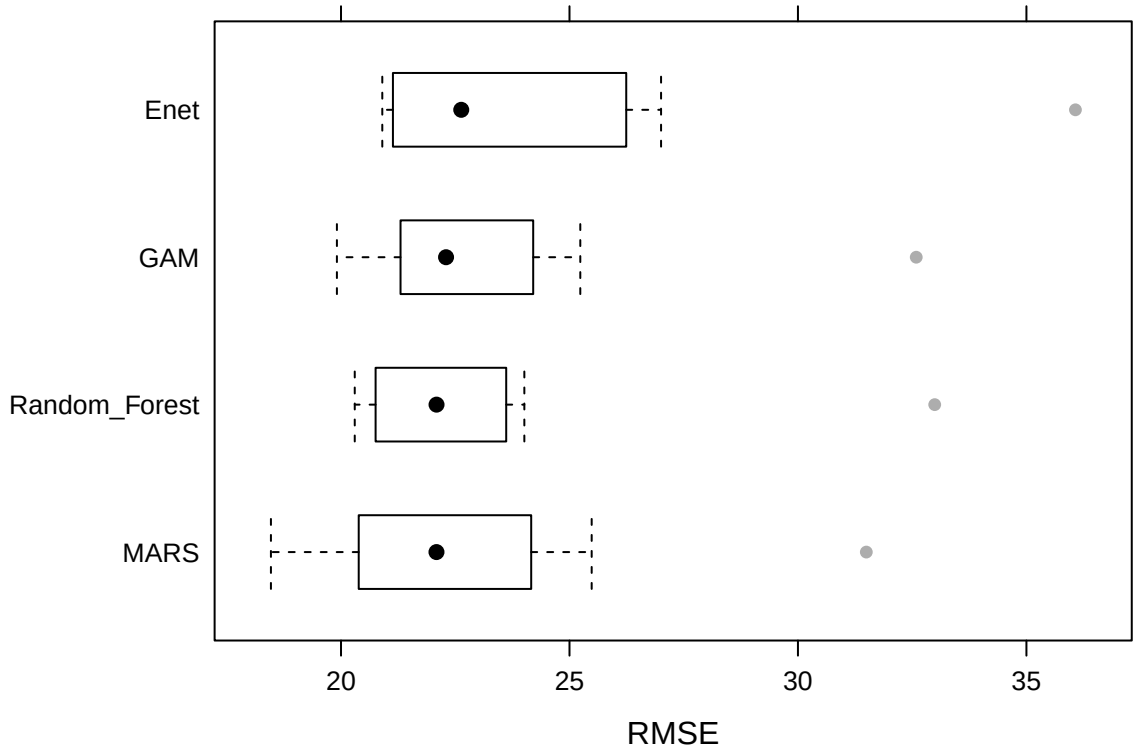
6. Comparison

Resample

```
set.seed(2)
resamp <- resamples(list(
                        Enet = enet_fit,
                        GAM = gam_fit,
                        MARS = mars_fit,
                        Random_Forest = boost_fit))
summary(resamp)
```

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: Enet, GAM, MARS, Random_Forest
## Number of resamples: 10
##
## MAE
##                   Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## Enet          14.74548 15.53000 16.17057 16.25737 16.64460 18.53102    0
## GAM           14.25550 14.92082 15.40294 15.60020 15.94239 17.67767    0
## MARS          13.41813 14.54189 15.07061 15.16312 15.71490 17.01722    0
## Random_Forest 13.76935 14.59759 14.86463 15.13552 15.50966 17.46935    0
##
## RMSE
##                   Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## Enet          20.90376 21.49485 22.63163 24.44640 25.77276 36.07284    0
## GAM           19.91016 21.33989 22.29883 23.37207 24.14763 32.58895    0
## MARS          18.46627 20.55596 22.08934 22.78254 23.83290 31.49622    0
## Random_Forest 20.30076 20.90800 22.09064 23.11414 23.58435 32.99304    0
##
## Rsquared
##                     Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## Enet          0.08490018 0.1577823 0.2099287 0.2125678 0.2846451 0.3062810    0
## GAM           0.13684006 0.1870357 0.2797383 0.2960434 0.4106777 0.4860241    0
## MARS          0.09164825 0.2105942 0.3252409 0.3311404 0.4658253 0.5424713    0
## Random_Forest 0.11671947 0.1957358 0.2733796 0.2926219 0.3956843 0.4862077    0
```

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



Test set performance

```
tibble(model = c("Linear", "Enet", "GAM", "MARS", "Random Forest"),
       mse = c(linear_ts_mse, enet_ts_mse, gam_ts_mse, mars_ts_mse, boost_ts_mse)) %>%
  arrange(mse) %>%
  kable(caption = "Test set performance")
```

Table 6: Test set performance

model	mse
Random Forest	606.3513
MARS	621.5237
GAM	655.4724
Linear	762.7512
Enet	770.7180