

# Linear Models

Yangyang Chen

2024-03-25

To gain a better understanding of the factors that predict recovery time from COVID-19 illness, a study was designed to combine three existing cohort studies that have been tracking participants for several years. The study collects recovery information through questionnaires and medical records, and leverages existing data on personal characteristics prior to the pandemic.

In this project, we predict the recovery time based on important risk factors. The training data is in “training\_df”, and the test data is in “training\_df”. The response is in the column “Time to recovery (tt\_recovery\_time)”, and other variables can be used as predictors. The variable definitions can be found in “dictionary.txt”.

First, we import the data and adjust the variable type. As we want to compare different models afterwards, we use caret.

```
set.seed(2716)
# Load Data
load("./Data/recovery.RData")
dat <- dat |>
  mutate(gender = as_factor(gender),
         diabetes = as_factor(diabetes),
         hypertension = as_factor(hypertension),
         vaccine = as_factor(vaccine),
         severity = as_factor(severity)) |>
  select(-id)
```

```
# data splitting
data.split = initial_split(dat, prop = 0.8)
training_data = training(data.split)
testing_data = testing(data.split)
```

```
# check missing values
training_data |> is.na() |> sum()
```

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

```
testing_data |> is.na() |> sum()
```

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

```
# summary of training and testing data
training_data |> summary()
```

##	age	gender	race	smoking	height	weight
##	Min. :42.00	0:1235	1:1569	0:1442	Min. :147.8	Min. : 57.80
##	1st Qu.:57.00	1:1165	2: 134	1: 700	1st Qu.:166.1	1st Qu.: 75.30
##	Median :60.00		3: 475	2: 258	Median :170.0	Median : 79.80
##	Mean :60.13		4: 222		Mean :170.0	Mean : 80.01

```
## 3rd Qu.:63.00          3rd Qu.:174.0  3rd Qu.: 84.83
## Max. :79.00          Max. :188.6  Max. :103.70
##      bmi      hypertension diabetes      SBP      LDL      vaccine
## Min. :18.80  0:1223      0:2032  Min. :105.0  Min. : 28.0  0: 961
## 1st Qu.:25.80  1:1177      1: 368  1st Qu.:125.0  1st Qu.: 98.0  1:1439
## Median :27.60          Median :130.0  Median :111.0
## Mean :27.76          Mean :130.3  Mean :110.8
## 3rd Qu.:29.52          3rd Qu.:136.0  3rd Qu.:125.0
## Max. :36.70          Max. :156.0  Max. :171.0
## severity      study      recovery_time
## 0:2145  Length:2400      Min. : 2.00
## 1: 255  Class :character  1st Qu.: 30.00
##      Mode :character  Median : 39.00
##      Mean : 42.03
##      3rd Qu.: 49.00
##      Max. :365.00
```

```
testing_data |> summary()
```

```
##      age      gender race      smoking      height      weight
## Min. :47.00  0:309   1:398  0:380  Min. :148.1  Min. : 55.90
## 1st Qu.:57.00  1:291   2: 24  1:159  1st Qu.:165.8  1st Qu.: 75.10
## Median :60.00          3:129  2: 61  Median :169.7  Median : 79.55
## Mean :60.47          4: 49      Mean :169.6  Mean : 79.75
## 3rd Qu.:64.00          3rd Qu.:173.7  3rd Qu.: 84.50
## Max. :74.00          Max. :186.1  Max. :100.00
##      bmi      hypertension diabetes      SBP      LDL      vaccine
## Min. :19.20  0:285      0:505  Min. :105  Min. : 57.0  0:251
## 1st Qu.:25.90  1:315      1: 95  1st Qu.:126  1st Qu.: 95.0  1:349
## Median :27.80          Median :131  Median :108.0
## Mean :27.78          Mean :131  Mean :108.9
## 3rd Qu.:29.50          3rd Qu.:136  3rd Qu.:123.0
## Max. :38.90          Max. :154  Max. :178.0
## severity      study      recovery_time
## 0:534  Length:600      Min. : 2.00
## 1: 66  Class :character  1st Qu.: 31.00
##      Mode :character  Median : 39.00
##      Mean : 42.73
##      3rd Qu.: 49.00
##      Max. :330.00
```

```
# training data
```

```
train.x = model.matrix(recovery_time ~ ., training_data)[, -1]
train.y = training_data$recovery_time
```

```
# test data
```

```
test.x = model.matrix(recovery_time ~ ., testing_data)[, -1]
test.y = testing_data$recovery_time
```

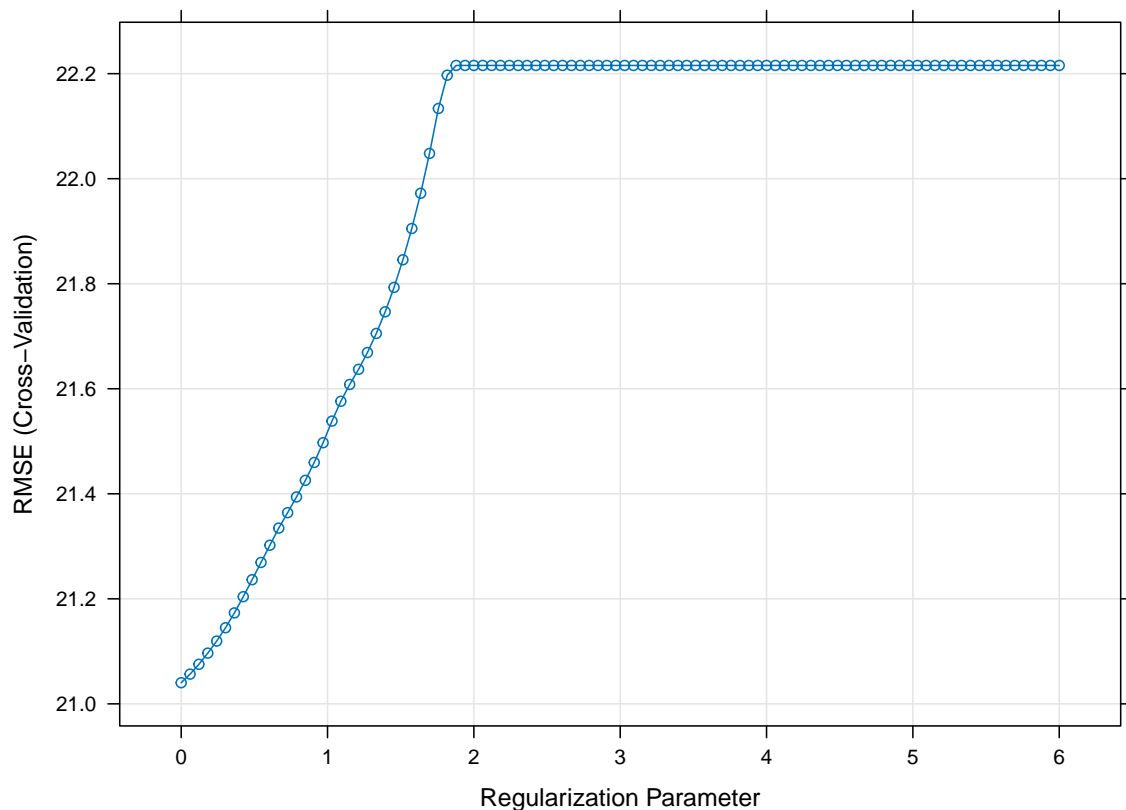
```
# cross validation
```

```
ctrl = trainControl(method = "cv", number = 10)
ctrl_1SE = trainControl(method = "cv", number = 10,
                        selectionFunction = "oneSE")
```

There is no missing data in both datasets. The training dataset has 2400 observation and 16 variables, and the test dataset has 600 samples and 16 variables.

## Lasso Regression Model

```
set.seed(2716)
lasso.fit = training_data |>
  train(recovery_time ~ ., data = _, method = "glmnet",
        tuneGrid = expand.grid(
          alpha = 1,
          lambda = exp(seq(6, 0, length = 100))),
        trControl = ctrl,
        preProcess = c("center", "scale"))
plot(lasso.fit, xTrans = log)
```



The flattening in the curve occurs because the lasso regression model has reached a stable solution where no more coefficients become zero as the regularization parameter increases. At this point, further increasing the regularization parameter does not change the set of non-zero coefficients or their values, resulting in a flat region where the model's performance remains constant. This behavior indicates that the model has achieved the optimal level of sparsity, with the regularization parameter value at the start of the flat region corresponding to the desired sparse solution.

```
lasso.pred = predict(lasso.fit, newdata = testing_data)
mse.lasso = mean((test.y - lasso.pred) ^ 2)
```

The selected tuning parameter is 1, and the test error (MSE) is 608.0764. Now, we apply the 1SE rule and refit the model.

```
lasso_1SE.fit = training_data |>
  train(recovery_time ~ .,
        data = _,
        method = "glmnet",
        tuneGrid = expand.grid(
```

```

alpha = 1,
lambda = exp(seq(6, 0, length = 100))),
trControl = ctrl_1SE)
lasso.coef_1SE = coef(lasso_1SE.fit$finalModel,
                      s = lasso_1SE.fit$bestTune$lambda)

```

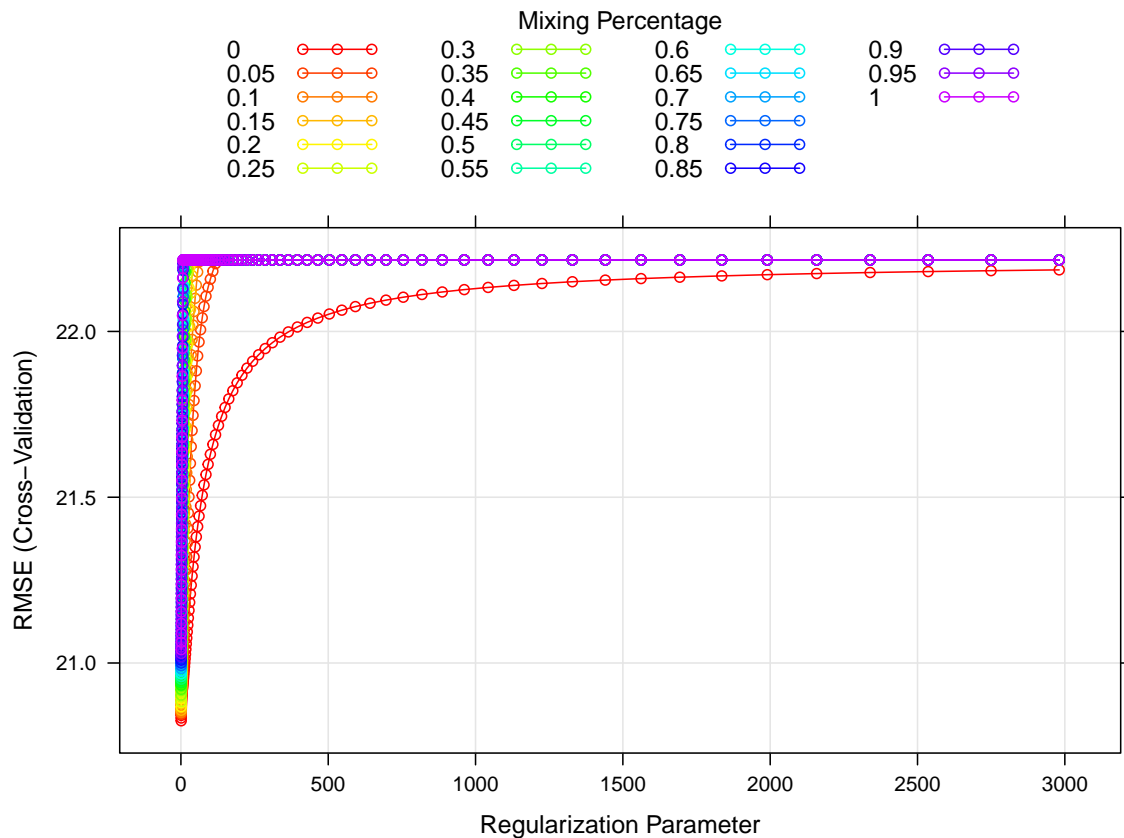
There exists problem in our model. When the 1SE rule is applied, the selected tuning parameter is  $\alpha = 1, \lambda = 403.43$ , and 0 predictors are included in the model.

## Elastic Net Model

```

set.seed(2716)
enet.fit = training_data |>
  train(recovery_time ~ ., data = _, method = "glmnet",
        tuneGrid = expand.grid(
          alpha = seq(0, 1, length = 21),
          lambda = exp(seq(8, 0, length = 100))),
        preProcess = c("center", "scale"),
        trControl = ctrl)
myCol = rainbow(25)
myPar =
  list(superpose.symbol = list(col = myCol),
        superpose.line = list(col = myCol))
plot(enet.fit, par.settings = myPar)

```



The vertical line pattern in the graph is a characteristic behavior of the elastic net model. It occurs because the elastic net combines lasso and ridge regularization, which can lead to sparse solutions with some coefficients

becoming exactly zero. As the regularization parameter increases, the model transitions between different sparse solutions, causing sudden drops or jumps in the cross-validation error curve.

```
enet.pred = predict(enet.fit, newdata = testing_data)
mse.enet = mean((test.y - enet.pred) ^ 2)
```

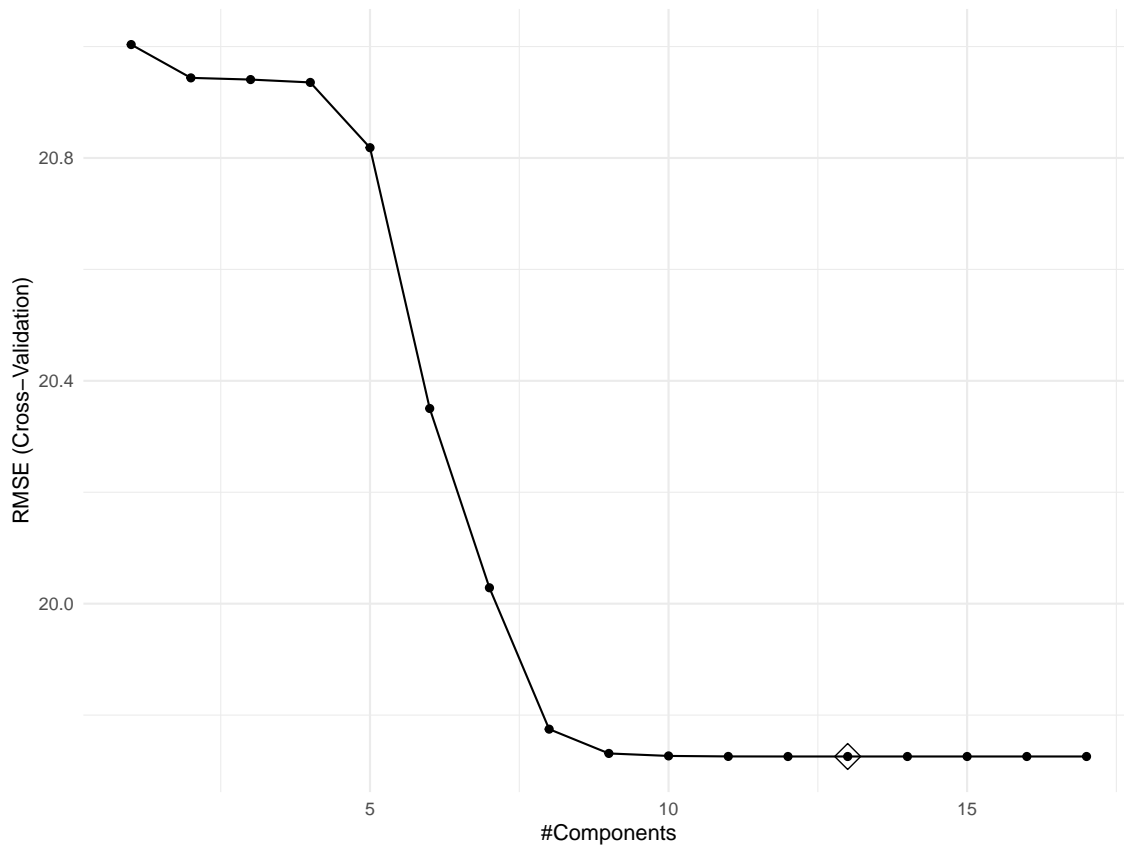
The selected tuning parameter is  $\alpha = 0$ ,  $\lambda = 1$ , and the test error (MSE) is 589.642.

## Partial Least Square

```
set.seed(2716)

pls.fit = train(train.x, train.y, method = "pls",
  tuneGrid = data.frame(ncomp = 1:17),
  trControl = ctrl, preProcess = c("center", "scale"))

ggplot(pls.fit, highlight = T)
```



```
pls.pred = predict(pls.fit, newdata = test.x)
mse.pls = mean((test.y - pls.pred) ^ 2)
```

As illustrated in the plot, 11 components are included in my model, and the test error (MSE) is 472.1037.

## Linear Model Comparison

Here, we compare the CV results of different models and choose the model with the smallest median RMSE.

```

resamp =
  resamples(list(lasso = lasso.fit,
                 lasso_1SE = lasso_1SE.fit,
                enet =enet.fit,
                 pls = pls.fit))
summary(resamp)

```

```

##
## Call:
## summary.resamples(object = resamp)
##
## Models: lasso, lasso_1SE, enet, pls
## Number of resamples: 10
##
## MAE
##           Min.   1st Qu.   Median     Mean 3rd Qu.     Max. NA's
## lasso      11.91749 12.22876 12.78822 13.10307 13.97511 14.75607    0
## lasso_1SE  12.74388 13.06221 13.23373 13.58633 13.55857 15.46776    0
## enet       11.82640 12.24000 12.86273 13.14786 14.09859 14.70406    0
## pls       12.00552 12.42079 13.07887 13.06909 13.69203 14.12036    0
##
## RMSE
##           Min.   1st Qu.   Median     Mean 3rd Qu.     Max. NA's
## lasso      16.36102 17.35644 19.62606 21.04028 24.59723 27.64478    0
## lasso_1SE  17.77243 19.70868 20.20005 22.16885 21.08886 32.28592    0
## enet       16.09922 17.14833 19.50831 20.82568 24.22164 27.34263    0
## pls       15.94665 16.77630 19.19268 19.72570 22.05070 24.84530    0
##
## Rsquared
##           Min.   1st Qu.   Median     Mean 3rd Qu.     Max. NA's
## lasso      0.07035909 0.08459142 0.1095733 0.1133160 0.1199068 0.1790213    0
## lasso_1SE      NA         NA         NA      NaN      NA      NA    10
## enet       0.08321485 0.10886399 0.1216388 0.1284815 0.1374788 0.1954873    0
## pls       0.10933283 0.16901995 0.2338977 0.2315719 0.2969252 0.3415088    0

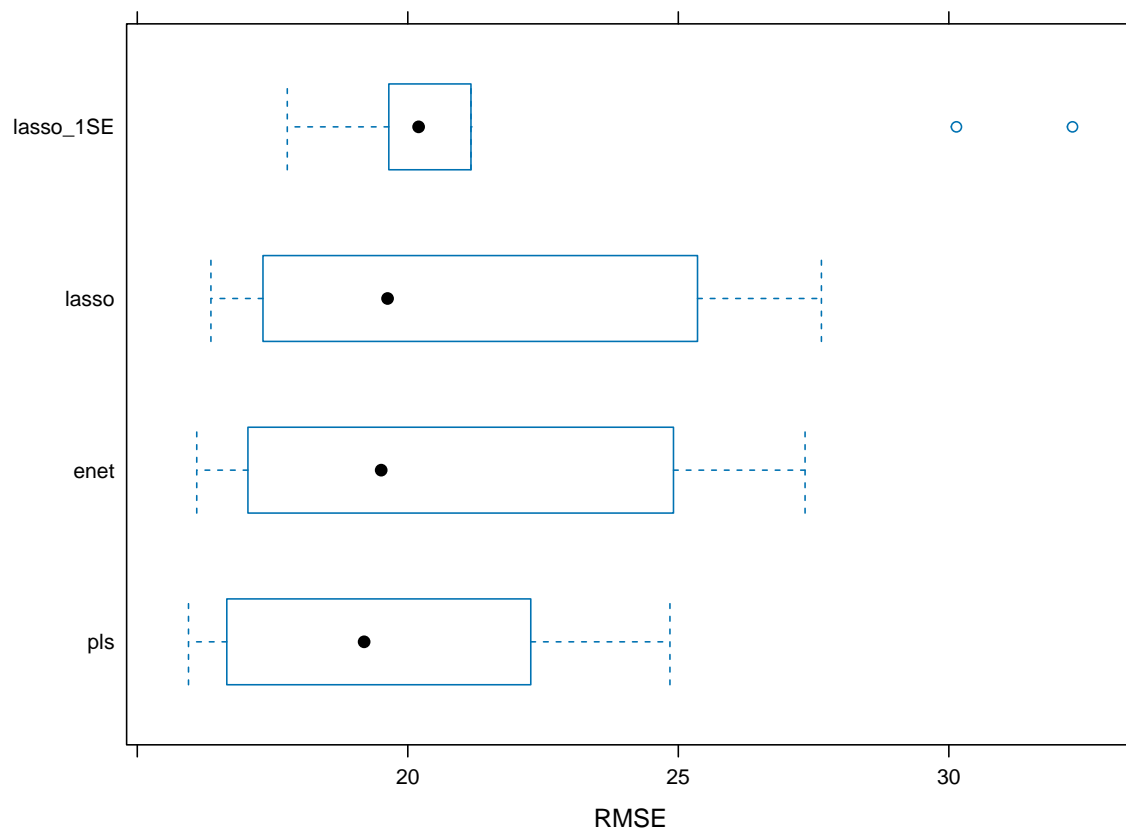
```

Using bw-plot to compare their RMSE.

```

bwplot(resamp, metric = "RMSE")

```



Hence, we selected partial least square model as it has smallest RMSE.

## MARS

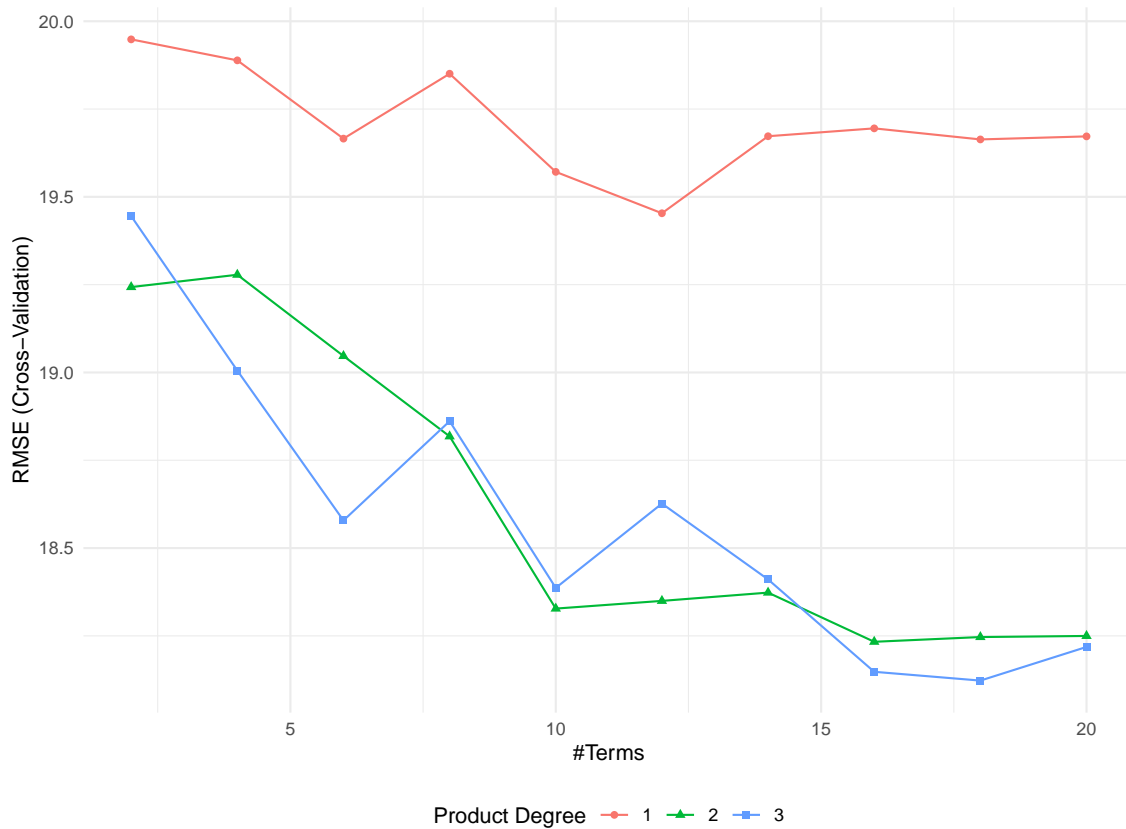
Now, train a multivariate adaptive regression spline (MARS) model to predict the response variable.

Since there are two tuning parameters associated with the MARS model: the degree of interactions and the number of retained terms, we need to perform a grid search to identify the optimal combination of these hyperparameters that minimize prediction error.

```
ctrl = trainControl(method = "cv", number = 10)
mars.grid = expand.grid(degree = 1 : 3, nprune = seq(2, 20, by = 2))
set.seed(2716)
mars.fit = train(train.x, train.y, method = "earth", tuneGrid = mars.grid, trControl = ctrl)

## Loading required package: earth
## Loading required package: Formula
## Loading required package: plotmo
## Loading required package: plotrix
##
## Attaching package: 'plotrix'
## The following object is masked from 'package:scales':
##
##   rescale
## Loading required package: TeachingDemos
```

```
ggplot(mars.fit)
```



```
summary(mars.fit$finalModel)
```

```
## Call: earth(x=matrix[2400,17], y=c(69,9,7,43,36,...), keepxy=TRUE, degree=3,
##          nprune=18)
##
##
## coefficients
## (Intercept)          17.425888
## gender1            -3.838226
## vaccine1           -5.523433
## severity1           5.146386
## h(bmi-25.7)          6.207509
## h(31.1-bmi)          4.065175
## h(bmi-35.4)         27.540176
## h(bmi-31.1) * studyB -305.070403
## h(bmi-25.7) * h(140-SBP) -0.064119
## smoking1 * h(bmi-31.1) * studyB 15.366973
## smoking2 * h(bmi-31.1) * studyB 28.176098
## h(age-61) * h(bmi-31.1) * studyB 5.249927
## h(weight-78.7) * h(bmi-31.1) * studyB 23.765436
## h(89.3-weight) * h(bmi-31.1) * studyB 24.335697
## h(weight-89.3) * h(bmi-31.1) * studyB -25.433893
## h(bmi-31.1) * h(LDL-94) * studyB -5.448260
## h(bmi-31.1) * h(94-LDL) * studyB 5.533714
## h(bmi-31.1) * h(LDL-84) * studyB 5.716112
##
```



```
## Selected 18 of 25 terms, and 11 of 17 predictors (nprune=18)
## Termination condition: Reached nk 35
## Importance: bmi, studyB, LDL, age, vaccine1, weight, smoking1, gender1, ...
## Number of terms at each degree of interaction: 1 6 2 9
## GCV 261.8665    RSS 605903.8    GRSq 0.4893553    RSq 0.5072879
```

```
## Coefficient of the MARS model
```

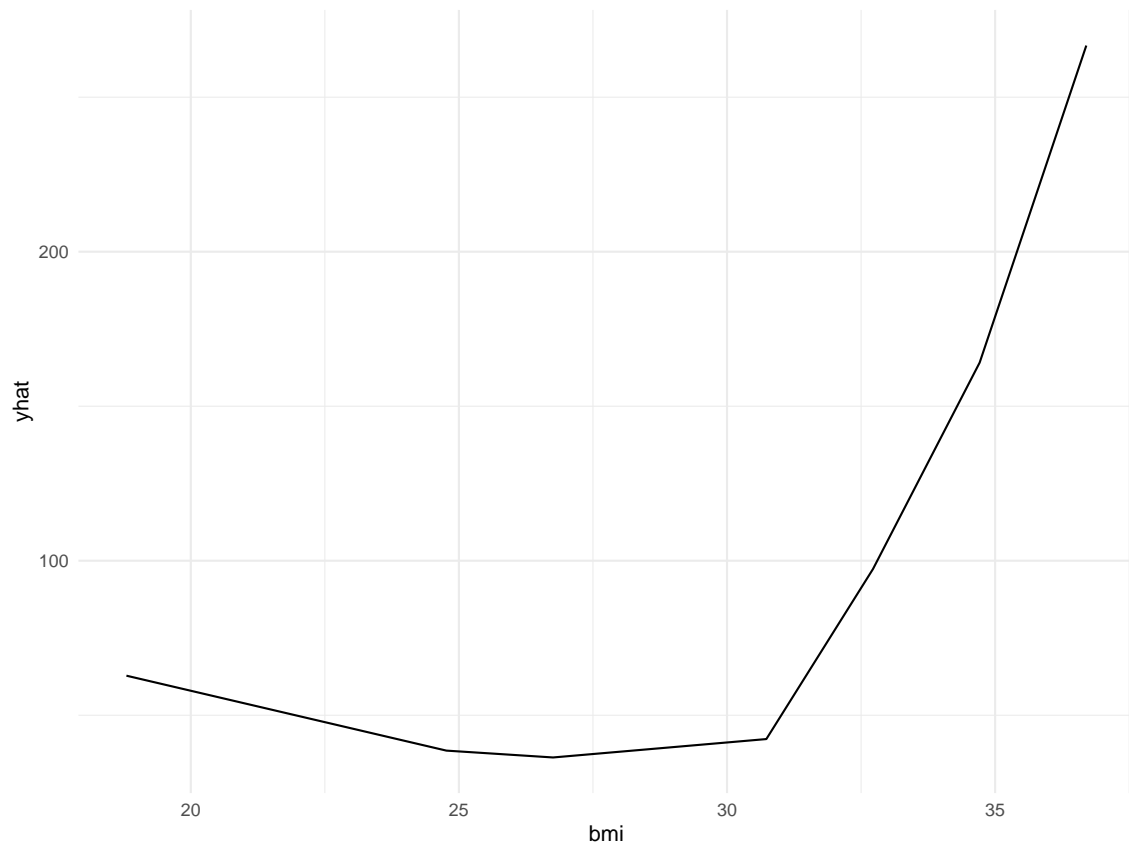
```
coef(mars.fit$finalModel)
```

```
##              (Intercept)              h(31.1-bmi)
##              17.42588791              4.06517510
##              h(bmi-31.1) * studyB      h(age-61) * h(bmi-31.1) * studyB
##              -305.07040296              5.24992718
##              h(bmi-25.7)                vaccine1
##              6.20750882                -5.52343301
##              h(bmi-31.1) * h(LDL-94) * studyB      h(bmi-31.1) * h(94-LDL) * studyB
##              -5.44825972              5.53371358
## h(weight-89.3) * h(bmi-31.1) * studyB h(89.3-weight) * h(bmi-31.1) * studyB
##              -25.43389302              24.33569724
##              h(bmi-31.1) * h(LDL-84) * studyB      smoking1 * h(bmi-31.1) * studyB
##              5.71611207              15.36697296
##              gender1                      severity1
##              -3.83822562              5.14638556
##              smoking2 * h(bmi-31.1) * studyB      h(bmi-25.7) * h(140-SBP)
##              28.17609759              -0.06411943
## h(weight-78.7) * h(bmi-31.1) * studyB      h(bmi-35.4)
##              23.76543599              27.54017572
```

The MARS model selects 2 of 25 terms, and 2 of 18 predictors. The most important variables are *bmi* (Body Mass Index; BMI = weight (in kilograms) / height (in meters) squared) and *studyB* (The study (A/B) that the participant belongs to).

To better understand the relationship between these features and outcome, we can create partial dependence plots (PDPs) for each feature individually and also an interaction PDP. To simplify, here we only present the PDP for number of full-time undergraduates *sbp*.

```
pdp::partial(mars.fit, pred.var = c("bmi"), grid.resolution = 10) |> autoplot()
```



Using the

final model, we can predict on the test data.

```
pred.mars = predict(mars.fit, newdata = test.x)
mse.mars = mean((pred.mars - test.y) ^ 2)
```

The test error measured by MSE using the final MARS model is 431.2879.

## GAM

```
set.seed(2716)
gam.fit = train(train.x, train.y, method = "gam", trControl = ctrl)
```

```
## Loading required package: mgcv
## Loading required package: nlme
##
## Attaching package: 'nlme'
## The following object is masked from 'package:dplyr':
##
## collapse
## This is mgcv 1.9-0. For overview type 'help("mgcv-package")'.
```

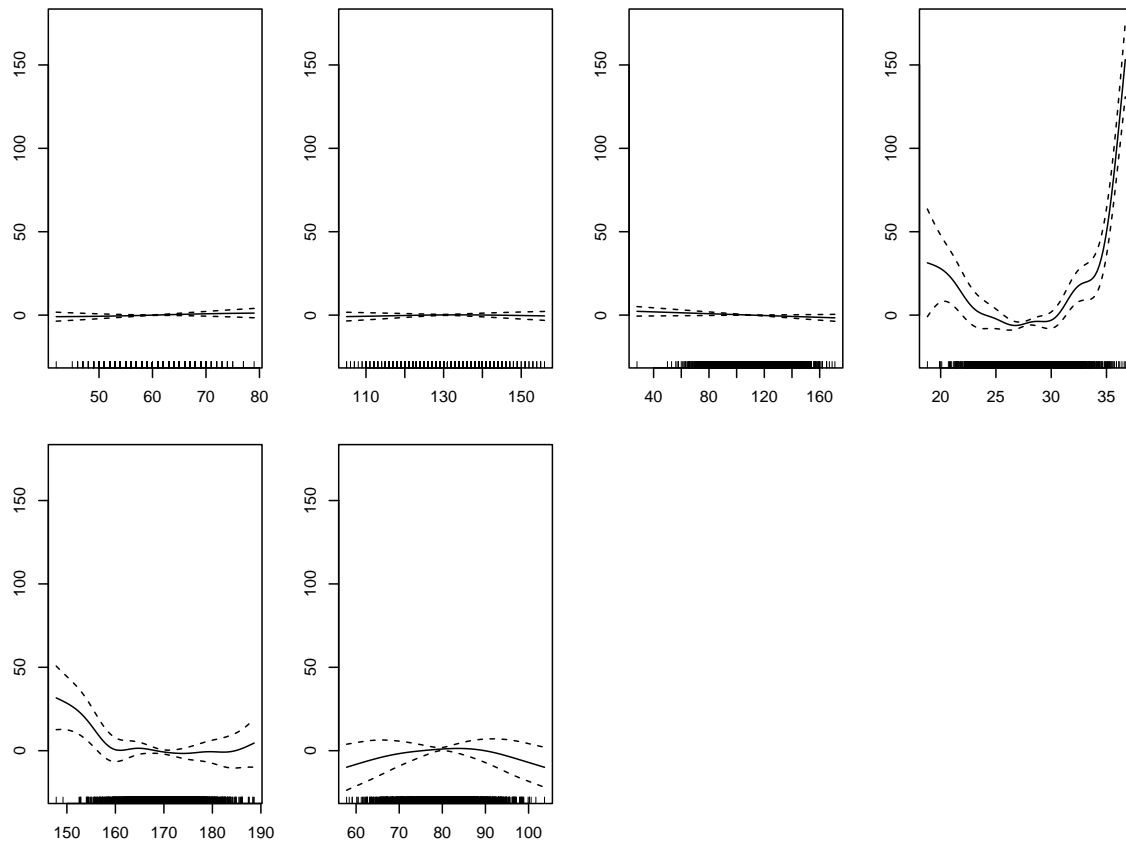
```
summary(gam.fit$finalModel)
```

```
##
## Family: gaussian
## Link function: identity
##
```

```

## Formula:
## .outcome ~ gender1 + race2 + race3 + race4 + smoking1 + smoking2 +
##      hypertension1 + diabetes1 + vaccine1 + severity1 + studyB +
##      s(age) + s(SBP) + s(LDL) + s(bmi) + s(height) + s(weight)
##
## Parametric coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   43.5319     1.0022  43.437 < 2e-16 ***
## gender1       -3.9286     0.7579  -5.184 2.36e-07 ***
## race2          2.4897     1.6763   1.485 0.13761
## race3          0.2184     0.9730   0.224 0.82239
## race4         -1.0791     1.3330  -0.810 0.41831
## smoking1       2.4333     0.8578   2.837 0.00460 **
## smoking2       3.7986     1.2536   3.030 0.00247 **
## hypertension1  2.2691     1.0394   2.183 0.02913 *
## diabetes1      -0.9912     1.0520  -0.942 0.34617
## vaccine1       -6.2577     0.7773  -8.051 1.29e-15 ***
## severity1      5.6267     1.2306   4.572 5.07e-06 ***
## studyB         4.1707     0.8026   5.197 2.20e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##              edf Ref.df      F  p-value
## s(age)         0.6260      9  0.153 0.115489
## s(SBP)         0.5946      9  0.079 0.293041
## s(LDL)         0.7589      9  0.277 0.068772 .
## s(bmi)         8.8974      9 67.421 < 2e-16 ***
## s(height)      6.6451      9  3.220 2.26e-05 ***
## s(weight)      2.3895      9  1.614 0.000227 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.334   Deviance explained = 34.3%
## GCV = 345.74   Scale est. = 341.14      n = 2400
par(mar = c(2, 2, 2, 2), mfrow = c(2, 4))
plot(gam.fit$finalModel)

```



It could be observed that certain variables (*age*, *sbp*, *ldl*) have no relationship with the *recovery\_time*, *bmi*, and *height* both have a positive relationship with *recovery\_time*.

Using the final model, we can predict on the test data.

```
pred.gam = predict(gam.fit, newdata = test.x)
mse.gam = mean((pred.gam - test.y) ^ 2)
```

The test error measured by MSE using the final GAM is 416.6977.