Linear Models

Yangyang Chen

2024-03-23

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.

```r
library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```r
library(glmnet)
```

```
## Loading required package: Matrix

## Loaded glmnet 4.1-8
```

```r
library(tidymodels)
```

```
## -- Attaching packages -------------------------------------- tidymodels 1.1.1 --

## v broom        1.0.5     v rsample      1.2.0
## v dials        1.2.0     v tibble       3.2.1
## v dplyr        1.1.4     v tidyr        1.3.1
## v infer        1.0.5     v tune         1.1.2
## v modeldata    1.2.0     v workflows    1.1.3
## v parsnip      1.1.1     v workflowsets 1.0.1
## v purrr        1.0.2     v yardstick    1.2.0
## v recipes      1.0.8

## -- Conflicts ----------------------------------------- tidymodels_conflicts() --
## x purrr::discard()         masks scales::discard()
## x tidyr::expand()          masks Matrix::expand()
## x dplyr::filter()          masks stats::filter()
## x dplyr::lag()             masks stats::lag()
## x purrr::lift()            masks caret::lift()
## x tidyr::pack()            masks Matrix::pack()
## x yardstick::precision()   masks caret::precision()
## x yardstick::recall()      masks caret::recall()
## x yardstick::sensitivity() masks caret::sensitivity()
## x yardstick::specificity() masks caret::specificity()
## x recipes::step()          masks stats::step()
## x tidyr::unpack()          masks Matrix::unpack()
## x recipes::update()        masks Matrix::update(), stats::update()
## * Learn how to get started at https://www.tidymodels.org/start/
```

```
set.seed(2024)
load("~/Desktop/Data Science II/Group_Project/recovery.RData")
dat = dat |> janitor::clean_names()

# 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()
```

```
##        id              age            gender        race     smoking
##  Min.   :   1.0   Min.   :42.00   Min.   :0.0000   1:1578   0:1444
##  1st Qu.: 741.8   1st Qu.:57.00   1st Qu.:0.0000   2: 128   1: 700
##  Median :1484.5   Median :60.00   Median :0.0000   3: 489   2: 256
##  Mean   :1490.8   Mean   :60.28   Mean   :0.4863   4: 205
##  3rd Qu.:2230.5   3rd Qu.:63.00   3rd Qu.:1.0000
##  Max.   :3000.0   Max.   :79.00   Max.   :1.0000
##      height          weight            bmi         hypertension
##  Min.   :147.8   Min.   : 55.90   Min.   :18.80   Min.   :0.0000
##  1st Qu.:166.0   1st Qu.: 75.30   1st Qu.:25.90   1st Qu.:0.0000
##  Median :169.9   Median : 79.75   Median :27.70   Median :0.0000
##  Mean   :169.9   Mean   : 79.99   Mean   :27.79   Mean   :0.4958
##  3rd Qu.:173.9   3rd Qu.: 84.90   3rd Qu.:29.60   3rd Qu.:1.0000
##  Max.   :188.6   Max.   :103.70   Max.   :38.90   Max.   :1.0000
##     diabetes          sbp             ldl            vaccine
##  Min.   :0.00   Min.   :105.0   Min.   : 28.0   Min.   :0.0000
##  1st Qu.:0.00   1st Qu.:125.8   1st Qu.: 97.0   1st Qu.:0.0000
##  Median :0.00   Median :130.0   Median :110.0   Median :1.0000
##  Mean   :0.15   Mean   :130.4   Mean   :110.7   Mean   :0.5933
##  3rd Qu.:0.00   3rd Qu.:136.0   3rd Qu.:125.0   3rd Qu.:1.0000
##  Max.   :1.00   Max.   :156.0   Max.   :178.0   Max.   :1.0000
##     severity          study           recovery_time
##  Min.   :0.0000   Length:2400        Min.   :  2.00
##  1st Qu.:0.0000   Class :character   1st Qu.: 31.00
##  Median :0.0000   Mode  :character   Median : 39.00
##  Mean   :0.1087                      Mean   : 42.02
##  3rd Qu.:0.0000                      3rd Qu.: 49.00
##  Max.   :1.0000                      Max.   :365.00
```

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

```
##        id              age            gender            race    smoking
##  Min.   :   4.0   Min.   :45.00   Min.   :0.0000   1:389   0:378
##  1st Qu.: 801.5   1st Qu.:57.00   1st Qu.:0.0000   2: 30   1:159
##  Median :1538.0   Median :60.00   Median :0.0000   3:115   2: 63
##  Mean   :1539.3   Mean   :59.86   Mean   :0.4817   4: 66
##  3rd Qu.:2312.2   3rd Qu.:63.00   3rd Qu.:1.0000
##  Max.   :2997.0   Max.   :72.00   Max.   :1.0000
##      height          weight            bmi         hypertension
##  Min.   :148.1   Min.   :60.20   Min.   :19.20   Min.   :0.0000
##  1st Qu.:166.2   1st Qu.:75.00   1st Qu.:25.50   1st Qu.:0.0000
##  Median :169.9   Median :79.80   Median :27.55   Median :1.0000
##  Mean   :170.1   Mean   :79.85   Mean   :27.67   Mean   :0.5033
##  3rd Qu.:174.0   3rd Qu.:84.50   3rd Qu.:29.50   3rd Qu.:1.0000
##  Max.   :188.5   Max.   :98.60   Max.   :35.90   Max.   :1.0000
##     diabetes            sbp             ldl            vaccine
##  Min.   :0.0000   Min.   :108.0   Min.   : 50.0   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:125.0   1st Qu.: 96.0   1st Qu.:0.0000
##  Median :0.0000   Median :131.0   Median :109.5   Median :1.0000
##  Mean   :0.1717   Mean   :130.7   Mean   :109.6   Mean   :0.6067
##  3rd Qu.:0.0000   3rd Qu.:136.0   3rd Qu.:123.0   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :155.0   Max.   :162.0   Max.   :1.0000
##     severity      study           recovery_time
##  Min.   :0.0   Length:600         Min.   :  2.00
##  1st Qu.:0.0   Class :character   1st Qu.: 30.00
##  Median :0.0   Mode  :character   Median : 40.00
##  Mean   :0.1                      Mean   : 42.76
##  3rd Qu.:0.0                      3rd Qu.: 49.00
##  Max.   :1.0                      Max.   :330.00
```

```r
# 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")
```

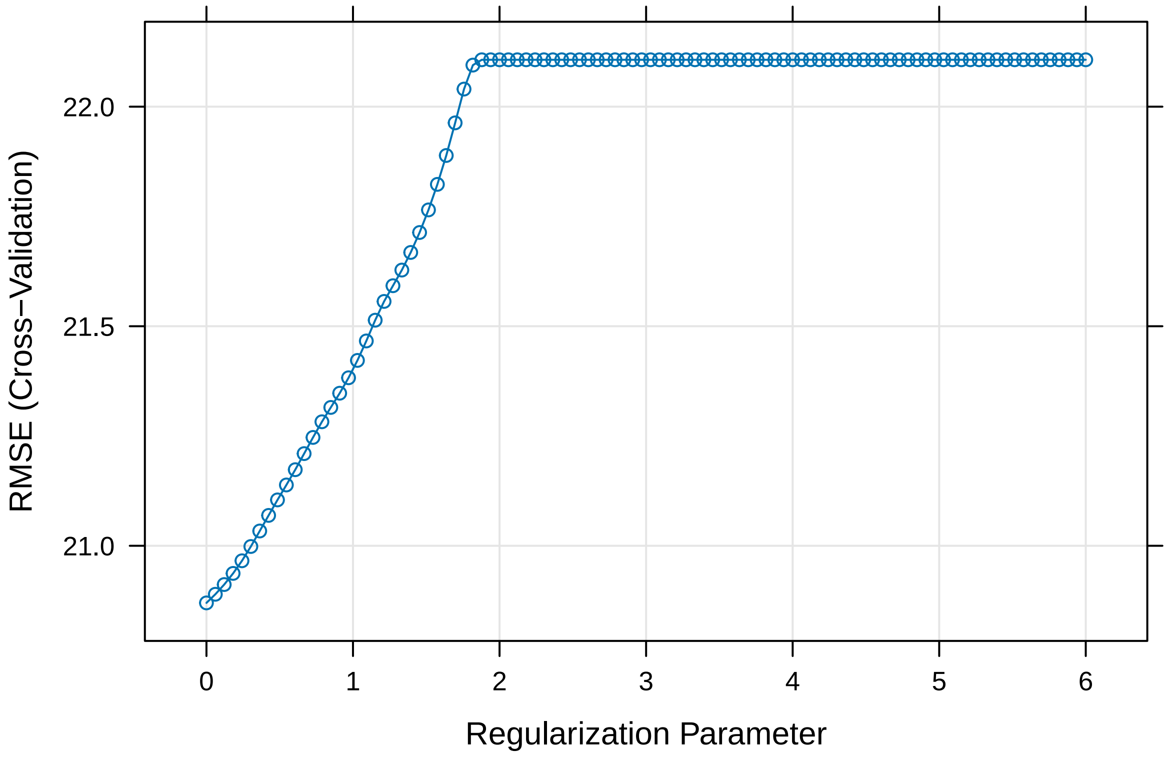
There is no missing data in both datasets. The training dataset has 2400 observation and 16 variables, and the test dataset has 600 samples and 16 variables.

Lasso Regression Model

```r
set.seed(2024)
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"))
```

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

```
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)
```

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

```
lasso.coef_1SE = coef(lasso_1SE.fit$finalModel,
                      s = lasso_1SE.fit$bestTune$lambda)
```

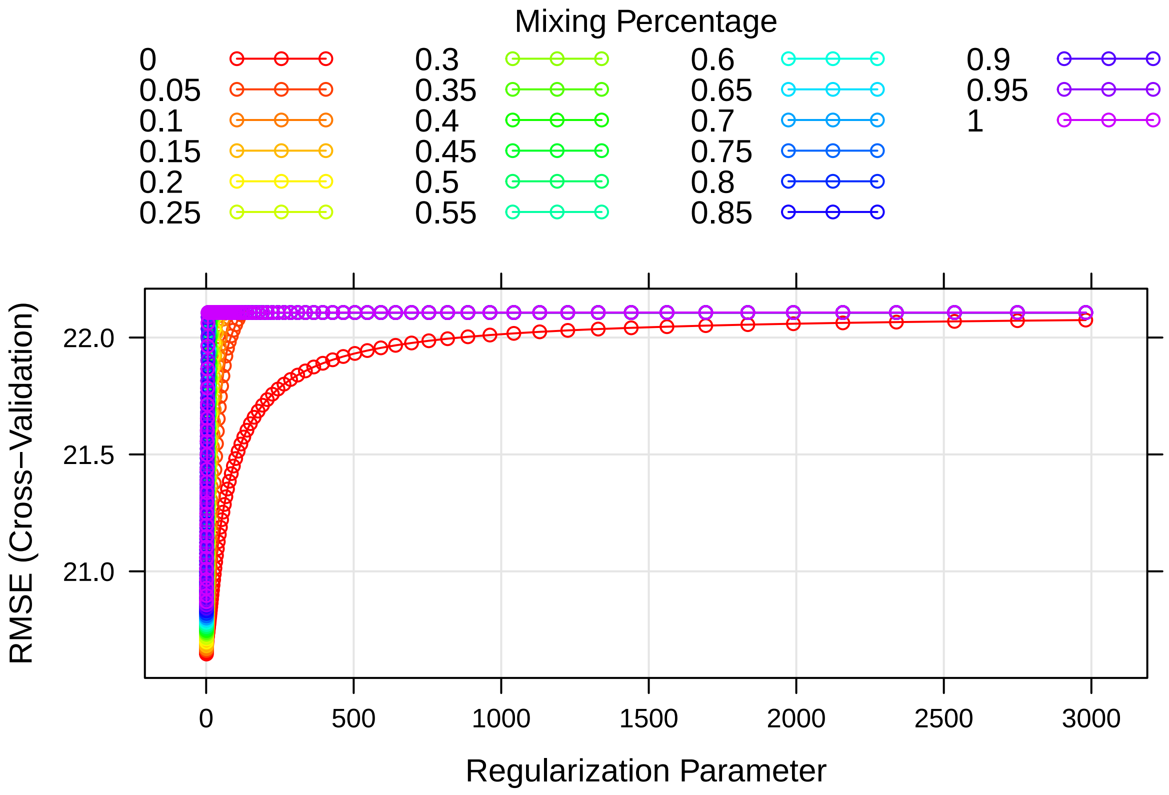
There exists problem in our model. When the 1SE rule is applied, the selected tuning parameter is $\alpha = 1, \lambda = 403.43$, and 0 predictors are included in the model.

Elastic Net Model

```
set.seed(2024)
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)
```

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

```
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. Now, we try to apply the 1SE rule and refit the model.

```
set.seed(2024)
enet_1SE.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_1SE)
```

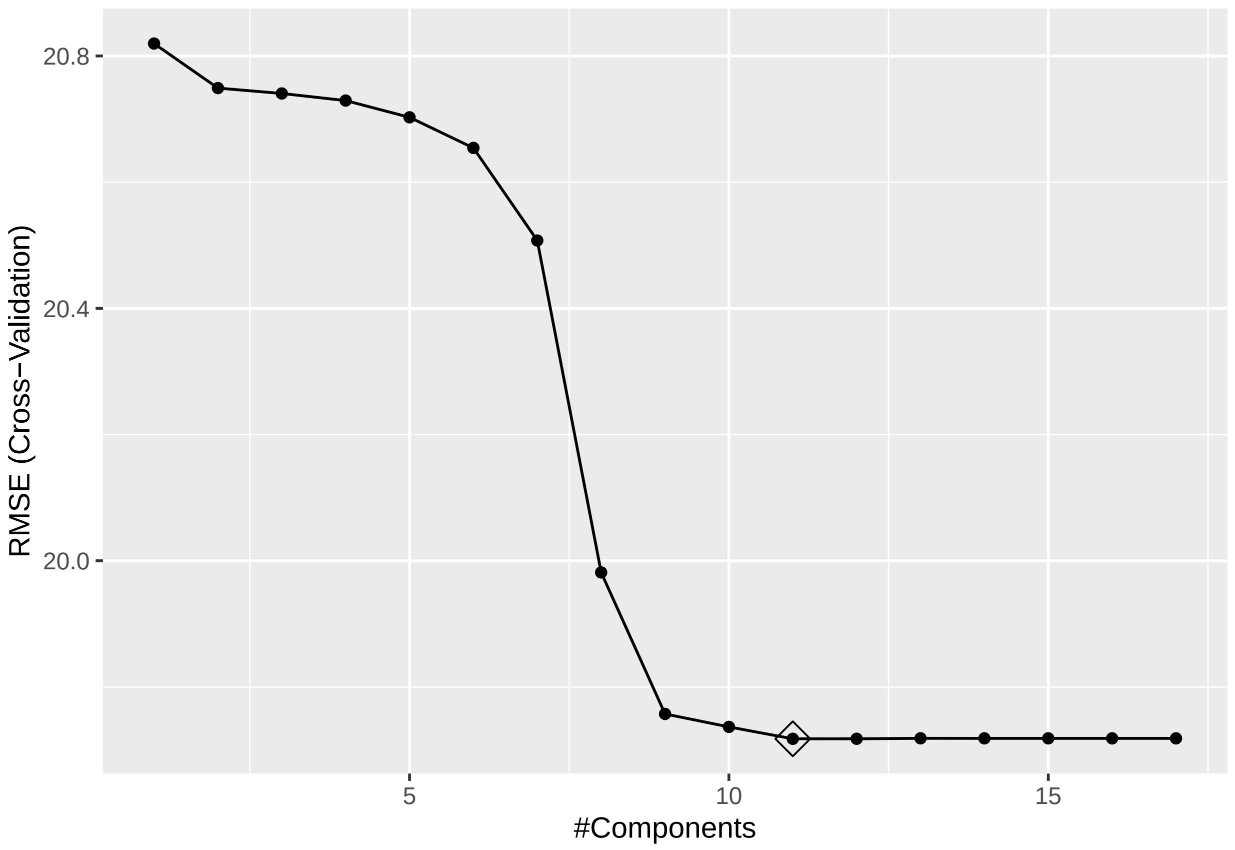
```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo,
## : There were missing values in resampled performance measures.
```

```
enet_1SE.pred = predict(enet_1SE.fit, newdata = testing_data)
mse.enet_1SE = mean((test.y - enet_1SE.pred) ^ 2)
```

When the 1SE rule is applied, the selected tuning parameter is $\alpha = 0$, $\lambda = 191.0481$, and the test error (MSE) is 654.9077.

Partial Least Square

```
set.seed(2024)

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

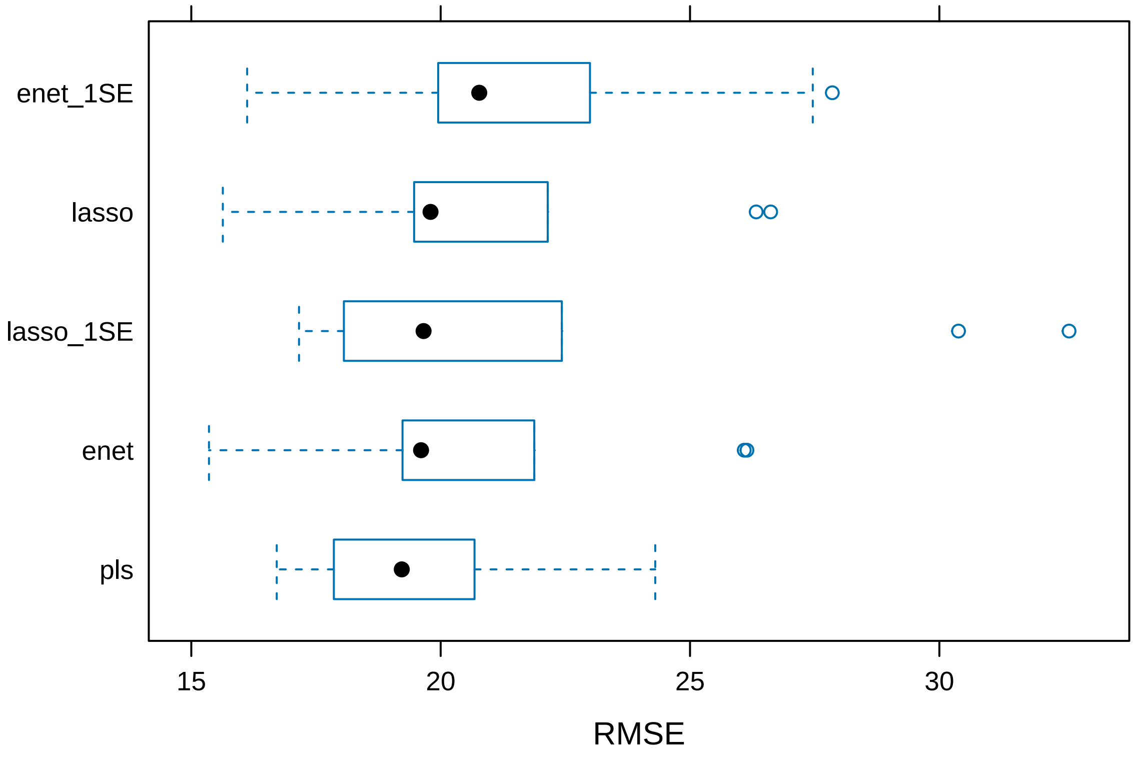
Here, we compare the CV results of different models and choose the model with the smallest median RMSE.

```
resamp =
  resamples(list(lasso = lasso.fit,
                 lasso_1SE = lasso_1SE.fit,
                 enet = enet.fit,
                 enet_1SE = enet_1SE.fit,
                 pls = pls.fit))
summary(resamp)
```

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: lasso, lasso_1SE, enet, enet_1SE, pls
## Number of resamples: 10
##
## MAE
##                Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## lasso     11.70820 12.80710 12.91981 12.98865 13.31698 14.11550    0
## lasso_1SE 12.02208 12.59751 13.18138 13.49821 13.78426 16.11071    0
## enet      11.61245 12.92027 13.01641 13.09761 13.50941 14.31622    0
## enet_1SE  11.89810 12.94601 13.07245 13.24746 13.72694 14.26906    0
## pls       11.73828 12.57180 13.04293 12.97226 13.36581 14.03871    0
##
## RMSE
##               Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## lasso     15.63217 19.47182 19.79685 20.86990 22.00463 26.61706    0
## lasso_1SE 17.16031 18.12459 19.65723 21.78256 22.37995 32.60110    0
## enet      15.35438 19.26202 19.60704 20.64619 21.75758 26.14188    0
## enet_1SE  16.12041 20.02620 20.77401 21.71048 22.90480 27.85372    0
## pls       16.71277 17.90735 19.22022 19.71794 20.53556 24.30313    0
##
## Rsquared
##                 Min.    1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## lasso     0.07871429 0.09239047 0.1087820 0.1188363 0.1385637 0.1964512    0
## lasso_1SE         NA         NA        NA       NaN        NA        NA   10
## enet      0.09101487 0.10983939 0.1291095 0.1318636 0.1408881 0.1941670    0
## enet_1SE  0.07217084 0.09373823 0.1192631 0.1199881 0.1304252 0.1780023    0
## pls       0.06761502 0.20238751 0.2266513 0.2120997 0.2458746 0.2837441    0
```

Using bw-plot to compare their RMSE.

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



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

MARS

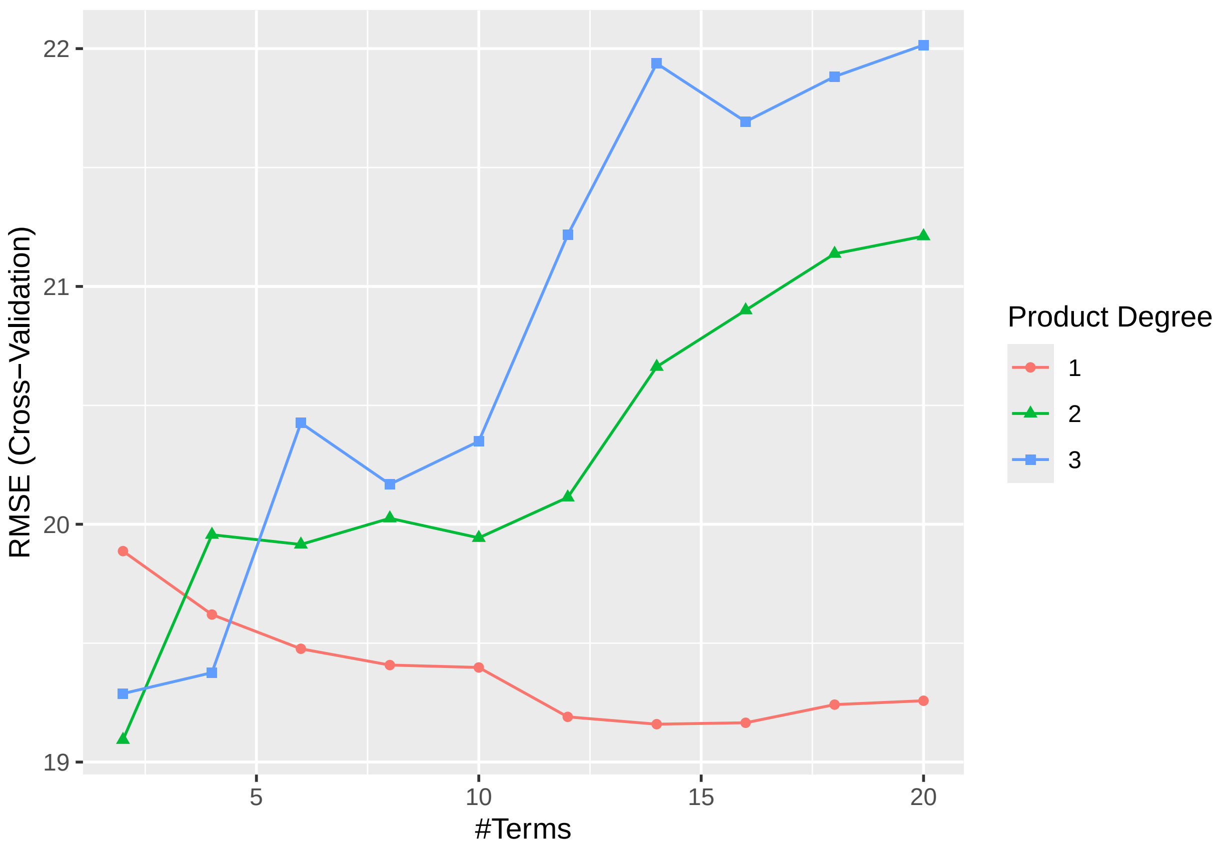
Now, train a multivariate adaptive regression spline (MARS) model to predict the response variable.

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(2024)
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[2400,18], y=c(33,44,33,27,6...), keepxy=TRUE, degree=2,
##             nprune=2)
##
##                       coefficients
## (Intercept)              39.72628
## h(bmi-30.3) * studyB     22.55495
##
## Selected 2 of 25 terms, and 2 of 18 predictors (nprune=2)
## Termination condition: Reached nk 37
## Importance: bmi, studyB, id-unused, age-unused, gender-unused, ...
## Number of terms at each degree of interaction: 1 0 1
## GCV 358.0043    RSS 856706.2    GRSq 0.2862284    RSq 0.2877152
```

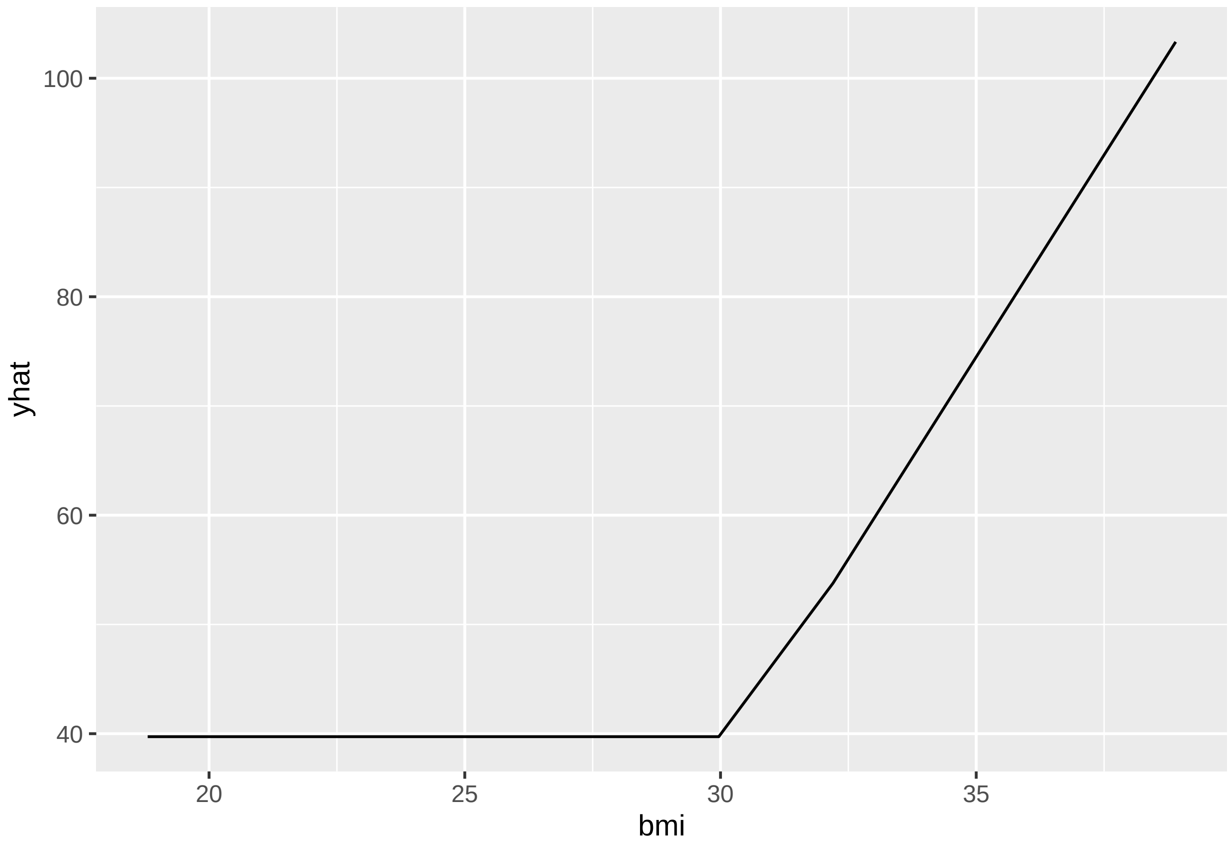
```r
## Coefficient of the MARS model
coef(mars.fit$finalModel)
```

```
##          (Intercept) h(bmi-30.3) * studyB
##             39.72628             22.55495
```

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 outstate, 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*.

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



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

```r
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

```r
set.seed(2024)
gam.fit = train(train.x, train.y, method = "gam", trControl = ctrl)
```

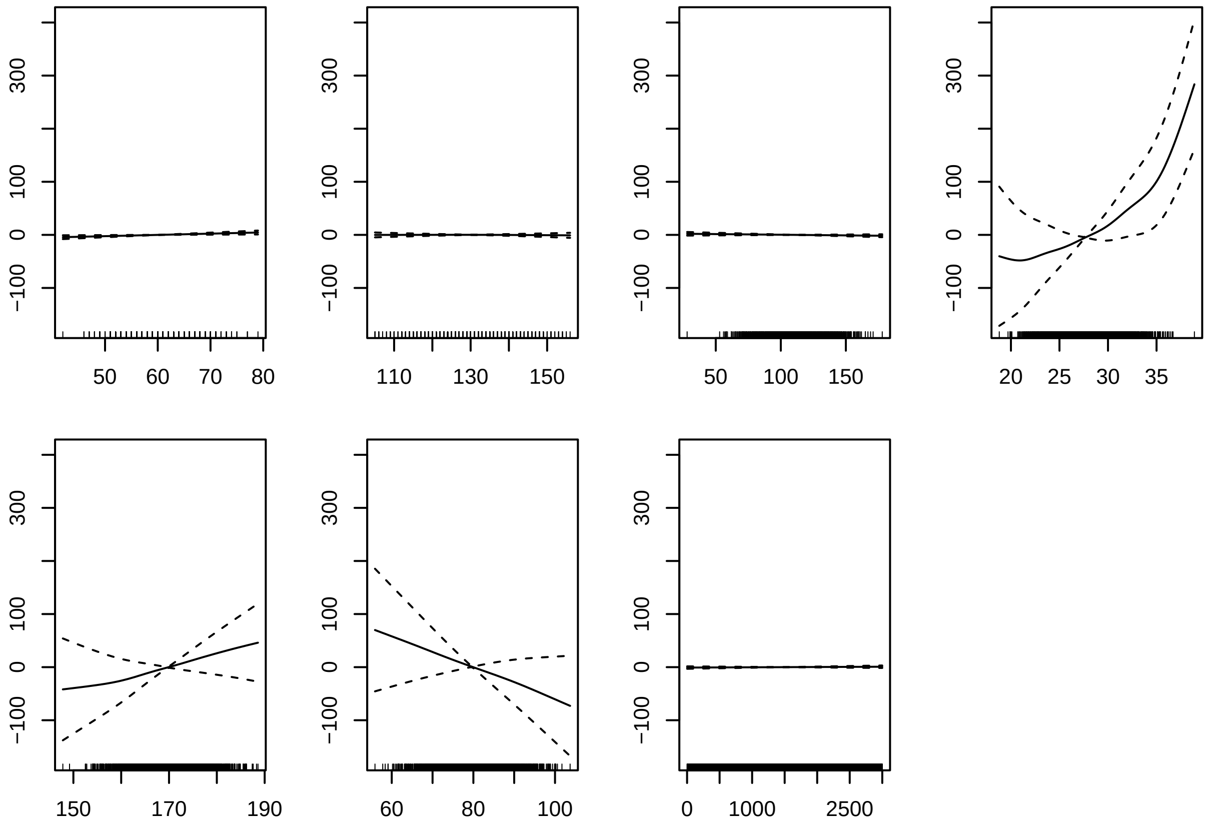
```
## Loading required package: mgcv

## 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")'.
```

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

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## .outcome ~ gender + race2 + race3 + race4 + smoking1 + smoking2 +
##     hypertension + diabetes + vaccine + severity + studyB + s(age) +
##     s(sbp) + s(ldl) + s(bmi) + s(height) + s(weight) + s(id)
##
## Parametric coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   43.4195     1.1263  38.552  < 2e-16 ***
## gender        -3.3873     0.7687  -4.406  1.1e-05 ***
## race2          2.6967     1.7301   1.559  0.11921
## race3         -0.6038     0.9746  -0.619  0.53566
## race4         -1.1766     1.3979  -0.842  0.40006
## smoking1       2.0218     0.8680   2.329  0.01993 *
## smoking2       4.1542     1.2772   3.253  0.00116 **
## hypertension   2.8107     1.2736   2.207  0.02741 *
## diabetes      -1.5319     1.0753  -1.425  0.15440
## vaccine       -6.3652     0.7830  -8.129  6.9e-16 ***
## severity       5.9555     1.2346   4.824  1.5e-06 ***
## studyB         3.8450     1.4076   2.732  0.00635 **
## ---
## 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)    1.000  1.000  5.949  0.0148 *
## s(sbp)    1.263  1.480  0.148  0.8750
## s(ldl)    1.000  1.000  1.467  0.2260
## s(bmi)    7.480  8.365 57.972  <2e-16 ***
## s(height) 4.517  5.604  1.925  0.0994 .
## s(weight) 2.576  3.498  2.914  0.0175 *
## s(id)     1.000  1.000  0.473  0.4917
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## R-sq.(adj) =  0.301   Deviance explained =   31%
## GCV = 355.11  Scale est. = 350.54    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.