ds2_hw2

Ruilian Zhang

3/6/2022

Contents


```
# data cleaning
df = read_csv("College.csv") %>%
  janitor::clean_names() %>%
  select(-college) %>%
  select(outstate, everything()) %>%
  na.omit()
```

```
## Rows: 565 Columns: 18
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (1): College
## dbl (17): Apps, Accept, Enroll, Top10perc, Top25perc, F.Undergrad, P.Undergr...
##
## 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 partition
indexTrain = createDataPartition(y = df$outstate,
                                 p = 0.8,
                                 list = FALSE)


train_df = df[indexTrain, ]
test_df = df[-indexTrain, ]

x_train = model.matrix(outstate ~ ., train_df)[ , -1]
y_train = train_df$outstate

x_test = model.matrix(outstate ~ ., test_df)[ , -1]
y_test = test_df$outstate
```

Exploratory data analysis (using train_df)

```
# data dimension and summary
dim(train_df)
```

```
## [1] 453  17
```

```
summary(train_df)
```

```
##     outstate          apps           accept          enroll
##  Min.   : 2340   Min.   :   81   Min.   :   72   Min.   :  35.0
##  1st Qu.: 9100   1st Qu.:  633   1st Qu.:  513   1st Qu.: 209.0
##  Median :11200   Median : 1160   Median :  866   Median : 328.0
##  Mean   :11849   Mean   : 1970   Mean   : 1309   Mean   : 452.2
##  3rd Qu.:13970   3rd Qu.: 2161   3rd Qu.: 1625   3rd Qu.: 528.0
##  Max.   :21700   Max.   :20192   Max.   :13007   Max.   :4615.0
##    top10perc       top25perc      f_undergrad     p_undergrad
##  Min.   : 1.00   Min.   :  9.00   Min.   :  139   Min.   :    1.0
##  1st Qu.:17.00   1st Qu.: 42.00   1st Qu.:  840   1st Qu.:   63.0
##  Median :25.00   Median : 55.00   Median : 1298   Median :  184.0
##  Mean   :29.36   Mean   : 56.99   Mean   : 1867   Mean   :  440.2
##  3rd Qu.:37.00   3rd Qu.: 70.00   3rd Qu.: 2110   3rd Qu.:  541.0
##  Max.   :96.00   Max.   :100.00   Max.   :27378   Max.   :10221.0
##    room_board       books           personal         ph_d
##  Min.   :2370   Min.   : 250.0   Min.   : 300   Min.   :  8.00
##  1st Qu.:3770   1st Qu.: 450.0   1st Qu.: 800   1st Qu.: 60.00
##  Median :4408   Median : 500.0   Median :1100   Median : 73.00
##  Mean   :4612   Mean   : 547.4   Mean   :1211   Mean   : 71.24
##  3rd Qu.:5420   3rd Qu.: 600.0   3rd Qu.:1500   3rd Qu.: 86.00
##  Max.   :8124   Max.   :2340.0   Max.   :6800   Max.   :100.00
##     terminal        s_f_ratio     perc_alumni        expend
##  Min.   : 24.00   Min.   : 2.5   Min.   : 2.00   Min.   : 3186
##  1st Qu.: 69.00   1st Qu.:11.1   1st Qu.:17.00   1st Qu.: 7550
##  Median : 81.00   Median :12.8   Median :25.00   Median : 8991
##  Mean   : 78.86   Mean   :12.9   Mean   :26.12   Mean   :10574
##  3rd Qu.: 92.00   3rd Qu.:14.5   3rd Qu.:34.00   3rd Qu.:11659
##  Max.   :100.00   Max.   :27.8   Max.   :64.00   Max.   :56233
##    grad_rate
##  Min.   : 15.00
##  1st Qu.: 59.00
##  Median : 69.00
##  Mean   : 69.21
##  3rd Qu.: 81.00
##  Max.   :118.00
```

```
skimr::skim(train_df)
```

Table 1: Data summary

Name	train_df

Table 1: Data summary

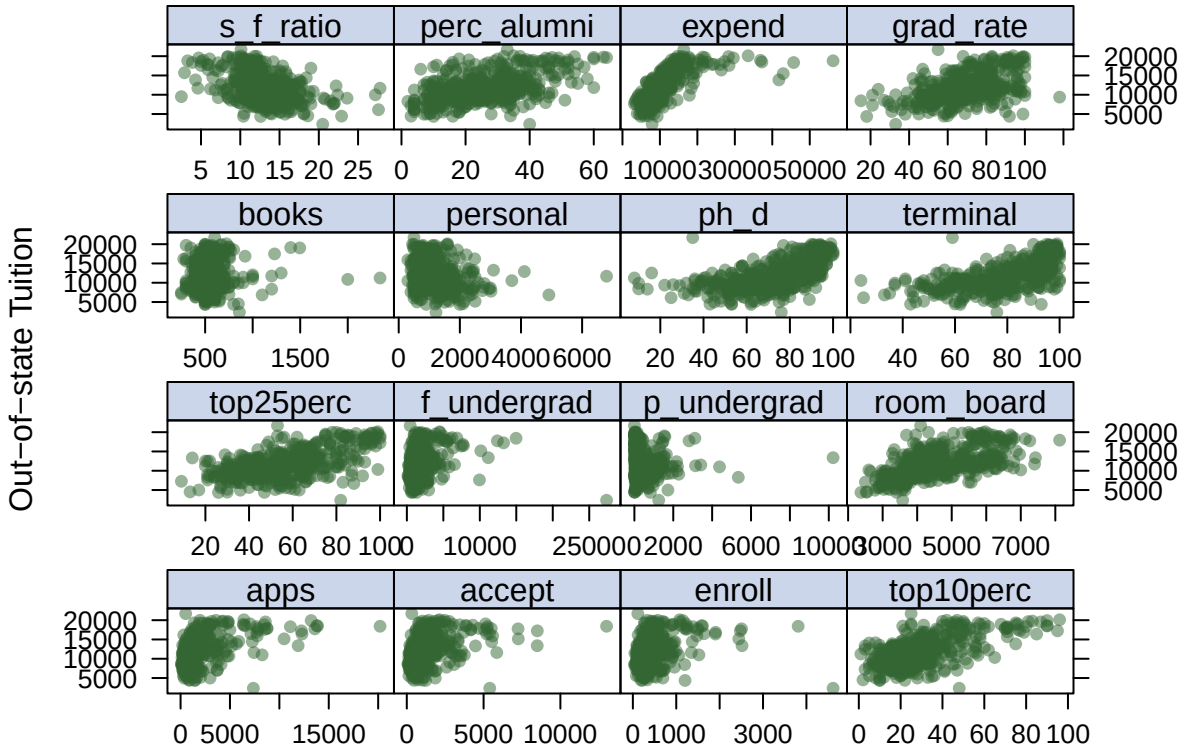
Number of rows	453
Number of columns	17
Column type frequency:	
numeric	17
Group variables	None

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
outstate	0	1	11849.44	3683.26	2340.0	9100.0	11200.0	13970.0	21700.0	
apps	0	1	1970.42	2420.32	81.0	633.0	1160.0	2161.0	20192.0	
accept	0	1	1308.74	1356.91	72.0	513.0	866.0	1625.0	13007.0	
enroll	0	1	452.23	443.32	35.0	209.0	328.0	528.0	4615.0	
top10perc	0	1	29.36	17.70	1.0	17.0	25.0	37.0	96.0	
top25perc	0	1	56.99	19.67	9.0	42.0	55.0	70.0	100.0	
f_undergrad	0	1	1867.29	2122.80	139.0	840.0	1298.0	2110.0	27378.0	
p_undergrad	0	1	440.24	751.47	1.0	63.0	184.0	541.0	10221.0	
room_board	0	1	4612.30	1072.75	2370.0	3770.0	4408.0	5420.0	8124.0	
books	0	1	547.44	184.41	250.0	450.0	500.0	600.0	2340.0	
personal	0	1	1210.98	640.81	300.0	800.0	1100.0	1500.0	6800.0	
ph_d	0	1	71.24	17.73	8.0	60.0	73.0	86.0	100.0	
terminal	0	1	78.86	15.40	24.0	69.0	81.0	92.0	100.0	
s_f_ratio	0	1	12.90	3.37	2.5	11.1	12.8	14.5	27.8	
perc_alumni	0	1	26.12	12.54	2.0	17.0	25.0	34.0	64.0	
expend	0	1	10573.57	5800.59	3186.0	7550.0	8991.0	11659.0	56233.0	
grad_rate	0	1	69.21	16.58	15.0	59.0	69.0	81.0	118.0	

There are 453 rows and 17 columns in training data, all the variables are numeric.

```
# set plot theme
theme1 = trellis.par.get()
theme1$plot.symbol$col = rgb(.2, .4, .2, .5)
theme1$plot.symbol$pch = 16
theme1$plot.line$col = rgb(.8, .1, .1, 1)
theme1$plot.line$lwd = 2
theme1$strip.background$col = rgb(.0, .2, .6, .2)
trellis.par.set(theme1)

# scatter plot
# all predictors are included since they are all continuous
featurePlot(
  x_train,
  y_train,
  plot = "scatter",
  labels = c("","Out-of-state Tuition"),
  layout = c(4, 4))
```



From the scatter plot above, we can see that there might be some linear trends between the outcome variable `outstate` and some of the predictors, for example, `phd` and `terminal`.

Smoothing splines

```
set.seed(2570)

# fit smoothing spline models using terminal as the only predictor of outstate
fit_ss = smooth.spline(x = train_df$terminal, y = train_df$outstate)

# optimal degree of freedom obtained by generalized cross-validation
fit_ss$df
```

```
## [1] 4.392806
```

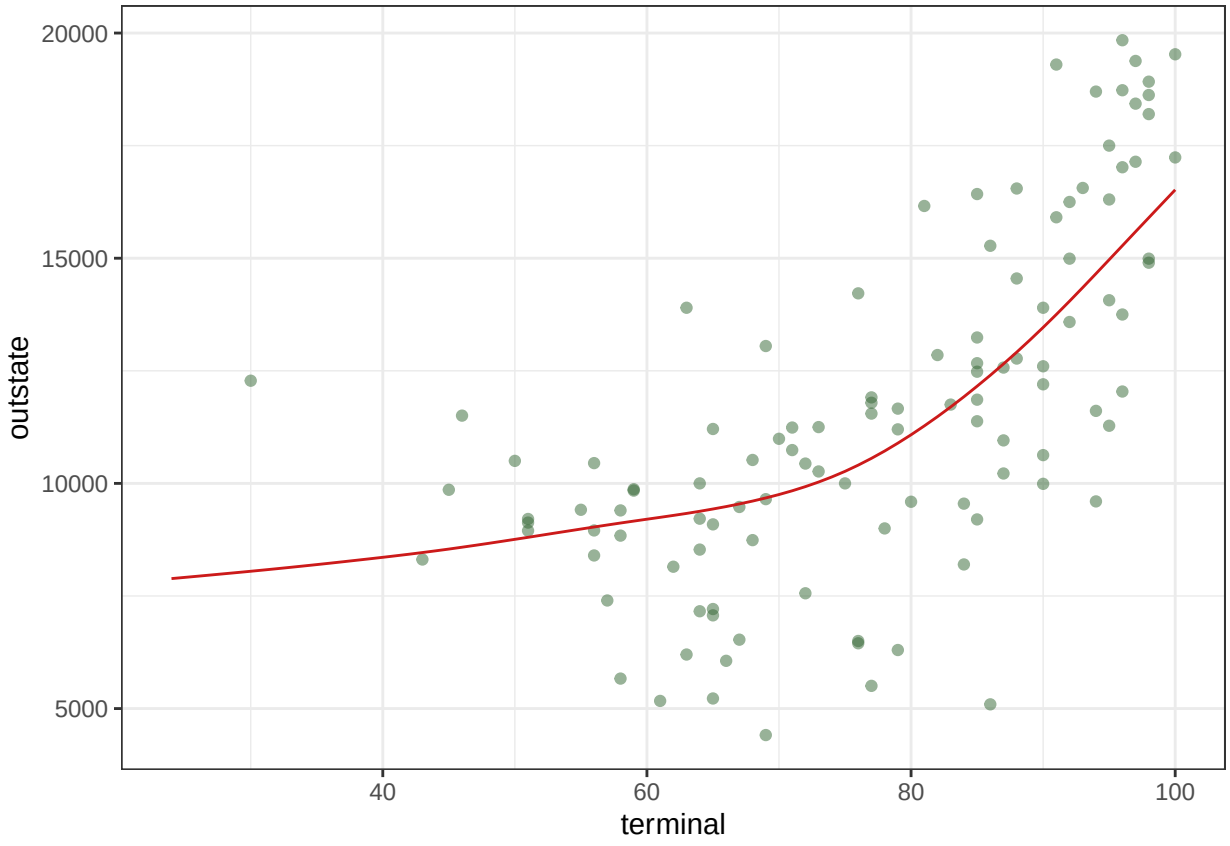
The optimal degree of freedom obtained by default cross validation is 4.393.

Use this **optimal degree of freedom** to make following predictions:

```
# make prediction using a grid of terminal values
# generate predictor grid
range(train_df$terminal)
```

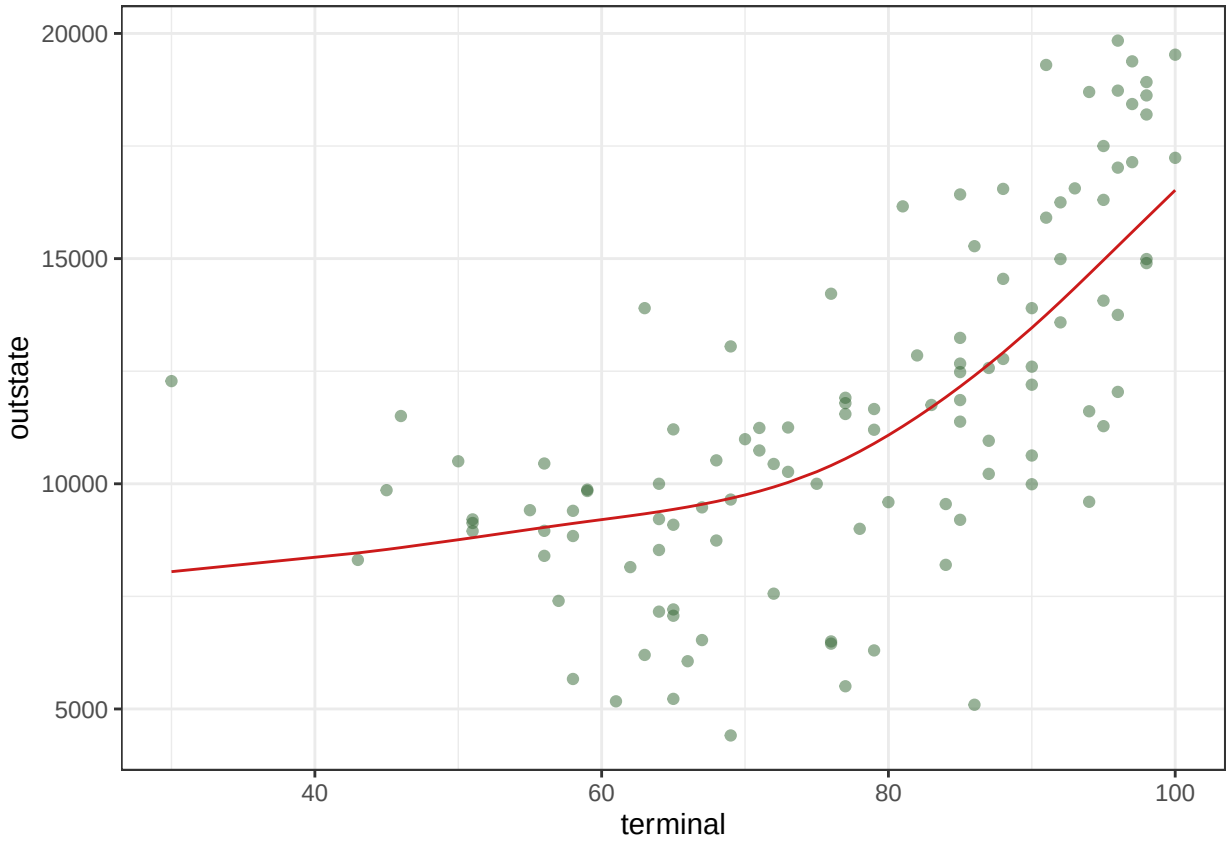
```
## [1]  24 100
```

```r
terminal_grid <- seq(from = 24, to = 100, by = 1)

# make prediction
pred_ss = predict(fit_ss,
                  x = terminal_grid)

pred_ss_df = data.frame(predicted = pred_ss$y,
                        terminal = terminal_grid)

# plot test data
p = ggplot(data = test_df, aes(x = terminal, y = outstate)) +
     geom_point(color = rgb(.2, .4, .2, .5))

# plot predicted value
p +
  geom_line(aes(x = terminal, y = predicted),
            data = pred_ss_df,
            color = rgb(.8, .1, .1, 1)) +
  theme_bw()
```




```r
# make prediction using test data
pred_ss_test = predict(fit_ss,
                       x = test_df$terminal)

pred_ss_test_df = data.frame(predicted = pred_ss_test$y,
```

```
                              terminal = test_df$terminal)

# plot predicted value
p +
  geom_line(aes(x = terminal, y = predicted),
            data = pred_ss_test_df,
            color = rgb(.8, .1, .1, 1)) +
  theme_bw()
```

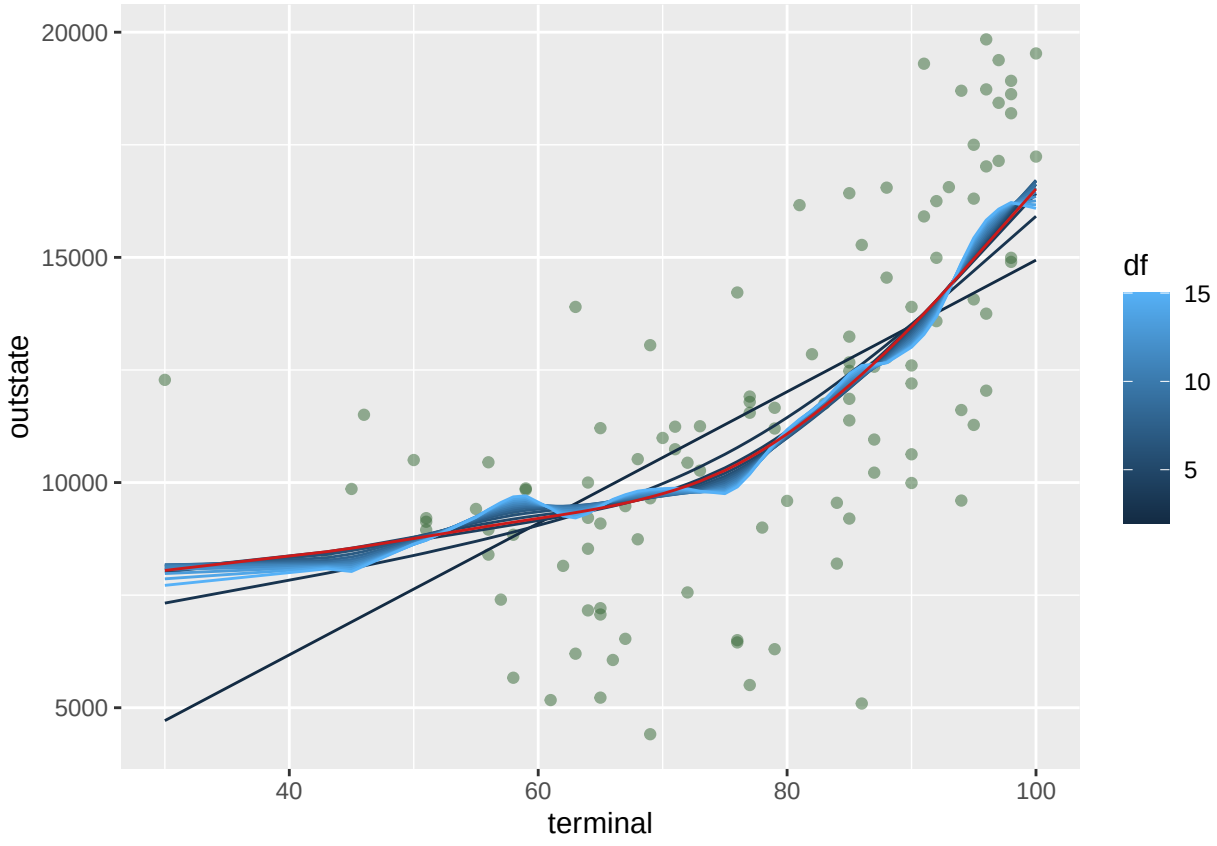


Use **a range of degree of freedom** to make predictions:

```
# write a function using a range of df to fit models
ss_func = function(df) {

  fit_ss_fun = smooth.spline(x = train_df$terminal,
                                    y = train_df$outstate,
                                    df = df)

  pred_ss_fun = predict(fit_ss_fun, x = test_df$terminal)


  pred_ss_df_fun = data.frame(predicted = pred_ss_fun$y,
                           terminal = test_df$terminal,
                           df = df)

}
```

```r
# create a list of df
# 1 < df <= 16 - 1
df_list = seq(2, 15, 1)

# run the function using df_list
output_ss = list()

for (x in df_list) {
  output_ss[[x]] = ss_func(x)
}

# do.call() executes a function by its name and a list of corresponding arguments
# e.g. do.call("any_function", arguments_list)
output_ss_df = do.call("rbind", output_ss) %>%
  as.data.frame()
```

```r
# plot results for a range of df
p +
  geom_line(aes(x = terminal, y = predicted, group = df, color = df), data = output_ss_df) +
  geom_line(aes(x = terminal, y = predicted), data = pred_ss_test_df, color = rgb(.8, .1, .1, 1))
```



The above plot shows the fitted smoothing spline models using a range of degree of freedoms. The lines wiggle around the red line, which is the model using the optimum degree of freedom.
As the degree of freedom approaching to 2, the line gets more linear; as the degree of freedom approaching to 15, the line gets more wiggled.

Among all the fitted lines within the (2, 15) degree of freedom range, df = 4.393 should be the nearest to the red line.

Generalized Additive Model (GAM)

```
set.seed(2570)

# set cross validation method
ctrl = trainControl(method = "cv", number = 10)

# fit a GAM model using all the predictors
# ngcv package not available for current R version, siwth to caret
gam_fit = train(x = x_train,
                y = y_train,
                method = "gam",
                #tuneGrid = data.frame(method = "GCV.Cp",select = c(TRUE, FALSE)),
                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 1.8-39. For overview type 'help("mgcv-package")'.
```

```
## Warning: model fit failed for Fold09: select= TRUE, method=GCV.Cp Error in magic(G$y, G$X, msp, G$S,
##   magic, the gcv/ubre optimizer, failed to converge after 400 iterations.
```

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

```
gam_fit$bestTune
```

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

```
gam_fit$finalModel
```

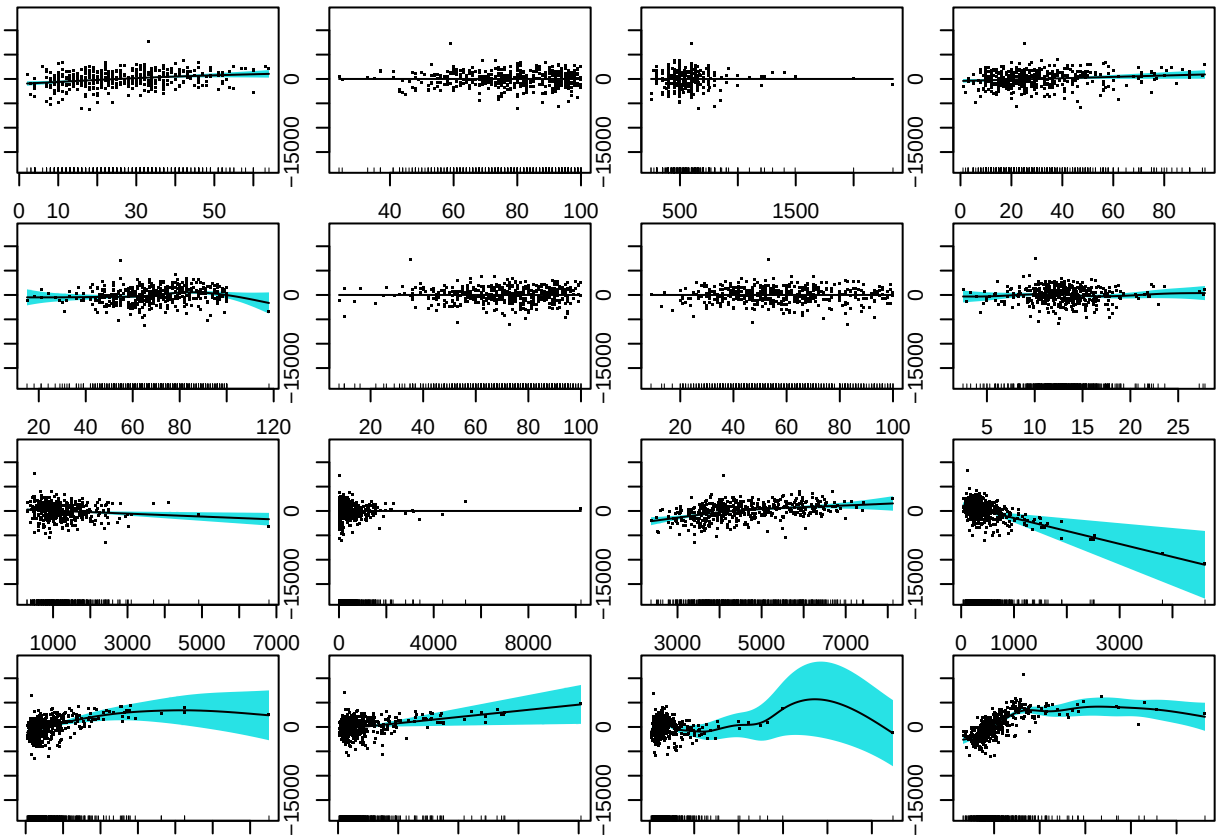
```
##
## Family: gaussian
## Link function: identity
##
```

```
## Formula:
## .outcome ~ s(perc_alumni) + s(terminal) + s(books) + s(top10perc) +
##     s(grad_rate) + s(ph_d) + s(top25perc) + s(s_f_ratio) + s(personal) +
##     s(p_undergrad) + s(room_board) + s(enroll) + s(accept) +
##     s(apps) + s(f_undergrad) + s(expend)
##
## Estimated degrees of freedom:
## 1.343 0.000 0.000 0.765 4.300 0.000 0.000
## 3.303 0.792 0.000 2.470 1.000 2.209 0.784
## 6.383 5.777  total = 30.12
##
## GCV score: 2861107
```

```
summary(gam_fit)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## .outcome ~ s(perc_alumni) + s(terminal) + s(books) + s(top10perc) +
##     s(grad_rate) + s(ph_d) + s(top25perc) + s(s_f_ratio) + s(personal) +
##     s(p_undergrad) + s(room_board) + s(enroll) + s(accept) +
##     s(apps) + s(f_undergrad) + s(expend)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 11849.44      76.78   154.3   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                       edf Ref.df      F  p-value
## s(perc_alumni) 1.342e+00      9  2.385 1.72e-06 ***
## s(terminal)    5.193e-08      9  0.000 0.451441
## s(books)       4.658e-08      9  0.000 0.643128
## s(top10perc)   7.648e-01      9  0.503 0.013797 *
## s(grad_rate)   4.300e+00      9  2.071 0.000478 ***
## s(ph_d)        6.600e-08      9  0.000 0.377735
## s(top25perc)   8.985e-08      9  0.000 0.549356
## s(s_f_ratio)   3.303e+00      9  0.556 0.170799
## s(personal)    7.917e-01      9  0.750 0.003322 **
## s(p_undergrad) 3.459e-08      9  0.000 0.836009
## s(room_board)  2.470e+00      9  5.607  < 2e-16 ***
## s(enroll)      1.000e+00      9  1.128 0.000756 ***
## s(accept)      2.209e+00      9  2.363 1.88e-06 ***
## s(apps)        7.836e-01      9  0.598 0.004737 **
## s(f_undergrad) 6.383e+00      9  2.744 0.000108 ***
## s(expend)      5.777e+00      9 21.533  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.803   Deviance explained = 81.6%
## GCV = 2.8611e+06  Scale est. = 2.6708e+06  n = 453
```

```
# plot the results
par(mar=c(1, 1, 1, 1))
par(mfrow = c(4, 4))

plot(gam_fit$finalModel,
     residuals = TRUE,
     all.terms = TRUE,
     shade = TRUE,
     shade.col = 5)
```




```
# train RMSE of final model
gam_train_rmse = sqrt(mean((y_train - predict(gam_fit)) ^ 2))
gam_train_rmse
```

```
## [1] 1578.995
```

```
# make predictions
gam_pred = predict(gam_fit, x_test)

# test RMSE of final model
gam_test_rmse = sqrt(mean(y_test - gam_pred) ^ 2)
gam_test_rmse
```

```
## [1] 120.7427
```

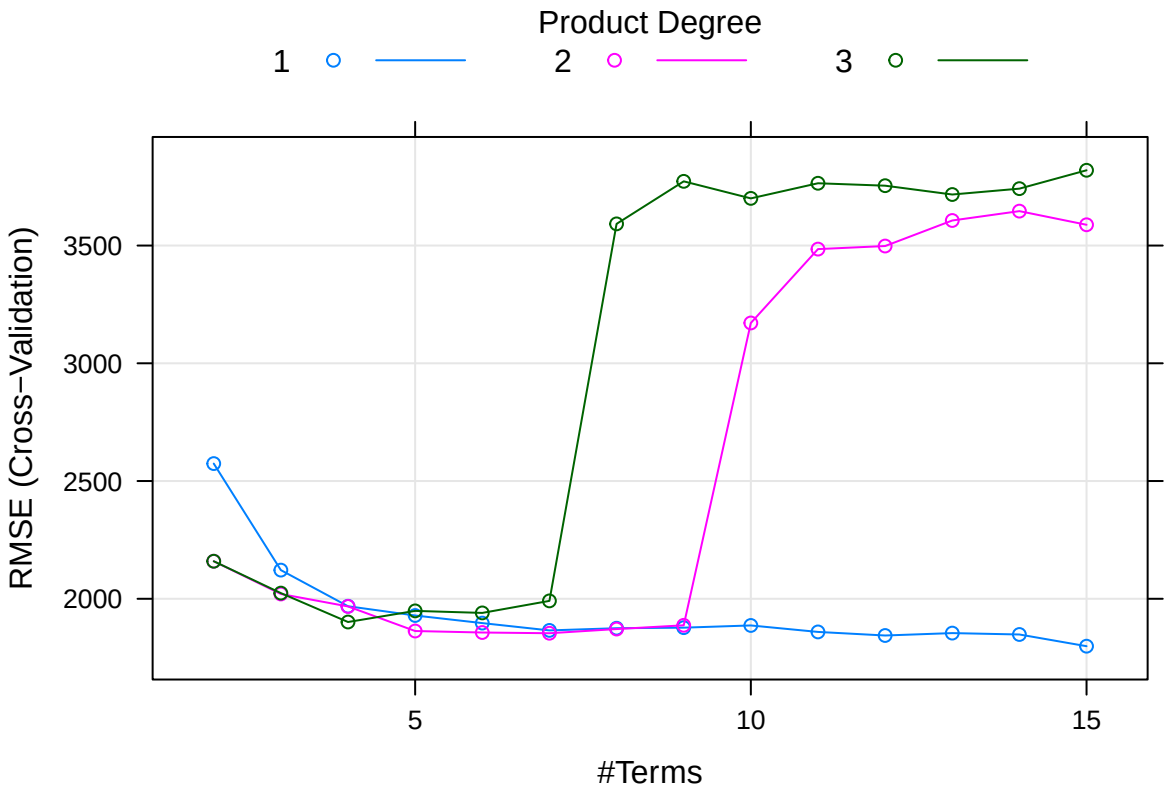
The training RMSE is1578.9953786 and the test RMSE is 120.7426953.
Coefficients are not printed for smooth terms because each smooth term has several coefficients corresponding to different basis functions. The degrees of freedom of each term represent the complexity of the smooth function.
In the final model, `perc_alumni`, `grad_rate`, `room_board`, `enroll`, `accept`, `f_undergrad`, and `expend` are the most significant terms.

Multivariate Adaptive Regression spline (MARS)

```
set.seed(2570)

# generate all possible combinations of degree and prune
mars_grid = expand.grid(degree = 1:3,
                        nprune = 2:15)

# fit MARS model using all predictors
mars_fit = train(x = x_train,
                 y = y_train,
                 method = "earth",
                 tuneGrid = mars_grid,
                 trControl = ctrl)
```

```
## Loading required package: earth

## Loading required package: Formula

## Loading required package: plotmo

## Loading required package: plotrix

## Loading required package: TeachingDemos
```

```
# plot results
plot(mars_fit)
```




```
mars_fit$bestTune
```

```
##    nprune degree
## 14     15      1
```

```
summary(mars_fit$finalModel)
```

```
## Call: earth(x=matrix[453,16], y=c(7440,11250,12...), keepxy=TRUE, degree=1,
##             nprune=15)
##
##                       coefficients
## (Intercept)             8916.1976
## h(apps-3713)               1.0184
## h(apps-7033)              -0.7749
## h(2092-accept)            -1.9571
## h(973-enroll)              4.8311
## h(1362-f_undergrad)       -1.5040
## h(f_undergrad-1362)       -0.3392
## h(4310-room_board)        -1.4142
## h(1300-personal)           0.7393
## h(14-perc_alumni)       -124.3679
## h(perc_alumni-14)         24.9379
## h(expend-6880)             0.7067
## h(expend-15687)           -0.7435
## h(grad_rate-64)           62.5246
```

```
## h(grad_rate-86)        -137.8786
##
## Selected 15 of 28 terms, and 9 of 16 predictors (nprune=15)
## Termination condition: Reached nk 33
## Importance: expend, room_board, grad_rate, perc_alumni, accept, ...
## Number of terms at each degree of interaction: 1 14 (additive model)
## GCV 2855581    RSS 1133255805    GRSq 0.7899753    RSq 0.8151901
```

```
coef(mars_fit$finalModel)
```

```
##         (Intercept)     h(expend-15687)     h(grad_rate-86)  h(4310-room_board)
##        8916.1975693          -0.7435281        -137.8785588         -1.4141701
## h(f_undergrad-1362) h(1362-f_undergrad)   h(perc_alumni-14)   h(14-perc_alumni)
##          -0.3391568          -1.5040457          24.9378899        -124.3679155
##        h(apps-7033)       h(973-enroll)    h(1300-personal)     h(grad_rate-64)
##          -0.7749292           4.8311314           0.7393380          62.5246315
##      h(2092-accept)      h(expend-6880)        h(apps-3713)
##          -1.9571447           0.7067262           1.0183684
```

```
# train RMSE of final model
mars_train_rmse = sqrt(mean((y_train - predict(mars_fit)) ^ 2))
mars_train_rmse
```

```
## [1] 1581.666
```

```
# make predictions
mars_pred = predict(mars_fit, x_test)

# test RMSE of final model
mars_test_rmse = sqrt(mean(y_test - gam_pred) ^ 2)
mars_test_rmse
```

```
## [1] 120.7427
```

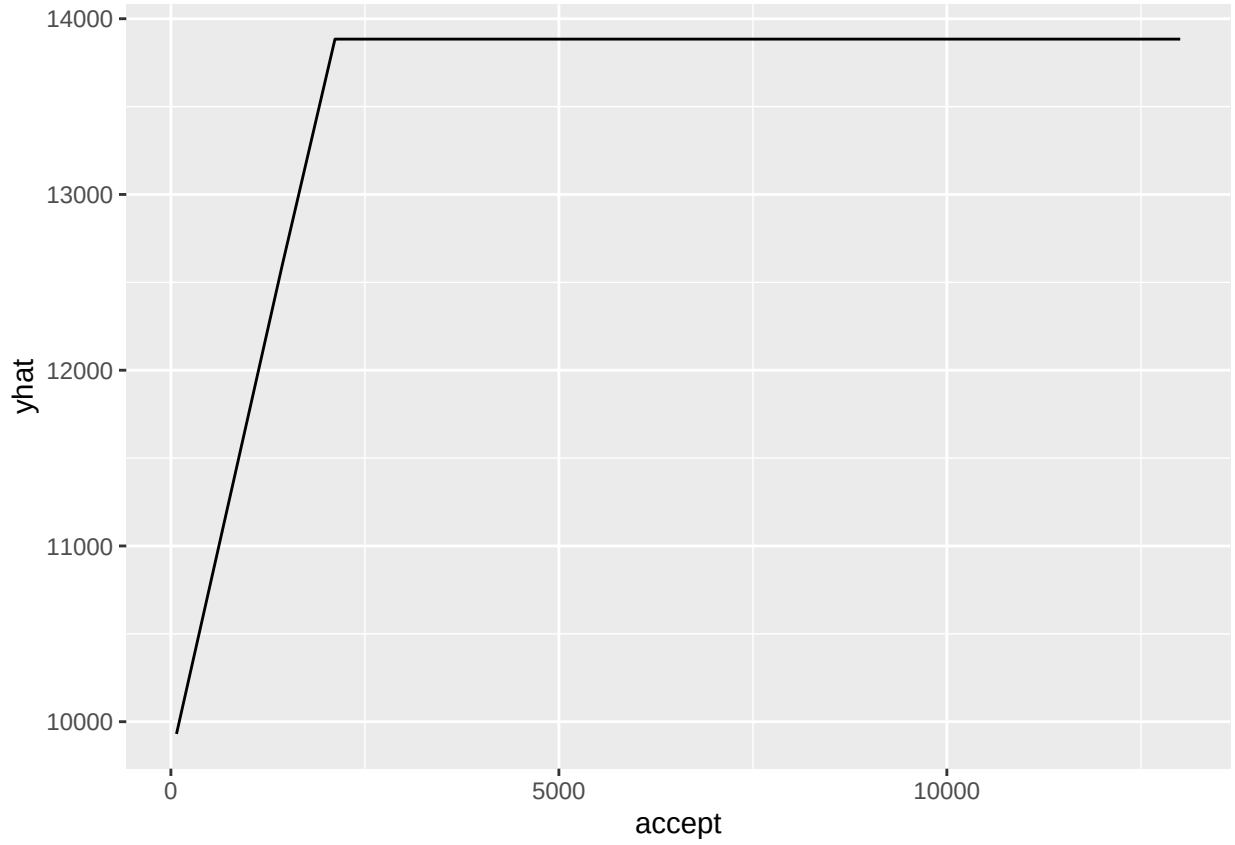
The training RMSE is1581.6663504 and the test RMSE is 120.7426953.
The final model's maximum degree of interactions is 1, which means the final model is an additive model.
`nprune` is 13, which means there are 13 terms in the final model, including intercept.
The most important terms in the final model are `expend`, `room_board`, `perc_alumni`, `accept`, and `enroll`.

To better understand the relationship between these features and response variable, we can create partial dependence plots (PDPs) for each feature individually, and also an interaction PDP. This is used to examine the marginal effects of predictors.

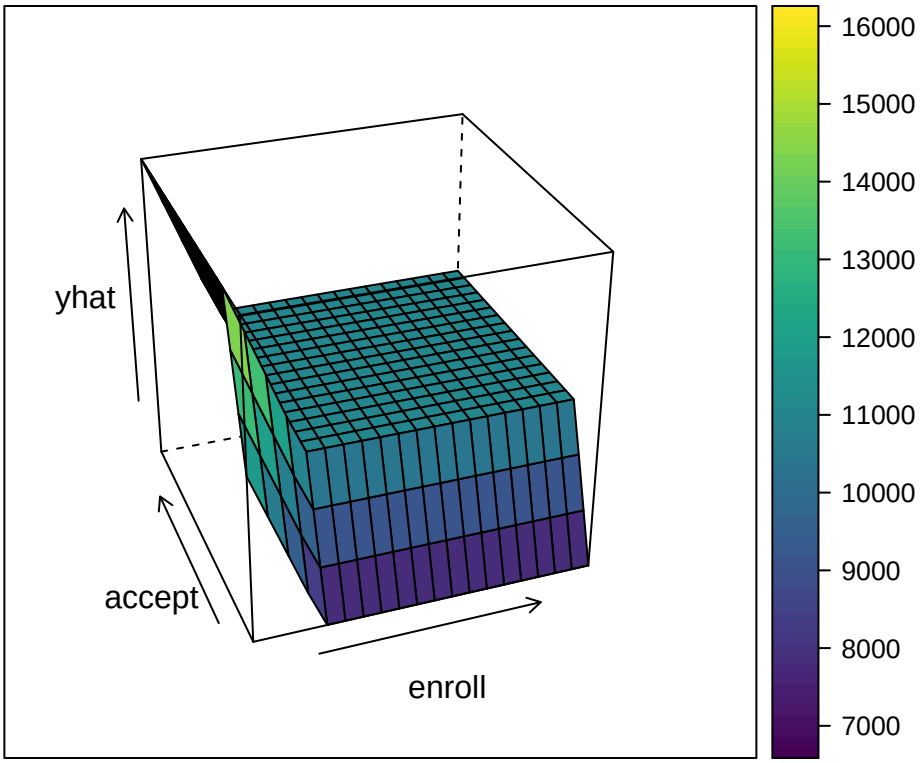
```
# partial dependence plot
# use `books` as predictor
pdp = pdp::partial(mars_fit,
                   pred.var = c("accept"),
                   grid.resolution = 20) %>%
  autoplot()
pdp
```

```
## Warning: Use of `object[[1L]]` is discouraged. Use `.data[[1L]]` instead.
```

```
## Warning: Use of `object[["yhat"]]` is discouraged. Use `.data[["yhat"]]`
## instead.
```




```
# use `books` and `expend` as predictors
pdp_2d = pdp::partial(mars_fit,
                      pred.var = c("enroll", "accept"),
                      grid.resolution = 20) %>%
  pdp::plotPartial(levelplot = FALSE,
                   zlab = "yhat",
                   drape = TRUE,
                   screen = list(z = 20, x = -60))
pdp_2d
```




```
# grid.arrange(pdp, pdp_2d, n_col = 2)
```

Within the range of approximately less than 2500, we can see a trend of decrease on the response variable when as `accpet` increases, and the value of response variable stays stable afterwards. This turning point is the knot.

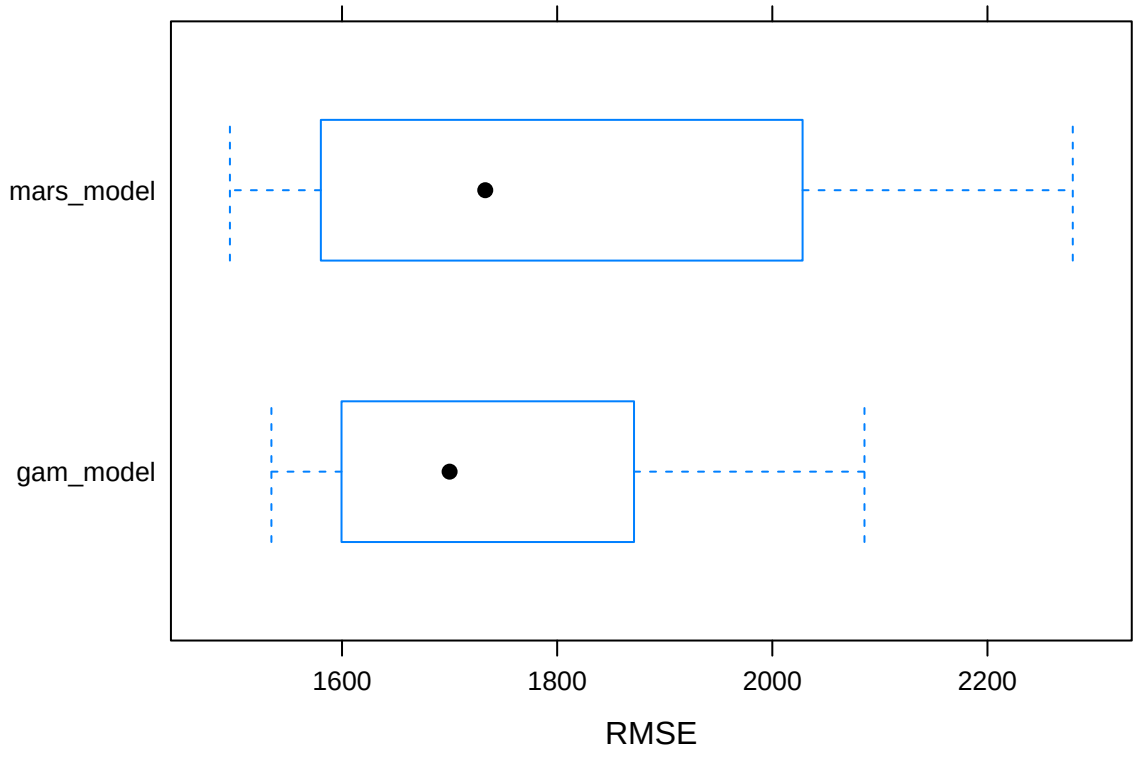
Model selection

```
resamp = resamples(list(gam_model = gam_fit,
                        mars_model = mars_fit))

summary(resamp)
```

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: gam_model, mars_model
## Number of resamples: 10
##
## MAE
##                Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## gam_model  1281.682 1321.201 1354.338 1395.333 1478.269 1605.173    1
```

```
## mars_model 1190.517 1265.526 1397.423 1406.505 1507.640 1693.124    0
##
## RMSE
##                Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## gam_model  1534.352 1599.533 1699.985 1754.383 1871.438 2085.668    1
## mars_model 1495.786 1585.187 1733.147 1798.590 2009.823 2279.285    0
##
## Rsquared
##                 Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## gam_model  0.6782564 0.7543747 0.7878427 0.7706326 0.7988537 0.8551056    1
## mars_model 0.6299040 0.7137411 0.7896711 0.7612189 0.8135660 0.8657178    0
```

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



In this data example, we might prefer the use of MARS model over linear model when predicting the out-of-state tuition, since the RMSE of MARS model is smaller, which indicates the MARS model fits the data better.