P8106 Homework 2

Mindy Tran mnt2130

3/5/2023

Contents

This reads the CSV file, cleans the variables names and eliminates any missing data.

```
set.seed(2132)
college_data = read_csv("./Data/College.csv") %>%
  janitor::clean_names() %>%
  na.omit() %>%
  relocate("outstate", .after = "grad_rate") %>%
  select(-college)
```

This next step will partition the data into training (80% of data) and test (20%) data sets and create matrices of the training and testing data frame for further analysis.

```
indexTrain = createDataPartition(y = college_data$outstate,
                                 p = 0.8,
                                 list = FALSE)
training_df = college_data[indexTrain, ]
testing_df = college_data[-indexTrain, ]

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

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

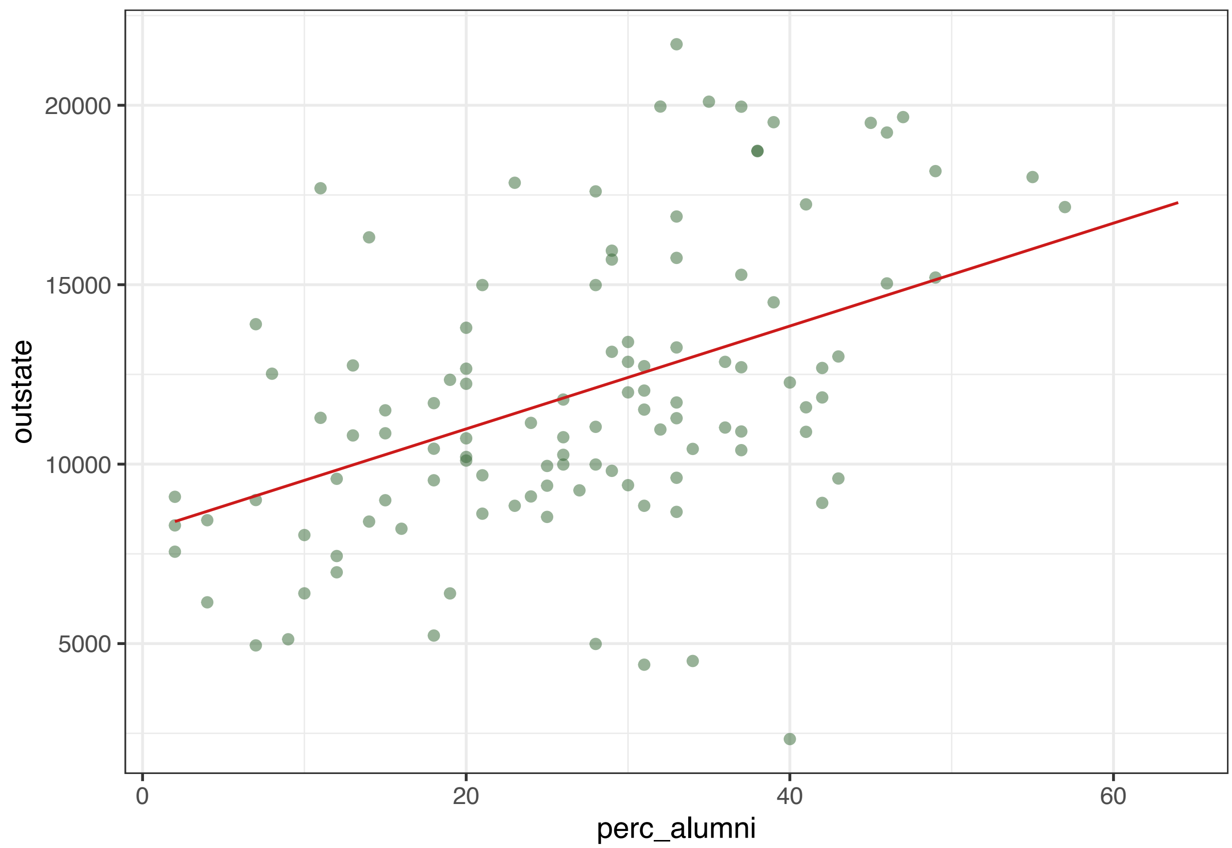
Part A: Smoothing Spline Model

Fit smoothing spline models using `perc_alumni` as the only predictor of `Outstate` for a range of degrees of freedom.

```r
# This code fits a smoothing spline model for perc_alumni as a predictor of outstate
fit_ss = smooth.spline(training_df$perc_alumni, training_df$outstate)

# This outputs the optimal degrees of freedom from the cross-validation
fit_ss$df
```

```
## [1] 2.00024
```

```r
alumni_grid = seq(from = 2, to = 64, by = 1)
pred_ss_grid = predict(fit_ss, x = alumni_grid)
pred_ss_grid_df = data.frame(predicted = pred_ss_grid$y,
                             perc_alumni = alumni_grid)
p = ggplot(data = testing_df, aes(x = perc_alumni, y = outstate)) +
     geom_point(color = rgb(.2, .4, .2, .5))

p_line = p + geom_line(aes(x = perc_alumni, y = predicted), data = pred_ss_grid_df,
              color = rgb(.8, .1, .1, 1)) + theme_bw()

p_line
```



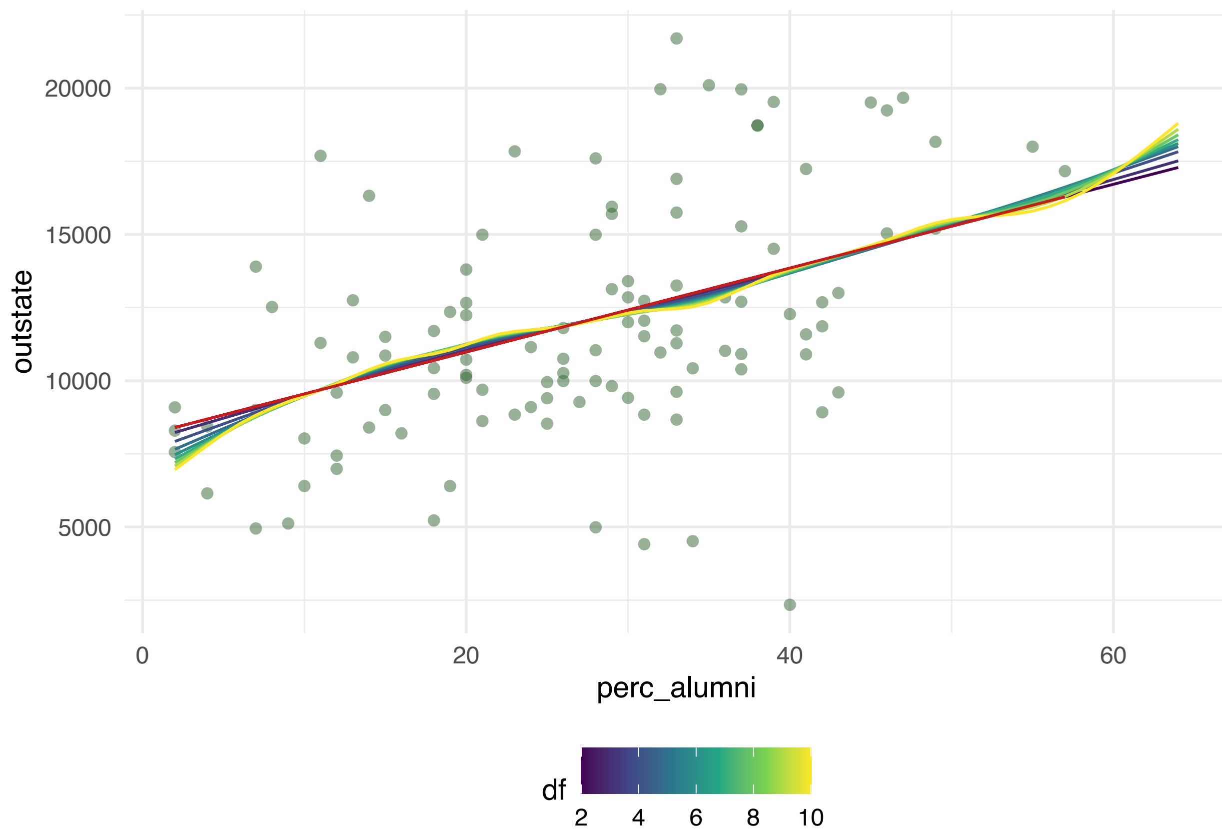

```r
# Now we can use it on the test data

pred_ss_testing = predict(fit_ss, x = testing_df$perc_alumni)
pred_ss_testing_df = data.frame(predicted = pred_ss_testing$y,
```

```
                                               perc_alumni = testing_df$perc_alumni)
predicted_plot = p + geom_line(aes(x = perc_alumni, y = predicted), data = pred_ss_testing_df,
          color = rgb(.8, .1, .1, 1)) + theme_bw()
```

This is the smoothing spline model using `perc_alumni` as the sole predictor for `outstate` using the optimal degrees of freedom (df=2.00024) outputted from the generalized cross validation.

Now we will fit a smoothing spline model for a range of degrees of freedom:

```
# Smoothing Spline Model with Range of DFs
spline_range = function(degree){

  spline_fit = smooth.spline(training_df$perc_alumni, training_df$outstate, df = degree)

  spline_pred = predict(spline_fit, x = alumni_grid)

  spline_df = data.frame(predicted = spline_pred$y,
                         perc_alumni = alumni_grid,
                         df = degree)
}

# Now we can run our spline function for DF values 2 through 10
datalist = list()
for (i in 2:10) {
  datalist[[i]] = spline_range(i)
}
all_data = do.call(rbind, datalist) %>%
  as.data.frame()

# Plot for range of degree of freedom where red line represents optimal DF

plot_range = p +
  geom_line(aes(x = perc_alumni, y = predicted, group = df, color = df), data = all_data) +
  geom_line(aes(x = perc_alumni, y = predicted), data = pred_ss_testing_df,
          color = rgb(.8, .1, .1, 1))

plot_range
```



When we overlay the models generated from different degree of freedoms, the model fittings are generally clustered and we see that with around 2-3 degree of freedoms, our model appears to fit more linear as represented by the purple lines but the lines (blue, green, yellow) starts to wiggle just a tiny bit when generated with higher degree of freedoms, suggesting slight potential over-fitting but not too much.

Part B: Generalized Additive Model

```
set.seed(2132)
ctrl1 = trainControl(method = "cv", number = 10)

sapply(x_train %>% as.data.frame(), n_distinct)
```

```
##        apps      accept      enroll   top10perc   top25perc f_undergrad
##         424         414         344          74          83         415
## p_undergrad  room_board       books    personal        ph_d    terminal
##         334         347          70         176          76          65
##   s_f_ratio perc_alumni      expend   grad_rate
##         131          57         444          75
```

```
#None of the predictors take on less than 10 values, thus we can proceed using the caret package, since

# We can now run GAM using the caret package and use automatic feature selection
gam = train(x_train, y_train,
```

```r
                method = "gam",
                tuneGrid = data.frame(method = "GCV.Cp",
                                      select = c(TRUE, FALSE)),
                trControl = ctrl1)

# This outputs the parameters which fit the best model
gam$bestTune
```

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

```r
gam$finalModel
```

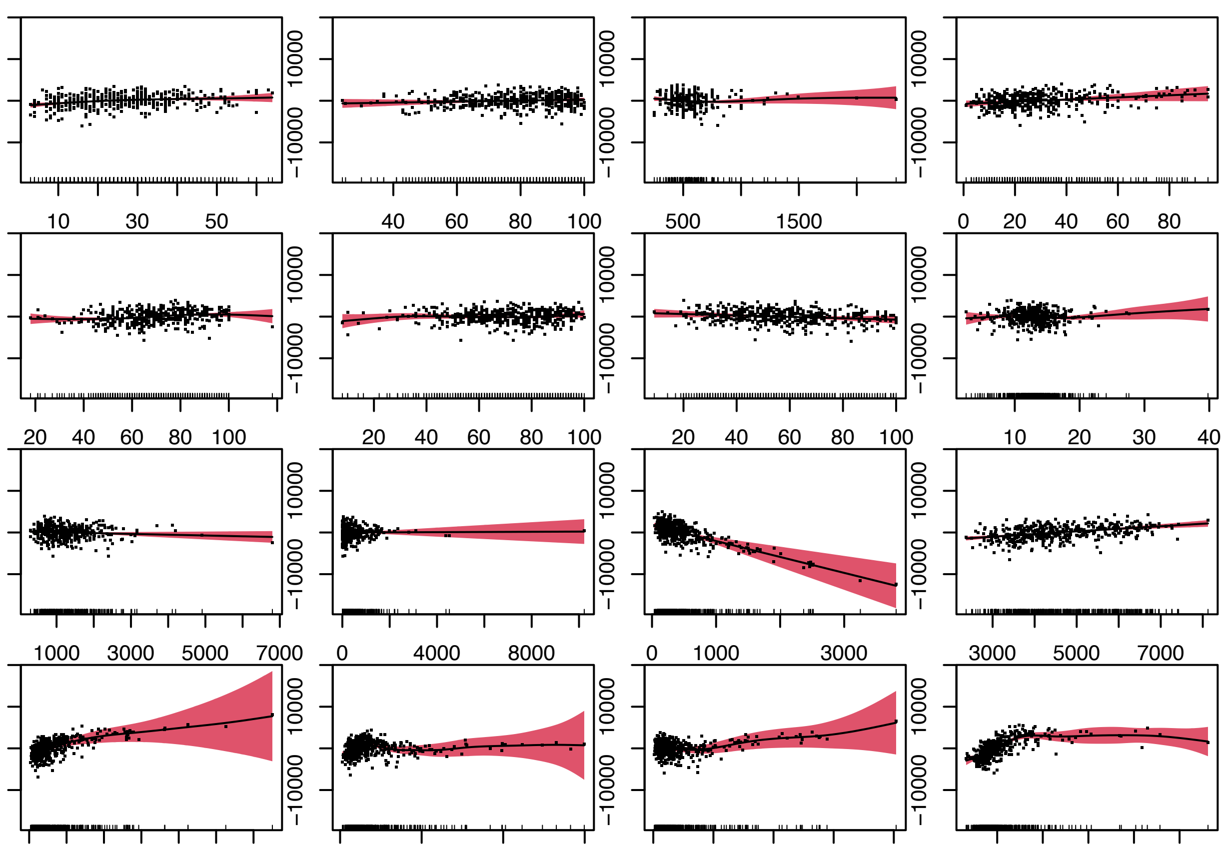
```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## .outcome ~ s(perc_alumni) + s(terminal) + s(books) + s(top10perc) +
##     s(grad_rate) + s(ph_d) + s(top25perc) + s(s_f_ratio) + s(personal) +
##     s(p_undergrad) + s(enroll) + s(room_board) + s(accept) +
##     s(f_undergrad) + s(apps) + s(expend)
##
## Estimated degrees of freedom:
## 2.24 1.00 2.73 1.00 3.16 3.70 1.00
## 3.68 1.00 1.00 1.00 1.15 3.18 6.24
## 4.21 5.70  total = 42.99
##
## GCV score: 2748289
```

```r
# This outputs the final model with the effective degree of freedoms
summary(gam)
```

```
##
## 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(enroll) + s(room_board) + s(accept) +
##     s(f_undergrad) + s(apps) + s(expend)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  11787.8       74.1   159.1   <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) 2.237  2.838  6.626 0.000514 ***
```

```
## s(terminal)    1.000  1.000  1.434 0.231744
## s(books)       2.731  3.403  1.949 0.125752
## s(top10perc)   1.000  1.000  3.502 0.061993 .
## s(grad_rate)   3.161  4.008  4.261 0.002163 **
## s(ph_d)        3.700  4.612  1.091 0.355130
## s(top25perc)   1.000  1.000  2.417 0.120794
## s(s_f_ratio)   3.684  4.622  1.563 0.215053
## s(personal)    1.000  1.000  2.307 0.129571
## s(p_undergrad) 1.000  1.000  0.023 0.880120
## s(enroll)      1.000  1.000 22.679 2.95e-06 ***
## s(room_board)  1.151  1.284 30.300  < 2e-16 ***
## s(accept)      3.179  4.002  3.867 0.004259 **
## s(f_undergrad) 6.239  7.304  4.559 5.21e-05 ***
## s(apps)        4.213  5.184  2.040 0.075568 .
## s(expend)      5.697  6.831 17.270  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.811   Deviance explained = 82.8%
## GCV = 2.7483e+06  Scale est. = 2.4875e+06  n = 453
```

```r
# Plot of the GAM Model

par(mar = c(1,1,1,1))
par(mfrow = c(4, 4))
plot(gam$finalModel, residuals = TRUE, all.terms = TRUE, shade = TRUE, shade.col = 2)
```



```r
# Now we calculate the training and test MSE and RMSE of the optimized model

#training MSE
gam_train_MSE = mean((y_train - predict(gam))^2)
gam_train_MSE
```

```
## [1] 2251375
```

```r
gam_train_RMSE = sqrt(gam_train_MSE)
gam_train_RMSE
```

```
## [1] 1500.458
```

```r
#test MSE

test_pred = predict(gam, x_test)

gam_test_MSE = mean((y_test - test_pred)^2)
gam_test_MSE
```

```
## [1] 3364712
```

```r
gam_test_RMSE = sqrt(gam_test_MSE)
gam_test_RMSE
```

```
## [1] 1834.315
```

The optimal GAM model does include all predictors: 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(enroll) + s(room_board) + s(accept) + s(f_undergrad) + s(apps) + s(expend).

6 variables: terminal, top10perc, top25perc, personal, p_undergrad, enroll, all have 1 degree of freedom, which corresponds by their straight line and confirmed as such in our plots. Variables with 2 degrees of freedom are incorporated as quadratics, while variables with 3 DFs are incorporated as cubic functions. perc_alumni, enroll, room_board, f_undergrad, and expend are the most significant smooth terms.

Generating a model using all predictors, the optimized model when used on the *training data* obtains MSE = 2251375 with RMSE= 1500.458 and when used on the *test data* obtains MSE= 3364712 and an RMSE= 1834.315.

Part C: Multivariate Adaptive Regression Spline Model

Now we'll train a MARS model with all predictors from the college dataset.

```r
set.seed(2132)
ctrl1 = trainControl(method = "cv", number = 10)
# Generate grid of tuning parameters
mars_grid = expand.grid(degree = 1:3,
                        nprune = 2:25)
# Fit the MARS model
```

```r
mars = train(x_train, y_train,
                  method = "earth",
                  tuneGrid = mars_grid,
                  trControl = ctrl1)
mars$bestTune
```

```
##    nprune degree
## 17     18      1
```

```r
#To minimize RMSE, we choose the model with 1 degree of freedom and 18 hinge functions

summary(mars$finalModel)
```

```
## Call: earth(x=matrix[453,16], y=c(12280,11250,1...), keepxy=TRUE, degree=1,
##             nprune=18)
##
##                      coefficients
## (Intercept)             7538.5533
## h(apps-3767)               0.3936
## h(2109-accept)            -1.5336
## h(accept-2109)             0.4331
## h(913-enroll)              5.4282
## h(enroll-913)             -2.9772
## h(1379-f_undergrad)       -2.2215
## h(4450-room_board)        -0.7688
## h(room_board-4450)         0.4496
## h(660-books)               2.3720
## h(ph_d-85)                96.5617
## h(21-perc_alumni)        -88.8273
## h(expend-5557)             0.6735
## h(expend-14773)           -0.6865
## h(grad_rate-44)           27.1928
##
## Selected 15 of 22 terms, and 10 of 16 predictors (nprune=18)
## Termination condition: RSq changed by less than 0.001 at 22 terms
## Importance: expend, grad_rate, accept, enroll, f_undergrad, room_board, ...
## Number of terms at each degree of interaction: 1 14 (additive model)
## GCV 2763911    RSS 1096876249    GRSq 0.7902001    RSq 0.8153879
```

```r
# We will use 10 of the 16 predictors

coef(mars$finalModel)
```

```
##         (Intercept)     h(expend-14773)     h(grad_rate-44)  h(room_board-4450)
##        7538.5533341          -0.6864909          27.1927519           0.4495547
##  h(4450-room_board) h(1379-f_undergrad)   h(21-perc_alumni)        h(apps-3767)
##          -0.7687973          -2.2215037         -88.8273388           0.3935794
##       h(enroll-913)       h(913-enroll)      h(accept-2109)      h(2109-accept)
##          -2.9772095           5.4282064           0.4330915          -1.5335817
##      h(expend-5557)          h(ph_d-85)        h(660-books)
##           0.6734655          96.5617402           2.3720382
```

```r
# Training MSE and RMSE
mars_train_MSE = mean((y_train - predict(mars))^2)
mars_train_MSE
```

```
## [1] 2421360
```

```r
mars_train_RMSE = sqrt(mars_train_MSE)
mars_train_RMSE
```

```
## [1] 1556.072
```

```r
# Test MSE and RMSE
test_pred_mars = predict(mars, x_test)

mars_test_MSE = mean((y_test - test_pred_mars)^2)
mars_test_MSE
```

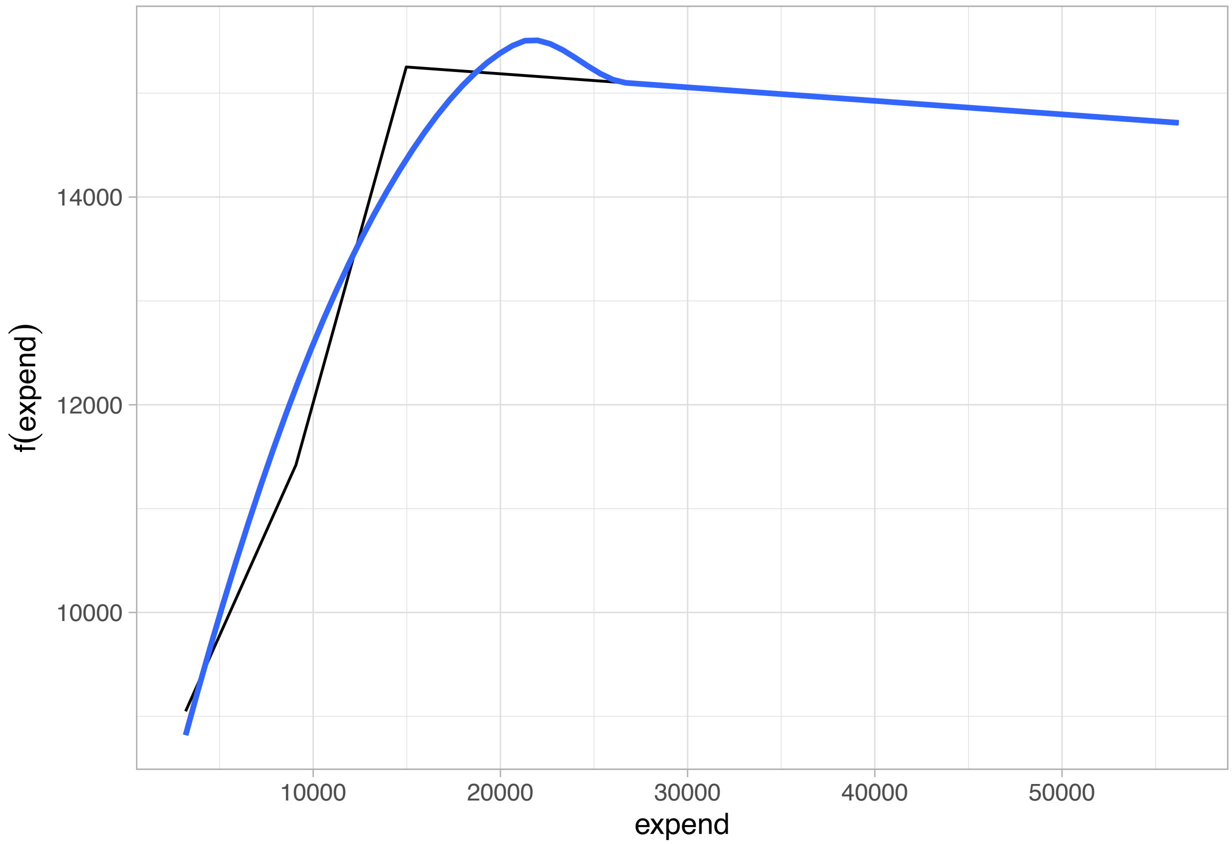
```
## [1] 3460709
```

```r
mars_test_RMSE = sqrt(mars_test_MSE)
mars_test_RMSE
```

```
## [1] 1860.298
```

Now that we have trained a MARS model using all predictors, the optimal model achieves MSE= 2421360 (RMSE=1556.1) when the model is applied to the training data and an MSE= 3460709 (RMSE =1860.3) applied to the partitioned test data. The final model minimizes RMSEby using one product degree (maximum degree of interactions) and 18 maximum terms, including intercept. 15 of 22 terms were used from 10 of the 16 original predictors. The 15 terms used include hinge functions and intercept. The most important predictors for outstate appear to be expend, grad_rate, accept, and enroll.

```r
# Present partial dependence plot of arbitrary predictor in final model
partial_pred = pdp::partial(mars, pred.var = c("expend"),
                  grid.resolution = 10) %>%
  autoplot(smooth = TRUE, ylab = expression(f(expend))) +
  theme_light()

partial_pred
```



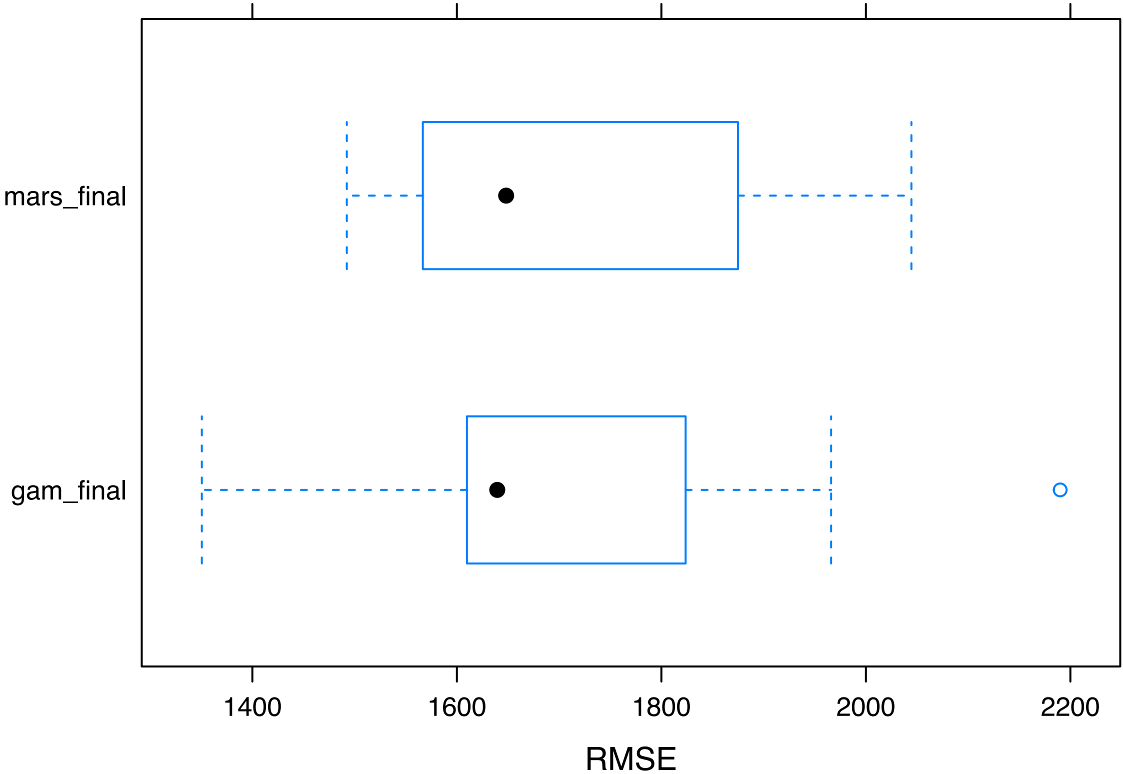
Here is the partial dependence plot: for the predictor expend. For this predictor, we observe a single internal knot located at 14773, which mirrors that reported in the MARS model summary generated previously. This means that as a college goes above the value 14773 on the expend metric, every additional unit of expend experiences decrease in outstate in comparison to that of colleges with less than 14773 in expend.

Part D: Selecting a Model

```
resamp = resamples(list(gam_final = gam,
                        mars_final = mars))
summary(resamp)
bwplot(resamp, metric = "RMSE")
```

For this case, we would prefer the GAM model over the MARS model for predicting out of state tuition since the GAM model is slightly more effective at minimizing the RMSE, as shown in the plot above.

I think for general approach, the MARS model is more flexible than a linear model as it can capture nonlinear relationships between the input variables and the response variable. On the other hand, linear models are simpler and more interpretable than MARS, and they may be more appropriate when the relationships between input and output variables are linear or nearly linear. It depends on the data being used and what we are trying to do with the data.



mars_final
gam_final
1400
1600
1800
2000
2200
RMSE