P8106_HW2

Naomi Simon-Kumar ns3782

02/03/2025

Contents

Loading libraries

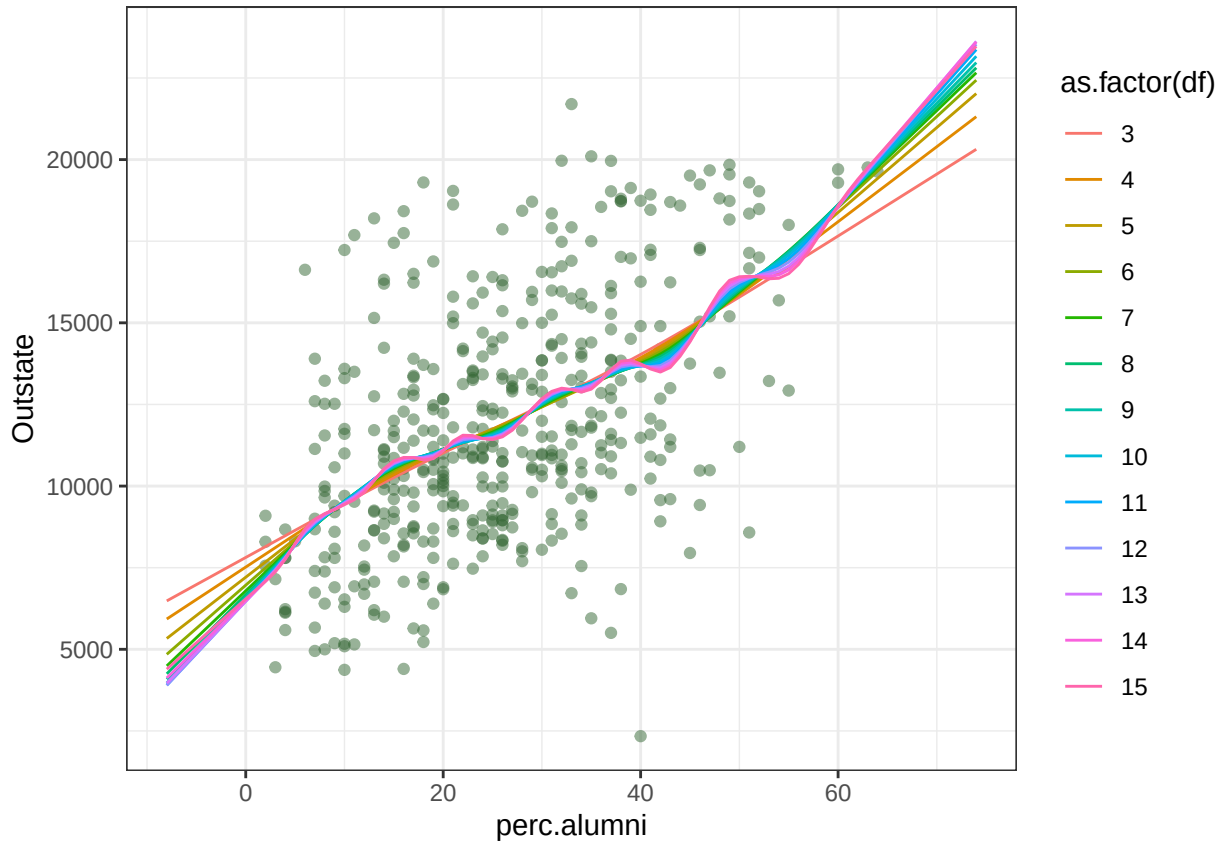
```
# Load libraries
library(earth)
library(tidyverse)
library(ISLR)
library(pls)
library(caret)
library(tidymodels)
```

Partition into training and testing set

```
# Read in dataset
college <- read.csv("College.csv")

# Remove NAs
college <- na.omit(college)

# Set seed for reproducibility
set.seed(299)

# Split data into training and testing data
data_split <- initial_split(college, prop = 0.8)


# Extract the training and test data
training_data <- training(data_split)
testing_data <- testing(data_split)
```

Question (a): Smoothing spline models

```r
# Set seed for reproducibility
set.seed(299)

# Fit smoothing spline
fit.ss <- smooth.spline(training_data$perc.alumni, training_data$Outstate)

# Define a grid for smooth predictions using dataset range
# Will illustrate curves beyond dataset boundary
perc.alumni.grid <- seq(from = min(training_data$perc.alumni) - 10,
                        to = max(training_data$perc.alumni) + 10,
                        by = 1)

# Create dataframe of degrees of freedom range
df_values <- seq(3, 15, by = 1)

# Create dataframe to store smoothing spline predictions
ss.predictions <- data.frame()

# Use for loop to populate ss.predictions dataframe
# Code Source: https://www.rdocumentation.org/packages/openintro/versions/2.4.0/topics/loop

for (df in df_values) {

  # Plot smoothing spline curves for different degrees of freedom
  # Code Source: https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/smooth.spline

  fit <- smooth.spline(training_data$perc.alumni, training_data$Outstate, df = df)
  pred <- predict(fit, x = perc.alumni.grid)

  # Store pred for current df
  temp_df <- data.frame(
    perc.alumni = perc.alumni.grid,
    pred = pred$y,
    df = df
  )

  ss.predictions <- rbind(ss.predictions, temp_df) # Add to ss.predictions
}

# Scatter plot of perc.alumni vs Outstate
ss.p <- ggplot(training_data, aes(x = perc.alumni, y = Outstate)) +
  geom_point(color = rgb(.2, .4, .2, .5))

# Add smoothing spline curves for df range 3-15
ss.p +
  geom_line(aes(x = perc.alumni, y = pred, color = as.factor(df)),
          data = ss.predictions) + theme_bw()
```



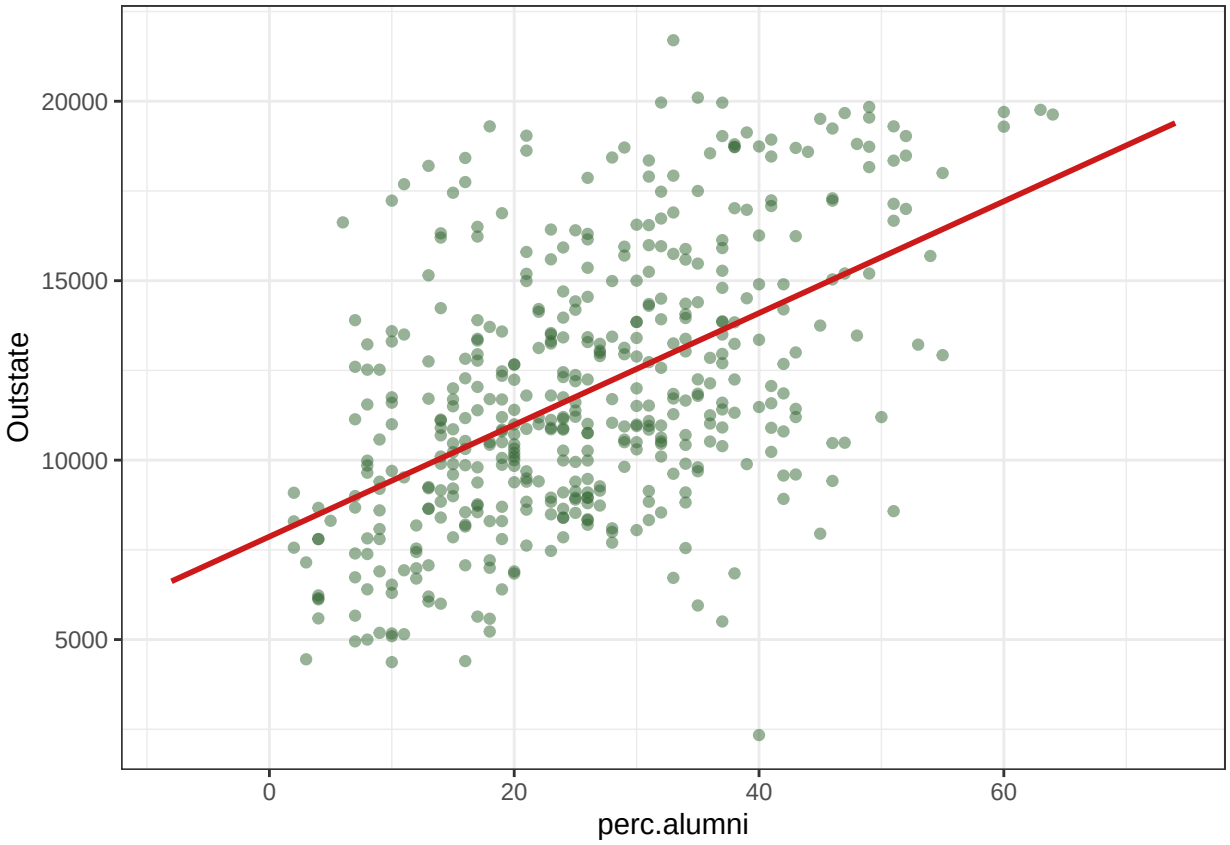
The plot I produced represents the fitted smoothing spline curves for each degree of freedom between the range of 3 and 15. It can be observed that as the degrees of freedom increases over this range, the smoothing spline goes from underfitting, to overfitting of the data. Specifically, as the degrees of freedom increases beyond ~10, the spline curve becomes highly wiggly, particularly at extreme values of perc.alumni, indicating overfitting of the data. For lower df (i.e., 3–5), the spline appears quite smooth.

```
# Set seed for reproducibility
set.seed(299)

# Fits a smoothing spline with automatically selected df using generalized cross-validation
fit.ss.gcv <- smooth.spline(training_data$perc.alumni, training_data$Outstate)

gcv_df <- fit.ss.gcv$df  # Extract optimal df selected by GCV
print(gcv_df) # 2.000237 appears to be the optimal df selected by GCV
```

```
## [1] 2.000237
```

```
# Predict values using GCV-selected df
pred.ss.gcv <- predict(fit.ss.gcv, x = perc.alumni.grid)

# Convert predictions to dataframe
ss.optimal <- data.frame(
  perc.alumni = perc.alumni.grid,
  pred = pred.ss.gcv$y
)
```

```r
# Add optimal spline curve to the plot
ss.p +
  geom_line(aes(x = perc.alumni, y = pred), data = ss.optimal,
  color = rgb(.8, .1, .1, 1), linewidth = 1) + theme_bw()
```




```r
# Checking lambda value

lambda_value <- fit.ss.gcv$lambda
print(lambda_value) # High lambda value 2384.249
```

```
## [1] 2384.249
```

For my smoothing spline model, I selected a degree of freedom of approximately 2, as determined by Generalized Cross-Validation (GCV). The GCV method minimises a criterion balancing model fit against model complexity, and in this case, selected a nearly linear model (df approx. 2.000237). This is evident in the plot above where the optimal smoothing spline appears as essentially a straight line, indicating that the relationship between percentage of alumni who donate and out-of-state tuition is best represented linearly.

While I explored models with higher degrees of freedom (ranging from 3 to 15), these more complex models with greater flexibility showed increasing wiggliness, especially at extreme values of perc.alumni as discussed, indicating potential overfitting. The GCV criterion effectively penalised this unnecessary complexity and favored the simpler model. From what we have covered in class, we know that smoothing splines can range from very flexible (high df) to very smooth, with a minimum of 2 degrees of freedom. As discussed in class, when lambda approaches infinity, the function becomes linear, and this model has a relatively large lambda value (2384.249), explaining why the optimal model with df approximately 2 appears as a straight line. This

relationship between lambda and degrees of freedom demonstrates how the roughness penalty controls the smoothness of the function.

The GCV has selected a model at approximately this minimum value, resulting in a nearly linear fit that would indicate out-of-state tuition (outstate) increases steadily with the percentage of alumni who donate (perc.alumni), without needing to be represented by a more complex nonlinear relationship.

Question (b): Multivariate Adaptive Regression Spline (MARS) model

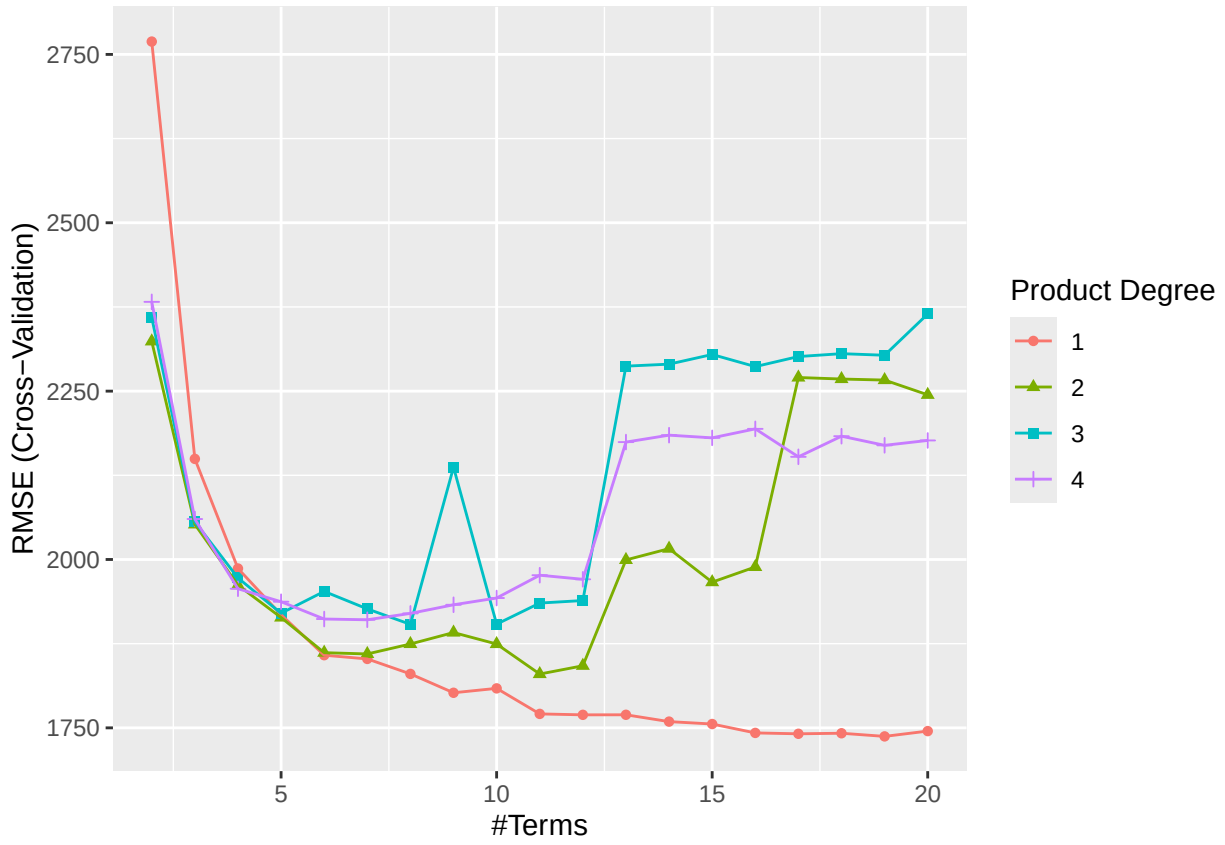
In order to fit a piecewise linear mode using MARs, it is necessary to undertake a grid search to identify the optimal combination of the two hyperparameters (the degree of interactions, and the number of retained terms) that minimise prediction error.

In the college dataset, there are greater than 15 predictors - we can therefore expect some interactions to potentially be important to the model. Therefore, I opted to set `degree` to 1:4, allowing me to represent an appropriate level of interactions (two-way interactions) without too much complexity.

For the number of retained terms (`nprune`), I set the range to 2:25 because I assumed this provides a good balance between model simplicity and flexibility given our dataset size. With 565 observations and 17 variables, setting the maximum number of terms to 25 allows us to avoid overfitting while still allowing the model to capture complexity in the data.

```
# Set seed for reproducibility
set.seed(299)

#
ctrl1 <- trainControl(method = "cv", number = 10)

# Convert predictors to matrix (excl. response variable)
x <- model.matrix(Outstate ~ ., college)[, -1]

# Vector of response
y <- college$Outstate


# Specify MARS grid - first attempt using Professor's example
# mars_grid <- expand.grid(
#  degree = 1:3, # degree of interactions
#  nprune = 2:15  # no. of retained terms
# )

# Specify MARS grid - expanding grid to observe pattern
mars_grid <- expand.grid(
  degree = 1:4, # degree of interactions
  nprune = 2:20  # no. of retained terms
)

#
mars.fit <- train(x, y,
                  method = "earth",
                  tuneGrid = mars_grid,
                  trControl = ctrl1)

# Fit the ggplot
```

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



To ensure I identified the optimal model complexity, I expanded the grid search to include degree = 1:4 and nprune = 2:20. This allowed me to observe how prediction error varied across different interaction levels and numbers of retained terms.

From the results, I observed a clear dip in RMSE when degree = 1, followed by stabilisation, indicating that interactions beyond first-order did not meaningfully improve predictive performance. When I increased the interaction degree, RMSE values became more variable rather than decreasing steadily, which indicated there was no reliable reduction in prediction error. Given this, I selected degree = 1 as the optimal choice. The range nprune = 2:20 was also appropriate, as RMSE initially decreased as nprune increased but then stabilised, indicating that adding more terms would not significantly improve prediction accuracy.

```
# Set seed for reproducibility
set.seed(299)

# Optimal parameters
mars.fit$bestTune
```

```
##    nprune degree
## 18     19      1
```

```
# Final model coefficients
mars_coef <- coef(mars.fit$finalModel)

mars_coef
```

```
##                              (Intercept)
##                            10476.6698258
##                          h(Expend-15622)
##                                -0.7318360
##                        h(4440-Room.Board)
##                                -1.1920042
##                           h(95-Grad.Rate)
##                               -24.6933615
##                       h(F.Undergrad-1350)
##                                -0.3237106
##                       h(1350-F.Undergrad)
##                                -1.4835098
##                         h(22-perc.alumni)
##                               -64.1867488
##                             h(Apps-3767)
##                                0.3686378
##                          h(1300-Personal)
##                                0.9708840
##                             h(888-Enroll)
##                                4.4059678
##                            h(2153-Accept)
##                                -1.8830272
##                 CollegeBennington College
##                              6114.5453214
## CollegeWentworth Institute of Technology
##                             -6123.9314722
##                CollegeLivingstone College
##                             -5895.7350028
##                    CollegeSpelman College
##                             -5459.9745913
##                           h(Expend-5970)
##                                0.7332190
##               CollegeCreighton University
##                             -5980.8633600
##                 CollegeTrinity University
##                             -5636.2060527
##   CollegeArkansas College (Lyon College)
##                             -5539.9128105
```

The best-tuned model selected `degree = 1` and `nprune = 19`. Therefore, the final model includes 19 retained terms with no interaction effects.

The regression equation is as follows:

$$
\begin{aligned}
\widehat{y} = {} & 10476.6698258 \\
& - 0.731836010774518 \cdot h(\text{Expend} - 15622) - 1.19200417180168 \cdot h(4440 - \text{Room.Board}) \\
& - 24.6933614706046 \cdot h(95 - \text{Grad.Rate}) - 0.323710582204791 \cdot h(\text{F.Undergrad} - 1350) \\
& - 1.48350979223261 \cdot h(1350 - \text{F.Undergrad}) - 64.1867488487131 \cdot h(22 - \text{perc.alumni}) \\
& + 0.368637778664394 \cdot h(\text{Apps} - 3767) + 0.970883958999635 \cdot h(1300 - \text{Personal}) \\
& + 4.40596783428093 \cdot h(888 - \text{Enroll}) - 1.88302716644046 \cdot h(2153 - \text{Accept}) \\
& + 6114.54532141715 \cdot I(\text{CollegeBenningtonCollege}) - 6123.93147218096 \cdot I(\text{CollegeWentworthInstituteofTechnology}) \\
& - 5895.73500281939 \cdot I(\text{CollegeLivingstoneCollege}) - 5459.97459128078 \cdot I(\text{CollegeSpelmanCollege}) \\
& + 0.73321903304858 \cdot h(\text{Expend} - 5970) - 5980.86335998011 \cdot I(\text{CollegeCreightonUniversity}) \\
& - 5636.20605272795 \cdot I(\text{CollegeTrinityUniversity}) - 5539.91281045007 \cdot I(\text{CollegeArkansasCollege(LyonCollege)})
\end{aligned}
$$