Module 6

> You can download the R Markdown file (https://gedeck.github.io/DS-6030/homework/Module-6.Rmd (https://gedeck.github.io/DS-6030/homework/Module-6.Rmd)) and use it to answer the following questions.
>
> If not otherwise stated, use Tidyverse and Tidymodels for the assignments.

1. Predict out of state tuition (feature selection)

The data `College.csv` contains a number of variables for 777 different universities and colleges in the US. In this exercise, we will try to predict the `Outstate` tuition fee using the other variables in the data set.

A. Data loading and preprocessing

```
#| message: FALSE
#| warning: FALSE
#| cache: TRUE

#Load packages

library(tidymodels)
```

```
## ── Attaching packages ─────────────────────────────────────── tidymodels 1.3.0 ──
```

```
## ✓ broom        1.0.8     ✓ recipes      1.3.0
## ✓ dials        1.4.0     ✓ rsample      1.3.0
## ✓ dplyr        1.1.4     ✓ tibble       3.2.1
## ✓ ggplot2      3.5.2     ✓ tidyr        1.3.1
## ✓ infer        1.0.8     ✓ tune         1.3.0
## ✓ modeldata    1.4.0     ✓ workflows    1.2.0
## ✓ parsnip      1.3.1     ✓ workflowsets 1.1.0
## ✓ purrr        1.0.4     ✓ yardstick    1.3.2
```

```
## ── Conflicts ────────────────────────────────────────── tidymodels_conflicts() ──
## ✖ purrr::discard() masks scales::discard()
## ✖ dplyr::filter()  masks stats::filter()
## ✖ dplyr::lag()     masks stats::lag()
## ✖ recipes::step()  masks stats::step()
```

```
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✓ forcats   1.0.0     ✓ readr     2.1.5
## ✓ lubridate 1.9.4     ✓ stringr   1.5.1
```

```
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ readr::col_factor() masks scales::col_factor()
## ✖ purrr::discard()    masks scales::discard()
## ✖ dplyr::filter()     masks stats::filter()
## ✖ stringr::fixed()    masks recipes::fixed()
## ✖ dplyr::lag()        masks stats::lag()
## ✖ readr::spec()       masks yardstick::spec()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
e errors
```

```
library(ISLR2)
library(mgcv)
```

```
## Loading required package: nlme
##
## Attaching package: 'nlme'
##
## The following object is masked from 'package:dplyr':
##
##     collapse
##
## This is mgcv 1.9-3. For overview type 'help("mgcv-package")'.
```

```
library(patchwork)
```

(1.1) Load the data from `ISLR2::College` and split into training and holdout sets using a 80/20 split. (1 point - coding)

```
set.seed(345)

#Load data

college <- ISLR2::College %>%
  as_tibble() %>%
  mutate(Private = as.factor(Private))

#Test/hold sets

set <- initial_split(college, prop = 0.8, strata = Outstate)
train <- training(set)
test <- testing(set)
```

B. Train linear regression and Lasso models

Use *tidymodels* to define a workflow and build a linear regression model to predict `Outstate` from all the other variables using L1 regularization (Lasso).

(1.2) For preprocessing, normalize all the numerical variables (`step_normalize(all_numeric_predictors()`) and convert the categorical / nominal variables to dummy variables (`step_dummy(all_nominal_predictors())`). (1 point - coding)

```
rec <- recipe(Outstate ~ ., data = train) %>%
  step_normalize(all_numeric_predictors()) %>%
  step_dummy(all_nominal_predictors())
```

(1.3) Train a normal linear regression model using the `lm` engine with the training set. Look at the p-values of the individual features. Which features are significant (use `extract_fit_engine` and `summary`). (1 point - coding/discussion)

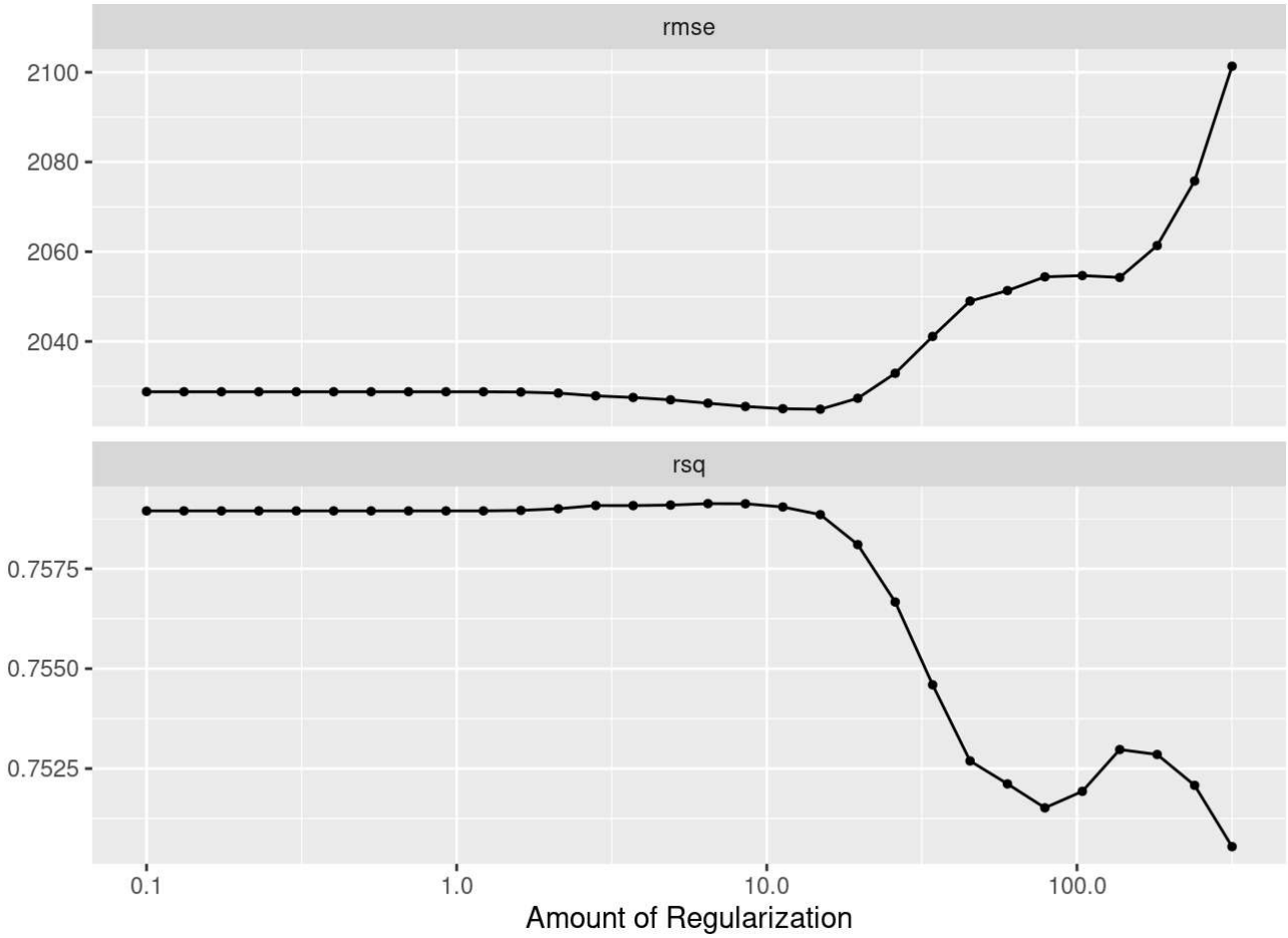
```
lm_model <- linear_reg() %>%
  set_engine("lm")

lm_wf <- workflow() %>%
  add_model(lm_model) %>%
  add_recipe(rec)

lm_fit <- lm_wf %>%
  fit(data = train)

#P-values

lm_summary <- lm_fit %>%
  extract_fit_engine() %>%
  summary()

lm_summary$coefficients
```

```
##                Estimate Std. Error    t value      Pr(>|t|)
## (Intercept)  9003.09353  220.91238 40.7541368 3.041944e-175
## Apps        -1252.48888  295.24569 -4.2421919  2.562231e-05
## Accept       1944.32624  354.69494  5.4816859  6.203584e-08
## Enroll       -115.77605  364.62948 -0.3175170  7.509615e-01
## Top10perc     646.03278  222.00638  2.9099740  3.748143e-03
## Top25perc    -201.42310  190.10963 -1.0595102  2.897924e-01
## F.Undergrad  -769.86022  340.91939 -2.2581884  2.429113e-02
## P.Undergrad    28.15111  107.84457  0.2610341  7.941555e-01
## Room.Board   1090.19191  106.89445 10.1987698  1.228352e-22
## Books        -109.92821   83.26308 -1.3202516  1.872528e-01
## Personal     -139.71091   89.93623 -1.5534442  1.208426e-01
## PhD           144.02247  156.52340  0.9201337  3.578714e-01
## Terminal      445.91182  155.14606  2.8741422  4.194293e-03
## S.F.Ratio    -272.95131  109.25008 -2.4984083  1.274077e-02
## perc.alumni   526.14587  105.83532  4.9713637  8.680641e-07
## Expend        922.44519  130.35158  7.0765937  4.124573e-12
## Grad.Rate     422.52680  104.69126  4.0359319  6.140038e-05
## Private_Yes  1968.53933  279.37023  7.0463461  5.043074e-12
```

Based on the p-values, it seems that the significant features are Grad.Rate, perc.alumni, Room.Board, Accept, Terminal, and Expend.

(1.4) Use `glmnet` and tune the L1 penalty parameter using 10-fold cross-validation. Tune the penalty over the range `penalty(c(-1, 2.5))` and check that the range is appropriate using autoplot. (2 points - coding)

```
lasso_model <- linear_reg(penalty = tune(), mixture = 1) %>%
  set_engine("glmnet")

lasso_wf <- workflow() %>%
  add_model(lasso_model) %>%
  add_recipe(rec)

#Grid

lasso_grid <- grid_regular(penalty(c(-1,2.5)), levels = 30)

#10-fold

set.seed(234)
lasso_res <- tune_grid(
  lasso_wf,
  resamples = vfold_cv(train, v = 10, strata = Outstate),
  grid = lasso_grid
)

autoplot(lasso_res)
```



(1.5) Determine the best penalty parameters using the lowest $RMSE$ and the penalty obtained from the one-standard-error rule (`select_by_one_std_err` , see online material (https://gedeck.github.io/DS-6030/book/model-tuning-examples.html#tune-one-standard-error)). For both penalties, finalize the workflow and train models using the full training set. Report the coefficients of each model. Which variables are selected in each case? (2 points - coding/discussion)

```
#Lowest RMSE

best_rmse <- select_best(lasso_res, metric = "rmse")

#One standard error

one_se <- select_by_one_std_err(
  lasso_res, metric = "rmse", desc(penalty)
)

#Final workflows

final_rmse_wf <- finalize_workflow(lasso_wf, best_rmse)
final_one_se_wf <- finalize_workflow(lasso_wf, one_se)

#Models fit

final_rmse_fit <- fit(final_rmse_wf, data = train)
final_one_se_fit <- fit(final_one_se_wf, data = train)

#Coefficients of models

coef_rmse <- tidy(final_rmse_fit)
coef_one_se <- tidy(final_one_se_fit)

coef_rmse
```

```
## # A tibble: 18 × 3
##    term         estimate penalty
##    <chr>           <dbl>   <dbl>
##  1 (Intercept) 8947.        14.9
##  2 Apps        -759.        14.9
##  3 Accept      1313.        14.9
##  4 Enroll         0         14.9
##  5 Top10perc    364.        14.9
##  6 Top25perc     -0.0197    14.9
##  7 F.Undergrad -679.        14.9
##  8 P.Undergrad    0         14.9
##  9 Room.Board  1089.        14.9
## 10 Books        -96.9       14.9
## 11 Personal    -140.        14.9
## 12 PhD          167.        14.9
## 13 Terminal     425.        14.9
## 14 S.F.Ratio   -273.        14.9
## 15 perc.alumni  517.        14.9
## 16 Expend       927.        14.9
## 17 Grad.Rate    414.        14.9
## 18 Private_Yes 2044.        14.9
```

```
coef_one_se
```

```
## # A tibble: 18 × 3
##    term        estimate penalty
##    <chr>          <dbl>   <dbl>
##  1 (Intercept)   9011.     240.
##  2 Apps             0      240.
##  3 Accept           0      240.
##  4 Enroll           0      240.
##  5 Top10perc      140.     240.
##  6 Top25perc        0      240.
##  7 F.Undergrad      0      240.
##  8 P.Undergrad      0      240.
##  9 Room.Board    1092.     240.
## 10 Books            0      240.
## 11 Personal       -49.7    240.
## 12 PhD             112.     240.
## 13 Terminal        310.     240.
## 14 S.F.Ratio      -225.     240.
## 15 perc.alumni     521.     240.
## 16 Expend          923.     240.
## 17 Grad.Rate       431.     240.
## 18 Private_Yes    1957.     240.
```

It seems that the One standard error model selects less variables than the RMSE model, thus making the model more regularlized. All the variables were selected for the One standard error model except Apps, Accept, Enroll, Top25Perc, F.Undergrad, P.Undergrad, and Books.

(1.6) Use the model from (1.3) and the two models from (1.5) to predict the `Outstate` variable on the training and test set. Report the RMSE and R^2 of each model on the training and test set. (1 point - coding/discussion)

```
get_metrics <- function(model, train, test) {
  bind_rows(
    metrics(model %>% augment(train), truth = Outstate, estimate = .pred) %>%
      mutate(dataset = "train"),
    metrics(model %>% augment(train), truth = Outstate, estimate = .pred) %>%
      mutate(dataset = "test")
  )
}

#Metrics

lm_metrics <- get_metrics(lm_fit, train, test) %>% mutate(model = "Linear")
rmse_metrics <- get_metrics(final_rmse_fit, train, test) %>% mutate(model = "Lasso RSME")
one_se_metrics <- get_metrics(final_one_se_fit, train, test) %>% mutate(model = "Lasso 1SE")

all_metrics <- bind_rows(lm_metrics, rmse_metrics, one_se_metrics)

all_metrics %>%
  pivot_wider(names_from = .metric, values_from = .estimate)
```

```
## # A tibble: 6 × 6
##   .estimator dataset model       rmse   rsq   mae
##   <chr>      <chr>   <chr>      <dbl> <dbl> <dbl>
## 1 standard   train   Linear     1934. 0.773 1520.
## 2 standard   test    Linear     1934. 0.773 1520.
## 3 standard   train   Lasso RSME 1941. 0.772 1524.
## 4 standard   test    Lasso RSME 1941. 0.772 1524.
## 5 standard   train   Lasso 1SE  2037. 0.756 1598.
## 6 standard   test    Lasso 1SE  2037. 0.756 1598.
```

Based on the values of the RSME and R^2 for each metric, it seems that linear regression and RSME perform well, but there could be some possible overfitting. The One standard error shows good peformance as well, but has less possibility of overfitting.

2. Predict out of state tuition (GAM model)

C. GAM model

Using the significant features from (1.3), build a generalized additive model (GAM) to predict `Outstate`. (see Generalized additive models (GAM) (https://gedeck.github.io/DS-6030/book/deep-dive-gen_additive_mod.html) for how to build GAM models in `tidymodels`)

(2.1) Define a model formula setting all numerical variables as splines and all categorical variables as factors (`Outstate ~ Private + s(Apps) + ...`) (1 point - coding)

```
gam_formula <- Outstate ~ Private + s(Grad.Rate) + s(perc.alumni) + s(Room.Board) + s(Expend) +
s(Terminal) + s(Accept)
```

(2.2) Define the `gen_additive_mod` model using `mgcv` as the engine and fit the model using the training data (2 points - coding)

```
gam_model <- gen_additive_mod() %>%
  set_engine("mgcv") %>%
  set_mode("regression")

#Model fit

gam_fit <- fit(gam_model, formula = gam_formula, data = train)
```

(2.3) Report the RMSE and R^2 of the model on the training and test set and compare with the result from **(1.6)** (1 point - discussion)

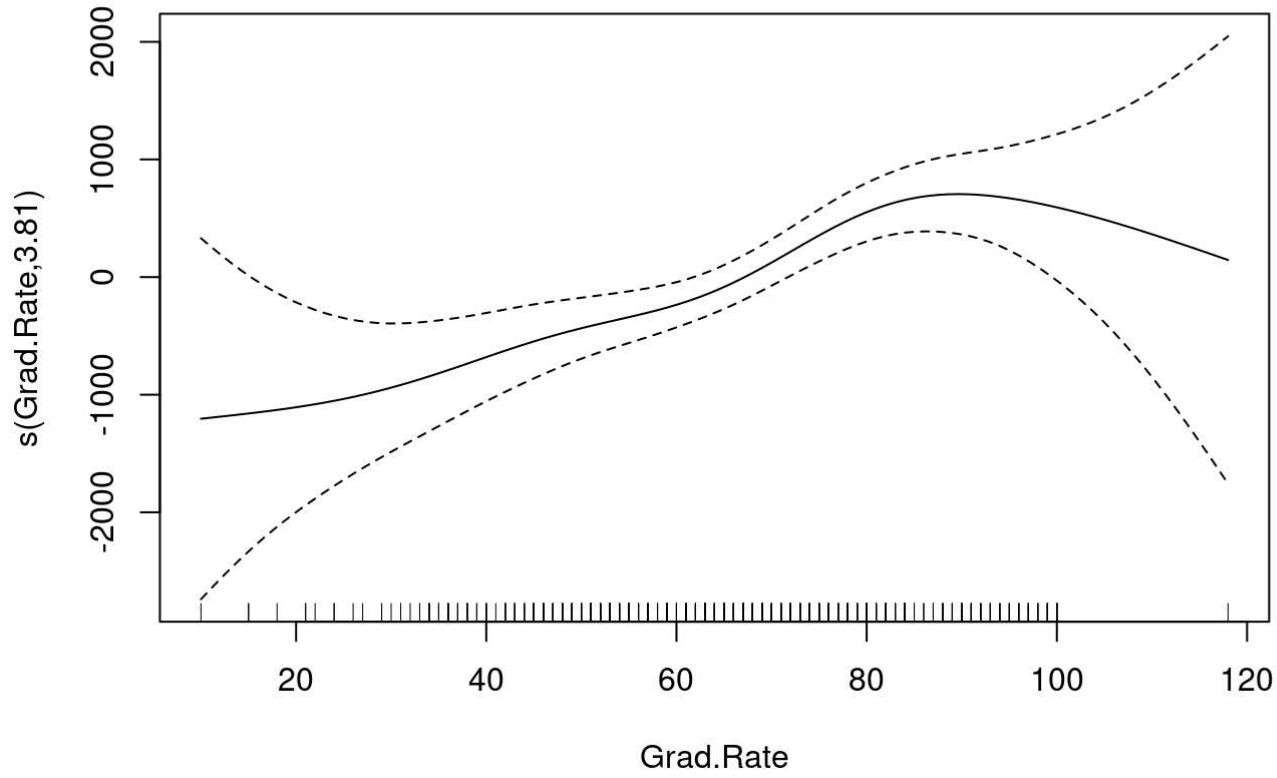
```
gam_metrics <- get_metrics(gam_fit, train, test) %>%
  mutate(model = "GAM")

gam_metrics %>%
  pivot_wider(names_from = .metric, values_from = .estimate)
```

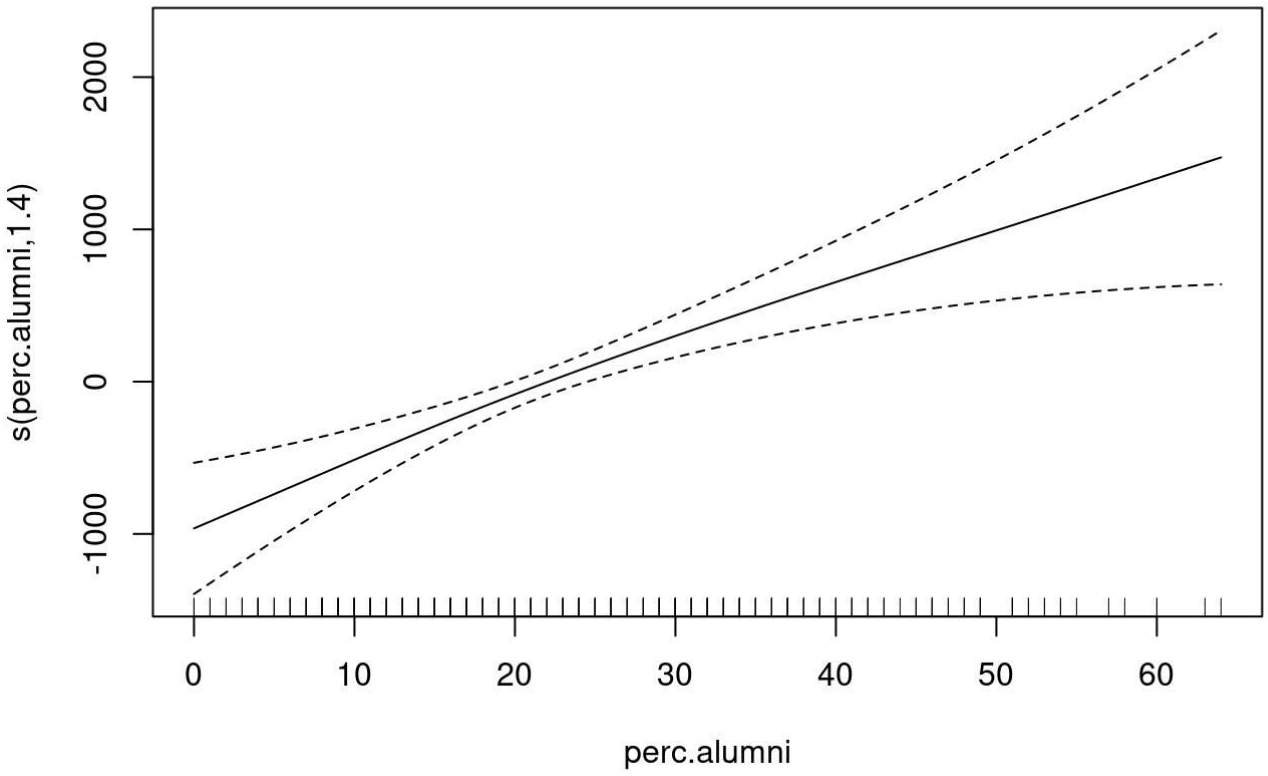
```
## # A tibble: 2 × 6
##   .estimator dataset model  rmse   rsq   mae
##   <chr>      <chr>   <chr> <dbl> <dbl> <dbl>
## 1 standard   train   GAM   1782. 0.807 1367.
## 2 standard   test    GAM   1782. 0.807 1367.
```

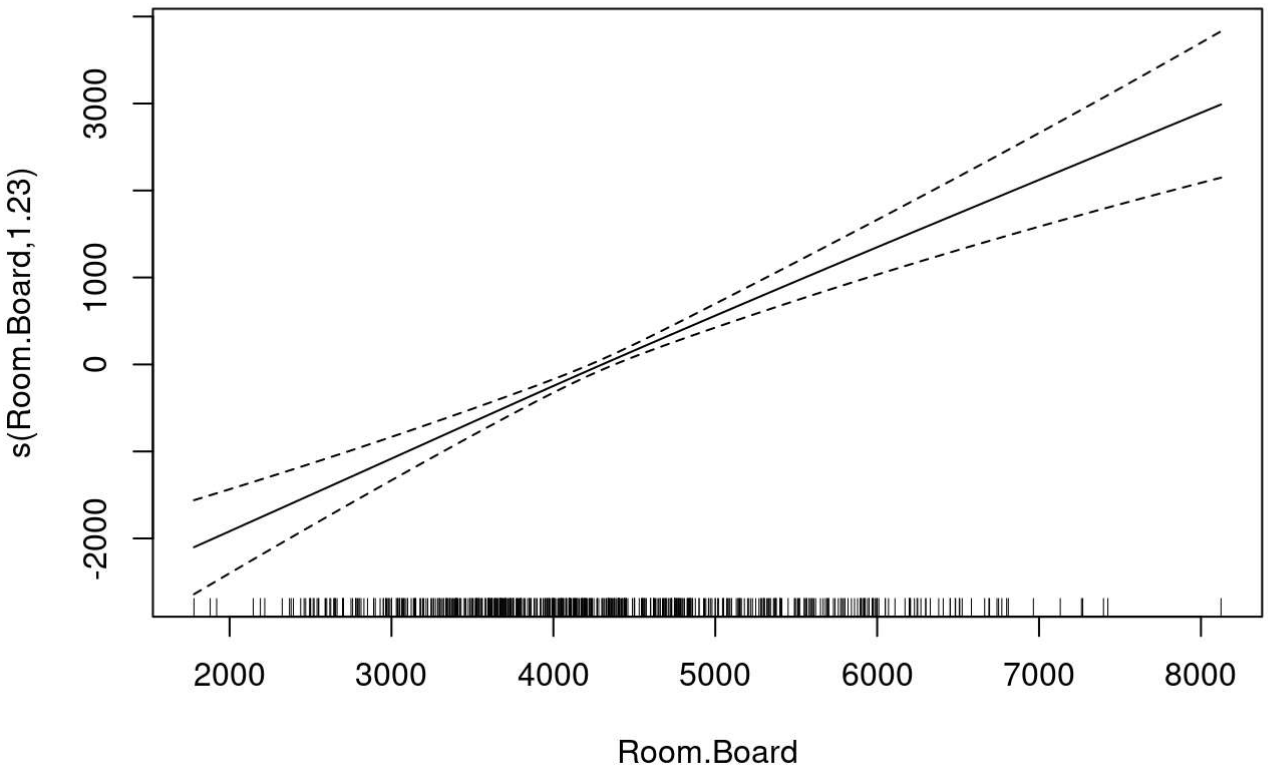
Based on the metric values from GAM, it looks like GAM performed better than the Linear and Lasso models. This is most likely due to its ability to capture non-linear relationships.

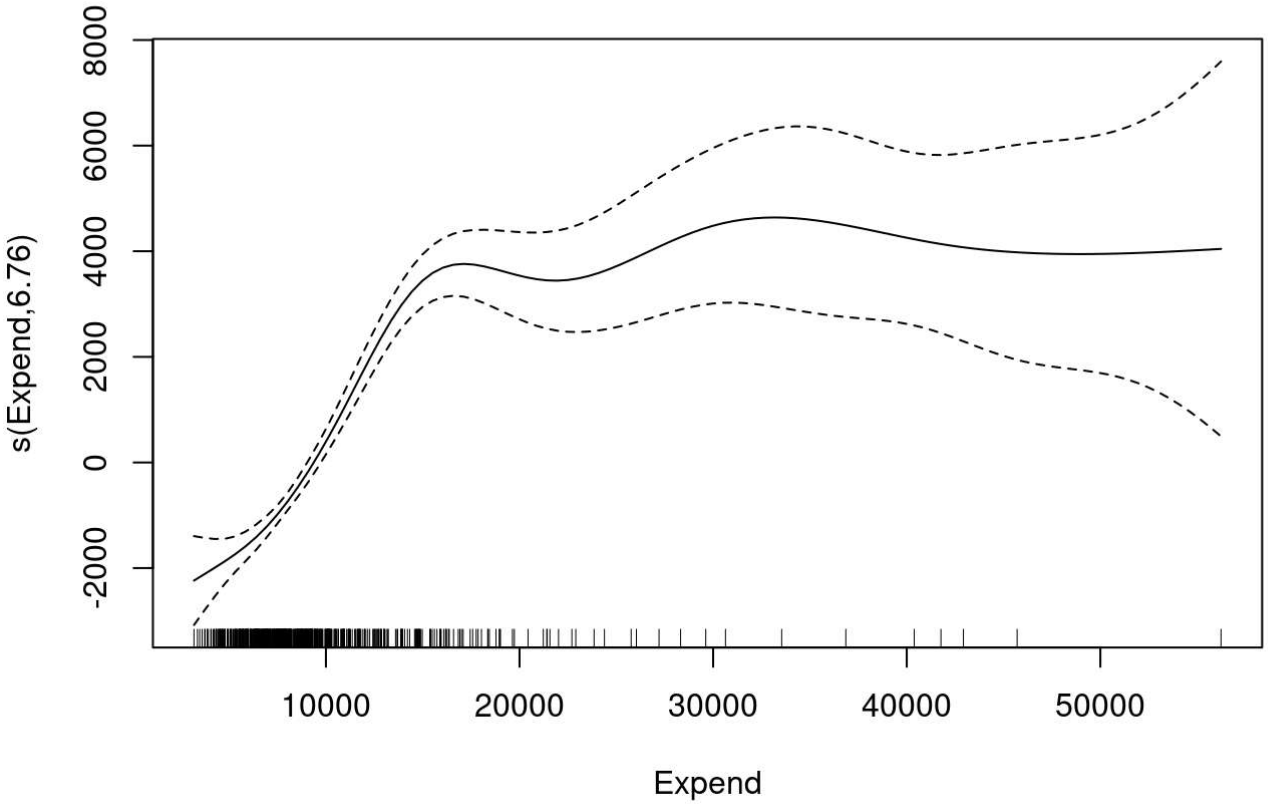
(2.4) Use the `plot` function with the fitted model (use `extract_fit_engine` to get the actual `mgcv` model for plotting; set `scale=0` to get individual y-scales). Describe your observations. (1 point - coding/discussion)

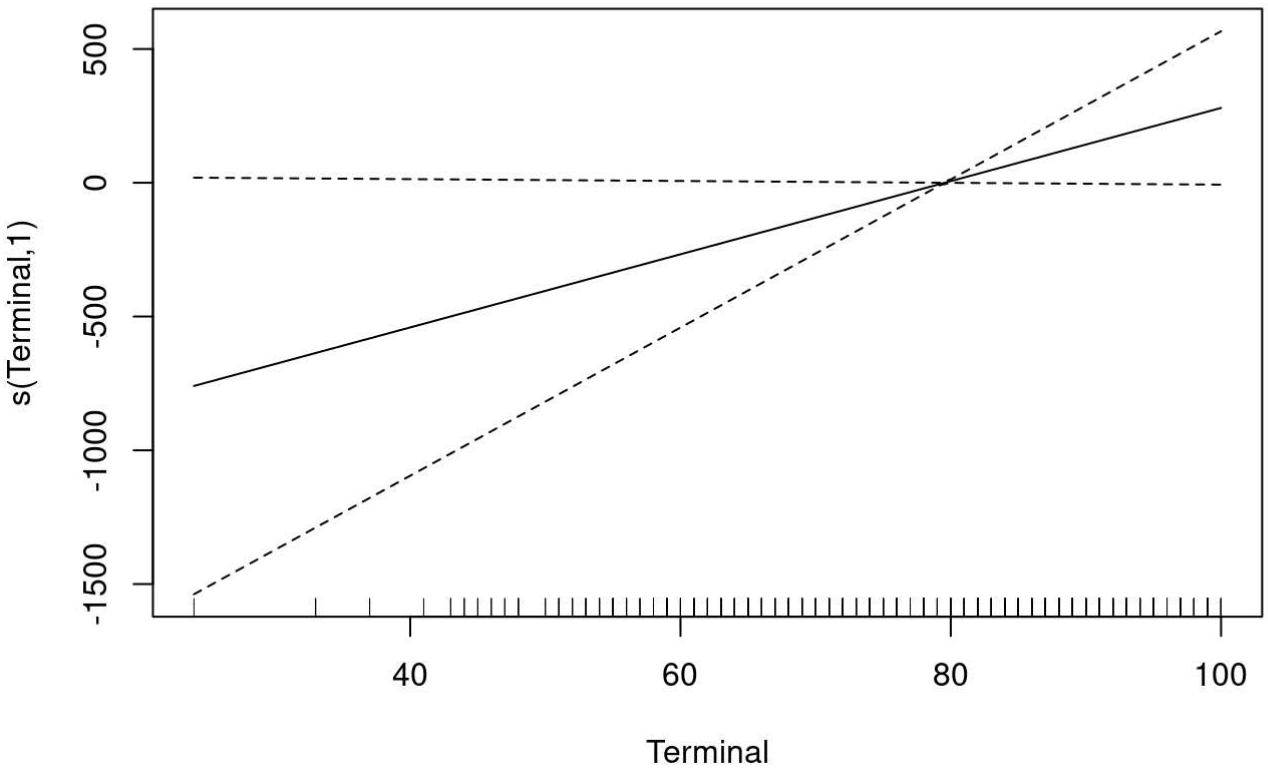
```
gam_engine <- gam_fit %>%
  extract_fit_engine()

plot(gam_engine, scale = 0)
```

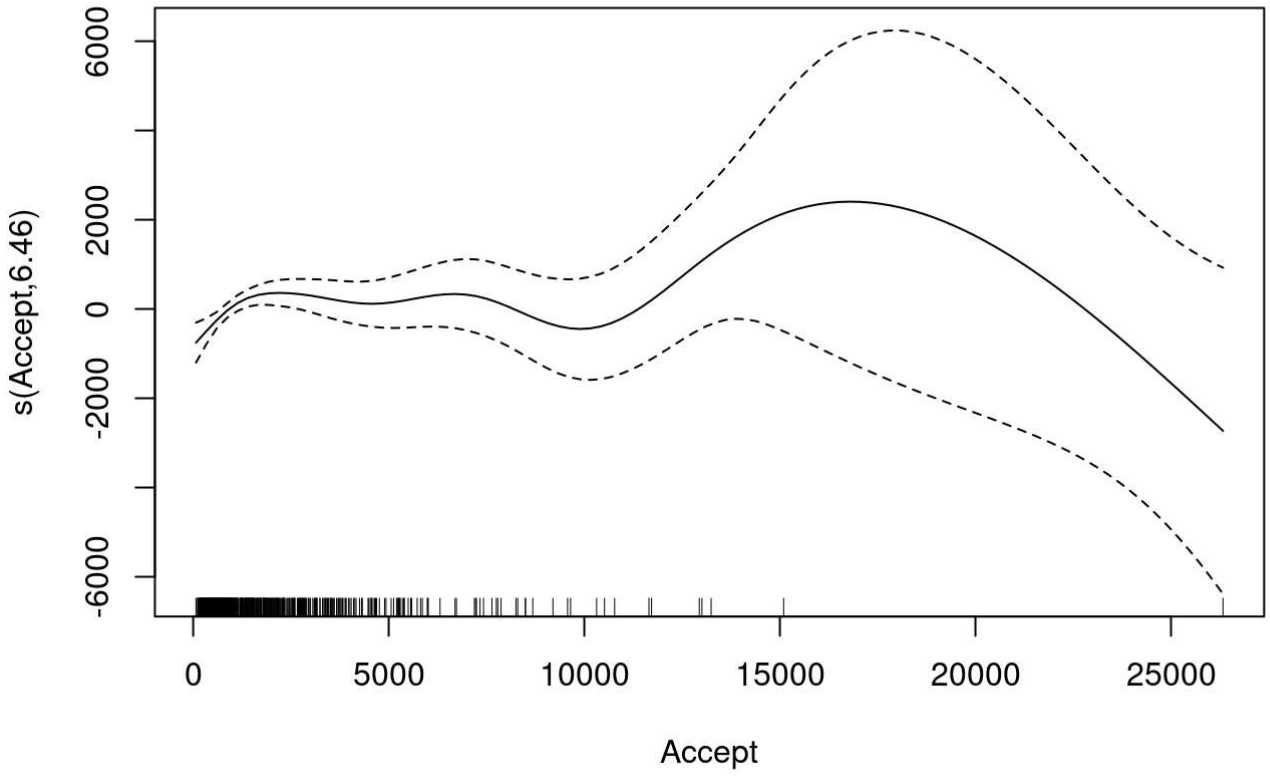



s(perc.alumni,1.4)
2000
1000
0
-1000
0
10
20
30
40
50
60
perc.alumni


s(Room.Board,1.23)
3000
1000
0
-2000
2000
3000
4000
5000
6000
7000
8000
Room.Board


s(Expend,6.76)
8000
6000
4000
2000
0
-2000
10000
20000
30000
40000
50000
Expend


s(Terminal,1)
500
0
-500
-1000
-1500
40
60
80
100
Terminal



The plots show smooth, non-linear relationships between predictors Accept, Expend, and Grad.Rate.

(2.5) Use the `summary` function to get information about the model. Based on the reported significance levels, could you simplify the model further? (1 point - coding/discussion)

```
summary(gam_engine)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## Outstate ~ Private + s(Grad.Rate) + s(perc.alumni) + s(Room.Board) +
##     s(Expend) + s(Terminal) + s(Accept)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   8646.5      205.8  42.006   <2e-16 ***
## PrivateYes    2451.2      260.6   9.407   <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(Grad.Rate)   3.814  4.806  5.932 5.37e-05 ***
## s(perc.alumni) 1.396  1.701 13.743 1.00e-05 ***
## s(Room.Board)  1.225  1.420 52.151  < 2e-16 ***
## s(Expend)      6.760  7.851 33.636  < 2e-16 ***
## s(Terminal)    1.000  1.000  3.806   0.0515 .
## s(Accept)      6.461  7.541  2.540   0.0110 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =    0.8   Deviance explained = 80.7%
## GCV = 3.4193e+06  Scale est. = 3.2944e+06  n = 620
```

Based on the p-values, it seems as though Terminal could be taken out to simplify the model.

(2.6) Simplify the model by removing the non-significant variables and re-fit the model. Report the RMSE and R^2 of the model on the training and test set. (1 point - coding/discussion)

```
gam_formula_simple <- Outstate ~ Private + s(Grad.Rate) + s(perc.alumni) + s(Room.Board) + s(Exp
end) + s(Accept)

gam_model_simple <- gen_additive_mod() %>%
  set_engine("mgcv") %>%
  set_mode("regression")

#Model fit

gam_fit_simple <- fit(gam_model_simple, formula = gam_formula_simple, data = train)

# Check performance
gam_simple_metrics <- get_metrics(gam_fit_simple, train, test) %>%
  mutate(model = "GAM Simplified")

gam_simple_metrics %>%
  pivot_wider(names_from = .metric, values_from = .estimate)
```

```
## # A tibble: 2 × 6
##   .estimator dataset model           rmse   rsq   mae
##   <chr>      <chr>   <chr>          <dbl> <dbl> <dbl>
## 1 standard   train   GAM Simplified 1786. 0.807 1374.
## 2 standard   test    GAM Simplified 1786. 0.807 1374.
```

The RMSE for the training and test set is 1785.732, and the R^2 for the training and test set is 0.8066 for both sets.

D. Comparison

(2.7) Compare the results from the three models from (B), the full GAM model from **(2.2)** and the reduced model from **(2.6)**? (2 points - discussion)

```
all_model_matrics <- bind_rows(
  all_metrics,
  gam_metrics,
  gam_simple_metrics
)

all_model_matrics %>%
  pivot_wider(names_from = .metric, values_from = .estimate)
```

```
## # A tibble: 10 × 6
##    .estimator dataset model           rmse   rsq   mae
##    <chr>      <chr>   <chr>          <dbl> <dbl> <dbl>
##  1 standard   train   Linear         1934. 0.773 1520.
##  2 standard   test    Linear         1934. 0.773 1520.
##  3 standard   train   Lasso RSME     1941. 0.772 1524.
##  4 standard   test    Lasso RSME     1941. 0.772 1524.
##  5 standard   train   Lasso 1SE      2037. 0.756 1598.
##  6 standard   test    Lasso 1SE      2037. 0.756 1598.
##  7 standard   train   GAM            1782. 0.807 1367.
##  8 standard   test    GAM            1782. 0.807 1367.
##  9 standard   train   GAM Simplified 1786. 0.807 1374.
## 10 standard   test    GAM Simplified 1786. 0.807 1374.
```

From all of the metrics, it seems that the full GAM model has the best performance. It has the lowest RMSE, as well as the highest R^2. The simplified GAM also seems to have performed better than the Linear and One stanard error models, but doesn't perform as well as the full GAM model.