yc4384_Yangyang_Chen_HW4

yc4384_Yangyang_Chen

2024-04-10

Load packages

```
library(tidyverse)
library(caret)
library(rpart.plot)
library(ranger)
library(gbm)
library(knitr)
library(party)
library(ISLR)
library(pROC)
```

Problem 1: How Much is Your Out-of-State Tuition?

Load and split data into training and testing sets

```
set.seed(2024)

# import and tidy
data = read_csv("College.csv") |>
  janitor::clean_names() |>
  select(-college)

# partition data into training and testing sets as randomized 4:1 splits
train_index = createDataPartition(y = data$outstate, p = 0.8, list = F)
train_data = data[train_index, ]
test_data = data[-train_index, ]

# testing set response for RMSE calculation
test_resp = test_data$outstate
```

Set cross validation methods

```
# for regression tree
ctrl_re = trainControl(method = "repeatedcv", number = 2, repeats = 5)

# for classification tree under the minimal MSE rule
ctrl = trainControl(method = "repeatedcv", number = 2, repeats = 5,
                       summaryFunction = twoClassSummary,
                       classProbs = TRUE)

# for classification tree under the 1SE rule
ctrl_1se = trainControl(method = "repeatedcv", number = 2, repeats = 5,
                        summaryFunction = twoClassSummary,
                        classProbs = TRUE,
                        selectionFunction = "oneSE")
```

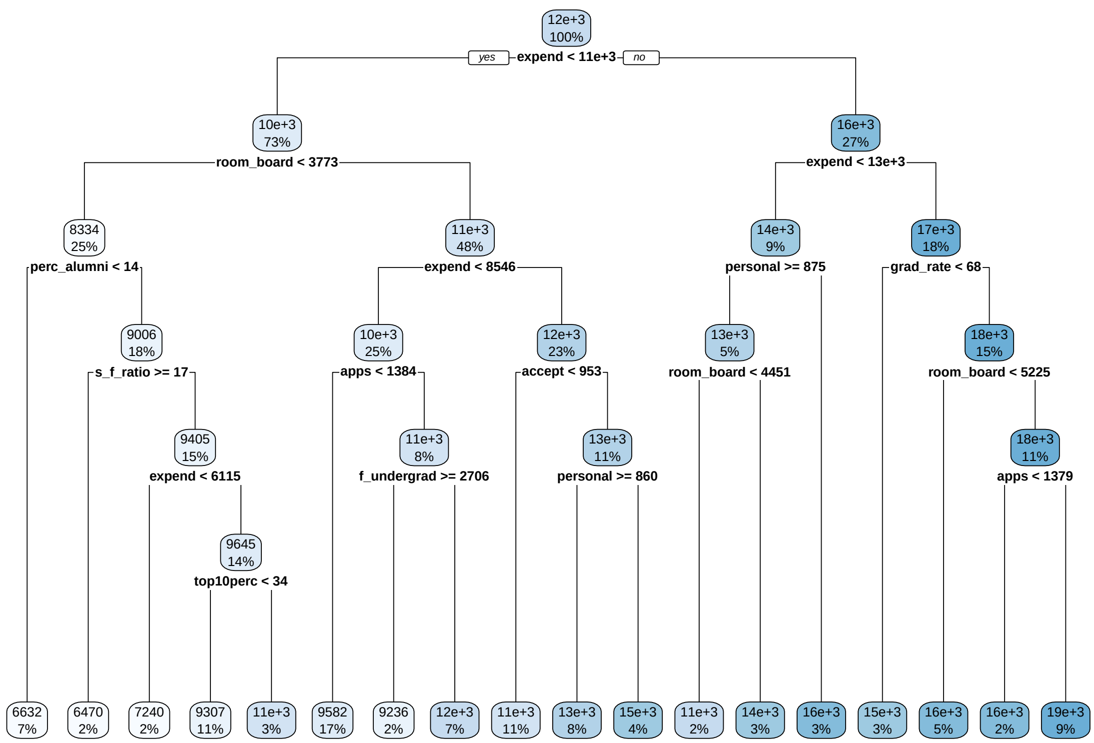
a) **Fit and plot a regression tree model**

Use the regression tree (CART) approach to graph an optimally pruned tree. At the top (root) of the tree, it is shown that splitting at `expend` over or under 11K provides significantly more accurate predictions for out-of-state tuitions than any other.

```
set.seed(2024)

rpart_grid = data.frame(cp = exp(seq(-8,-5, length = 100)))
rpart_fit = train(outstate ~ . ,
                  data,
                  subset = train_index,
                  method = "rpart",
                  tuneGrid = rpart_grid,
                  trControl = ctrl_re)
# ggplot(rpart_fit, highlight = TRUE)

rpart.plot(rpart_fit$finalModel)
```



For comparison, the following is the code using the conditional inference tree (CIT) approach. The code generates an overly cluttered graph but `expend` is still atop the decision tree.

```
set.seed(2022)

ctree_grid = data.frame(mincriterion = 1-exp(seq(-2, 0, length = 100)))
ctree_fit = train(outstate ~ . ,
                  data,
                  subset = train_index,
                  method = "ctree",
                  tuneGrid = ctree_grid,
```

```r
                   trControl = ctrl_re)
ggplot(ctree_fit, highlight = TRUE)

plot(ctree_fit$finalModel)

RMSE(predict(ctree_fit, newdata = test_data), test_resp)
```

b) **Fit and evaluate a random forest regression model**

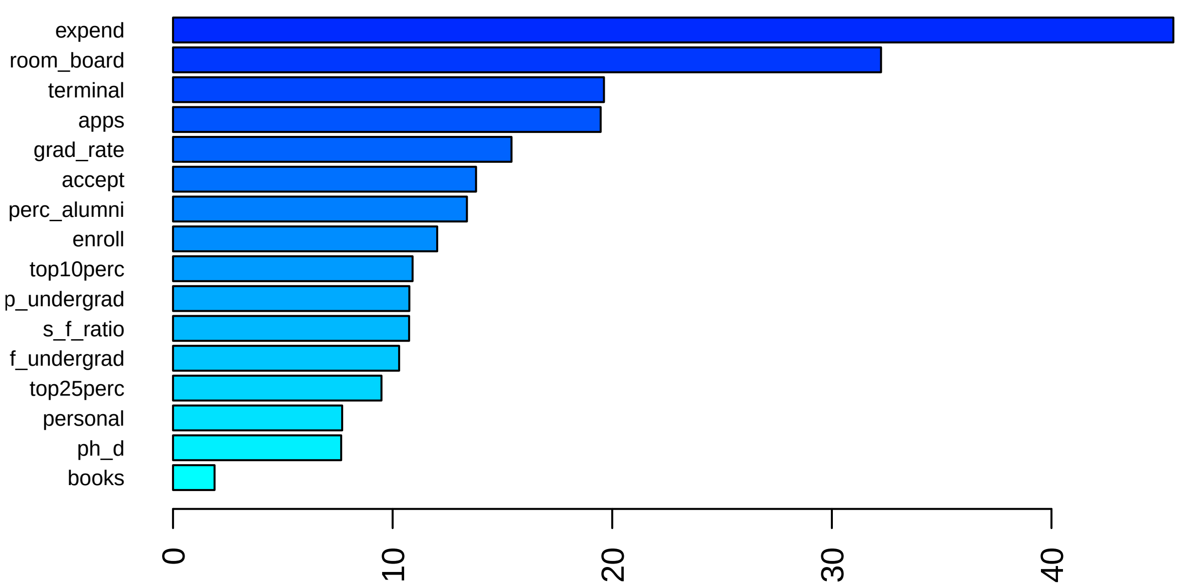
```
set.seed(2022)

rf_grid = expand.grid(mtry = 1:16,
                      splitrule = "variance",
                      min.node.size = 1:6)
rf_fit = train(outstate ~ . ,
               data,
               subset = train_index,
               method = "ranger",
               tuneGrid = rf_grid,
               trControl = ctrl_re)
# ggplot(rf_fit, highlight = TRUE)
```

Calculate and graph variable importance using permutation and impurity metrics. Similarly, both evaluations suggest `expend` as the most important predictor for regressing out-of-state tuition, followed by `room-board`.

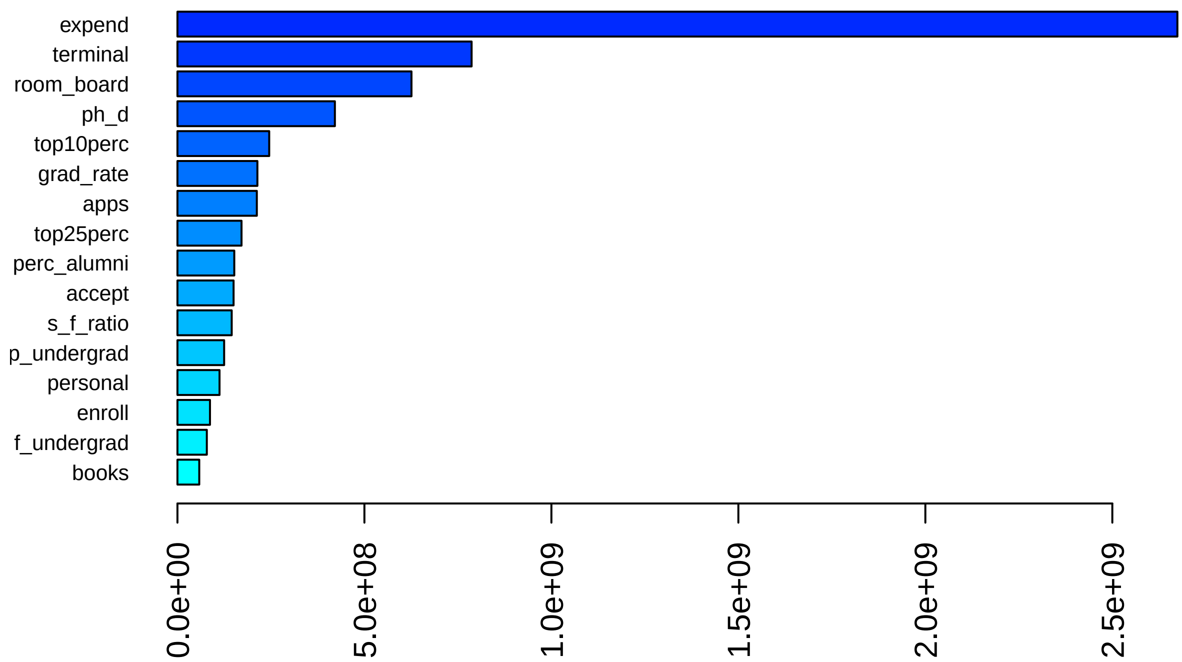
```
set.seed(2022)

rf_perm = ranger(outstate ~ . ,
                 train_data,
                 mtry = rf_fit$bestTune[[1]],
                 splitrule = "variance",
                 min.node.size = rf_fit$bestTune[[3]],
                 importance = "permutation",
                 scale.permutation.importance = TRUE)
barplot(sort(importance(rf_perm), decreasing = FALSE),
        las = 2, horiz = TRUE, cex.names = 0.7,
        col = colorRampPalette(colors = c("cyan","blue"))(19))
```




```
rf_imp = ranger(outstate ~ . ,
                train_data,
                mtry = rf_fit$bestTune[[1]],
```

```
                 splitrule = "variance",
                 min.node.size = rf_fit$bestTune[[3]],
                 importance = "impurity")
barplot(sort(importance(rf_imp), decreasing = FALSE),
        las = 2, horiz = TRUE, cex.names = 0.7,
        col = colorRampPalette(colors = c("cyan","blue"))(19))
```



For the random forest model test error and its interpretation, see the end of part C).

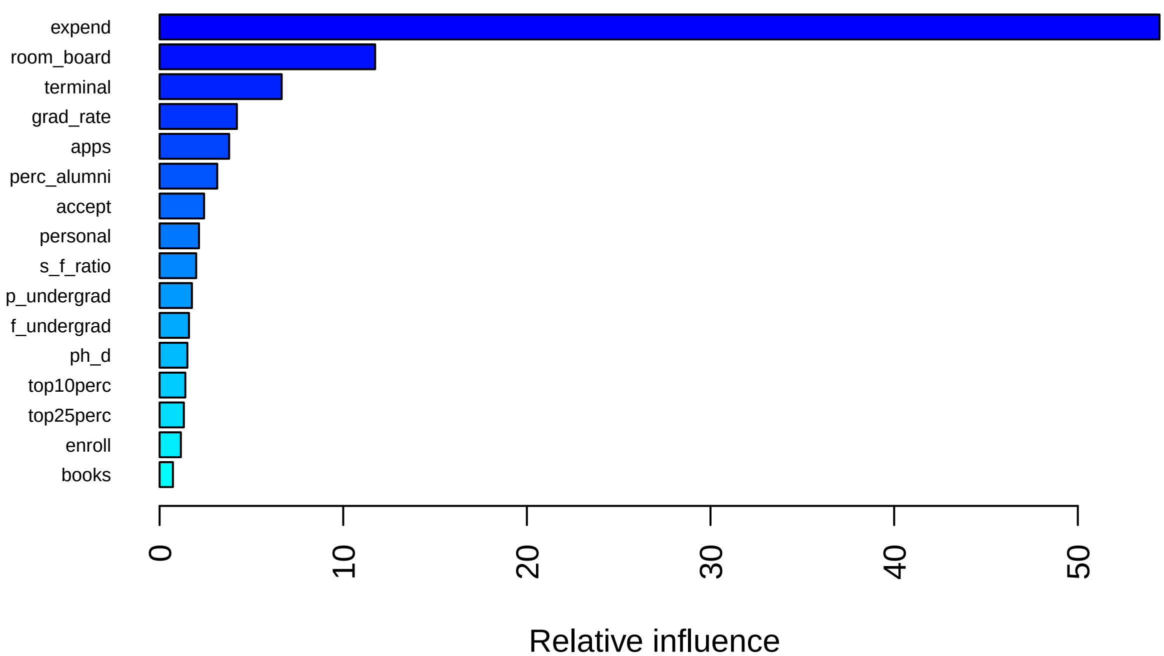
c) **Fit and evaluate a gradient boosting regression model**

```r
set.seed(2022)

gbm_grid = expand.grid(n.trees = c(2000, 3000, 4000, 5000),
                       interaction.depth = 1:5,
                       shrinkage = c(0.001,0.003,0.005),
                       n.minobsinnode = c(1, 10))
gbm_fit = train(outstate ~ . ,
                train_data,
                method = "gbm",
                tuneGrid = gbm_grid,
                trControl = ctrl_re,
                verbose = FALSE)
# ggplot(gbm_fit, highlight = TRUE)
```

Calculate, list and graph variable importance. Again, boosting suggests `expend` and `room-board` as the 2 most important predictors for regressing out-of-state tuition.

```r
summary(gbm_fit$finalModel, las = 2, cBars = 19, cex.names = 0.6)
```




```
##                     var   rel.inf
## expend           expend 54.450006
## room_board   room_board 11.737135
## terminal       terminal  6.647805
## grad_rate     grad_rate  4.208257
## apps               apps  3.787567
## perc_alumni perc_alumni  3.137512
## accept           accept  2.421755
## personal       personal  2.145077
## s_f_ratio     s_f_ratio  1.991794
## p_undergrad p_undergrad  1.759194
```

```
## f_undergrad f_undergrad  1.598989
## ph_d               ph_d  1.512105
## top10perc     top10perc  1.401320
## top25perc     top25perc  1.319406
## enroll           enroll  1.156840
## books             books  0.725240
```

Show test errors for both the random forest and boosting models, and compare them with their cross-validation errors. The boosting model has a lower test error and cross-validation error than those of the random forest model. Notice both their test RMSEs fall in the 4th quartile of their cross-validation errors, which is rather high but still within expectation, and both models could be applied to other new testing sets.

```r
rf_test_rmse = RMSE(predict(rf_fit, newdata = test_data), test_resp)
boost_test_rmse = RMSE(predict(gbm_fit, newdata = test_data), test_resp)
kable(c(rf = rf_test_rmse, boost = boost_test_rmse), col.names = "RMSE", "simple")
```

	RMSE
rf	1681.019
boost	1620.118

```r
summary(resamples(list(rf = rf_fit, boost = gbm_fit)))
```

```
##
## Call:
## summary.resamples(object = resamples(list(rf = rf_fit, boost = gbm_fit)))
##
## Models: rf, boost
## Number of resamples: 10
##
## MAE
##           Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## rf    1366.511 1392.461 1400.688 1415.125 1440.627 1488.689    0
## boost 1306.480 1316.049 1375.384 1364.273 1387.250 1452.675    0
##
## RMSE
##           Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## rf    1770.666 1794.449 1844.571 1871.957 1948.034 2041.791    0
## boost 1712.816 1745.902 1817.403 1815.381 1869.809 1966.971    0
##
## Rsquared
##            Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## rf    0.7012198 0.7409961 0.7586605 0.7541177 0.7748242 0.7782317    0
## boost 0.7256600 0.7577853 0.7693608 0.7664569 0.7746099 0.7963634    0
```