P8106 HW3

Shihui Zhu

Contents

Regression - College Data **2**

(a) Build a regression tree on the training data to predict the response. 2

(b) Perform random forest on the training data. Report the variable importance and the test error. 3

(c) Perform boosting on the training data 5

Classification - `OJ` data **7**

(a) Build a classification tree using the training data, with Purchase as the response and the other variables as predictors. 7

(b) Perform boosting on the training data and report the variable importance. 10

Regression - College Data

```
college <- read_csv("College.csv")[-1] %>%
  janitor::clean_names() %>%
  na.omit()

# training data (80%) and test data (20%)
set.seed(1)
rowTrain <- createDataPartition(y = college$outstate,
                                p = 0.8,
                                list = FALSE)
college <- as.data.frame(college)
```

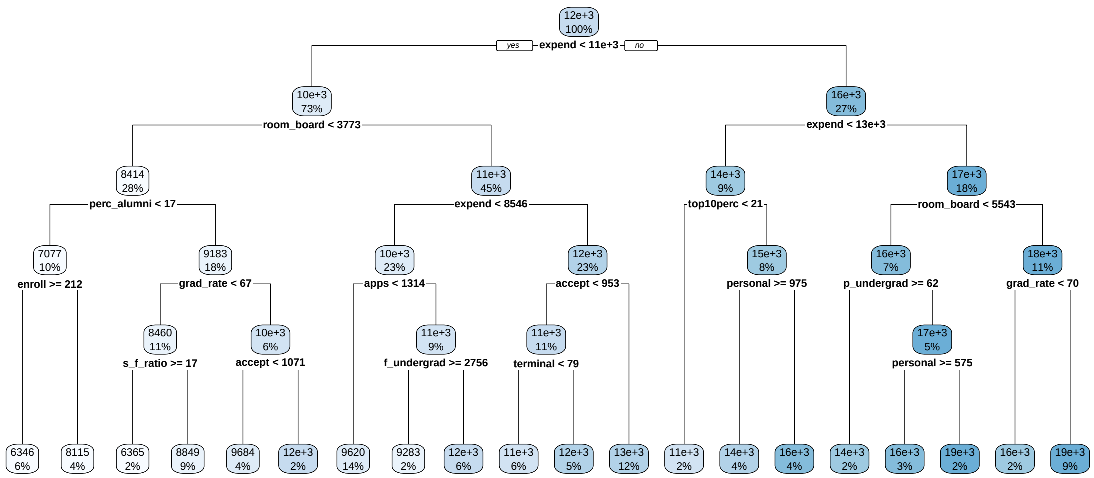
(a) Build a regression tree on the training data to predict the response.

Create a plot of the tree:

```
ctrl <- trainControl(method = "cv")
# train model
set.seed(1)
model.rpart <- train(outstate ~., data = college[rowTrain,],
                     method = "rpart",
                     tuneGrid = data.frame(cp = exp(seq(-6, -2, length = 50))),
                     trControl = ctrl)
model.rpart$bestTune
```

```
##            cp
## 8 0.004389362
```

```
# plot the tree
rpart.plot::rpart.plot(model.rpart$finalModel)
```



(b) Perform random forest on the training data. Report the variable importance and the test error.

Model Tuning

```
# Try more if possible
rf.grid <- expand.grid(mtry = 1:16,
                       splitrule = "variance",
                       min.node.size = 1:6)
set.seed(1)
rf.fit <- train(outstate ~.,
                data = college[rowTrain,],
                method = "ranger",
                tuneGrid = rf.grid,
                trControl = ctrl)

rf.fit$bestTune
```

```
##    mtry splitrule min.node.size
## 39    7  variance             3
```

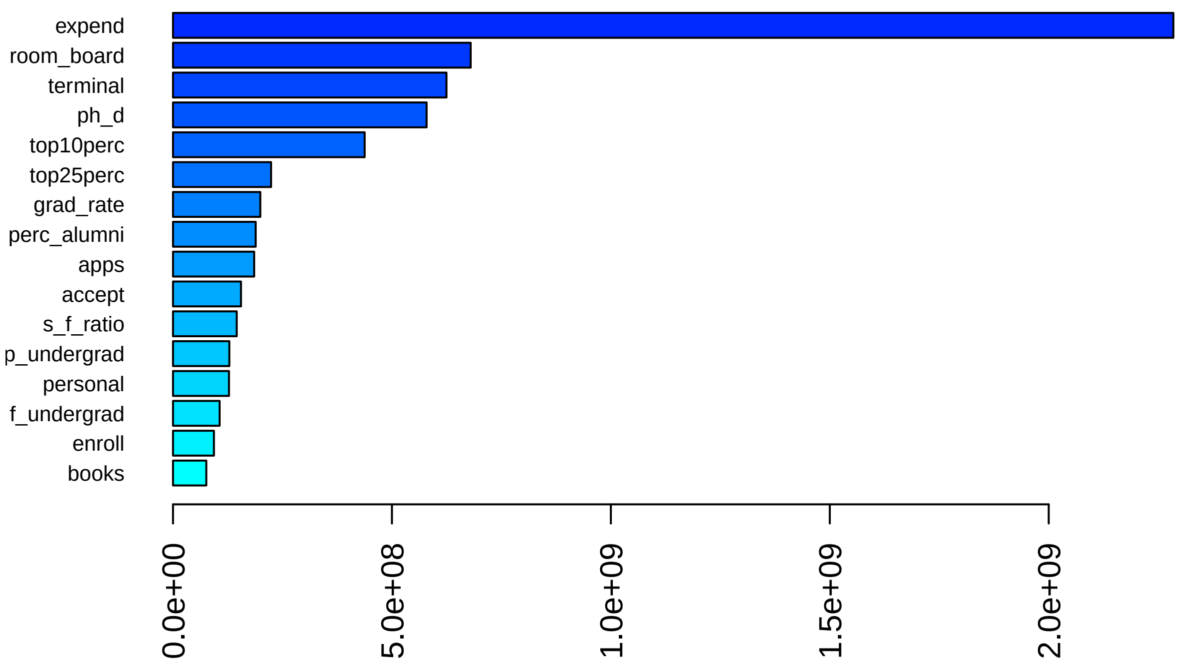
The best model selected via CV has 7 splitting variables with minimum node size of 3.

Variable Importance

The total decrease in node impurities from splitting on the variable, averaged over all trees:

```
set.seed(1)
rf.imp <- ranger(outstate ~ . ,
                 college[rowTrain,],
                 mtry = rf.fit$bestTune[[1]],
                 splitrule = "variance",
                 min.node.size = rf.fit$bestTune[[3]],
                 importance = "impurity")

barplot(sort(ranger::importance(rf.imp), decreasing = FALSE),
        las = 2, horiz = TRUE, cex.names = 0.7,
        col = colorRampPalette(colors = c("cyan","blue"))(19))
```



The most importance factor affecting the out-of-state tuition is the Instructional expenditure per student (`expend`), then it is room and board costs(`room_board`), pct. of faculty with terminal degree (`terminal`), pct. of faculty with Ph.D.'s (`ph_d`), and pct. of new students from top 10% of H.S. class (`Top10perc`).

Test error

```
pred.rf <- predict(rf.fit, newdata = college[-rowTrain,])
RMSE(pred.rf, college$outstate[-rowTrain])
```

```
## [1] 1623.252
```

The test error (RMSE) is 1623.252.

(c) Perform boosting on the training data

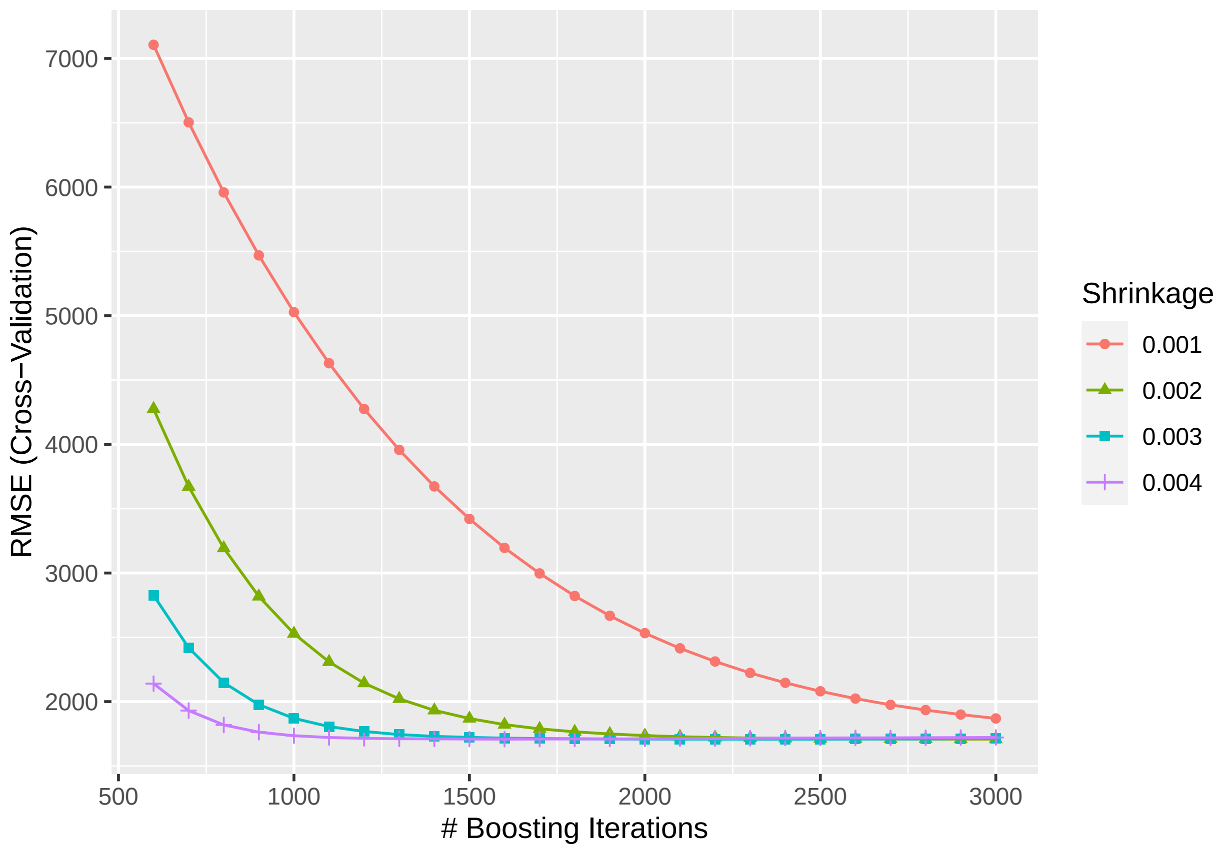
Perform Boosting via XGBoost (Extreme Gradient Boosting)

Tune by (step by step): 1. Number of Iterations and the Learning Rate (some small `eta`) 2. Maximum Depth and Minimum Child Weight (`max_depth` and `min_child_weight`) 3. Column and Row Sampling (`colsample_bytree` and `subsample`) 4. Gamma 5. Reducing the Learning Rate (increase rounds and try very small `eta`)

```r
# Tune by Number of Iterations and the Learning Rate
xgb.grid <- expand.grid(
  nrounds = seq(from = 600, to = 3000, by = 100),
  eta = c(0.001, 0.002, 0.003, 0.004),
  max_depth = 3, # best tuned max_depth
  gamma = 0.5, # best tuned gamma
  colsample_bytree = 0.6, # best tuned colsample_bytree
  min_child_weight = 3, # best tuned min_child_weight
  subsample = 0.5 # best tuned subsample
)

set.seed(1)
xgb.fit <- train(outstate ~ . ,
                 college[rowTrain,],
                 method = "xgbTree",
                 tuneGrid = xgb.grid,
                 trControl = ctrl,
                 verbose = FALSE,
                 verbosity = 0)

ggplot(xgb.fit)
```

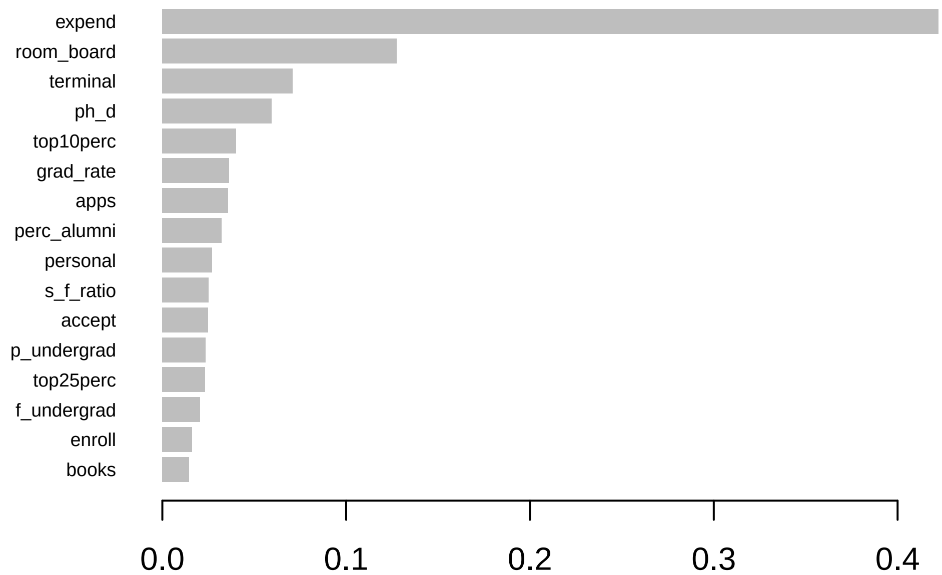



```r
xgb.fit$bestTune
```

```
##    nrounds max_depth   eta gamma colsample_bytree min_child_weight subsample
## 69    2400         3 0.003   0.5              0.6                3       0.5
```

Variable Importance

```r
xgb_imp <- xgb.importance(feature_names = xgb.fit$finalModel$feature_names,
               model = xgb.fit$finalModel)
xgb.plot.importance(xgb_imp)
```



The most importance factor affecting the out-of-state tuition is the Instructional expenditure per student (`expend`), then it is room and board costs(`room_board`), pct. of faculty with terminal degree (`terminal`), pct. of faculty with Ph.D.'s (`ph_d`), and pct. of new students from top 10% of H.S. class (`Top10perc`).

Test error

```
pred.xgb <- predict(xgb.fit, newdata = college[-rowTrain,])
RMSE(pred.xgb, college$outstate[-rowTrain])
```

```
## [1] 1600.625
```

The test error (RMSE) is 1600.625, smaller than that of the random forest model.

Classification - `OJ` data

(a) Build a classification tree using the training data, with Purchase as the response and the other variables as predictors.

```
data(OJ)
OJ <- na.omit(OJ)
OJ$Purchase <- factor(OJ$Purchase, c("CH","MM"))
```

```
set.seed(1)
rowTrain.oj <- createDataPartition(y = OJ$Purchase,
                                   p = 0.653,
                                   list = FALSE)
```

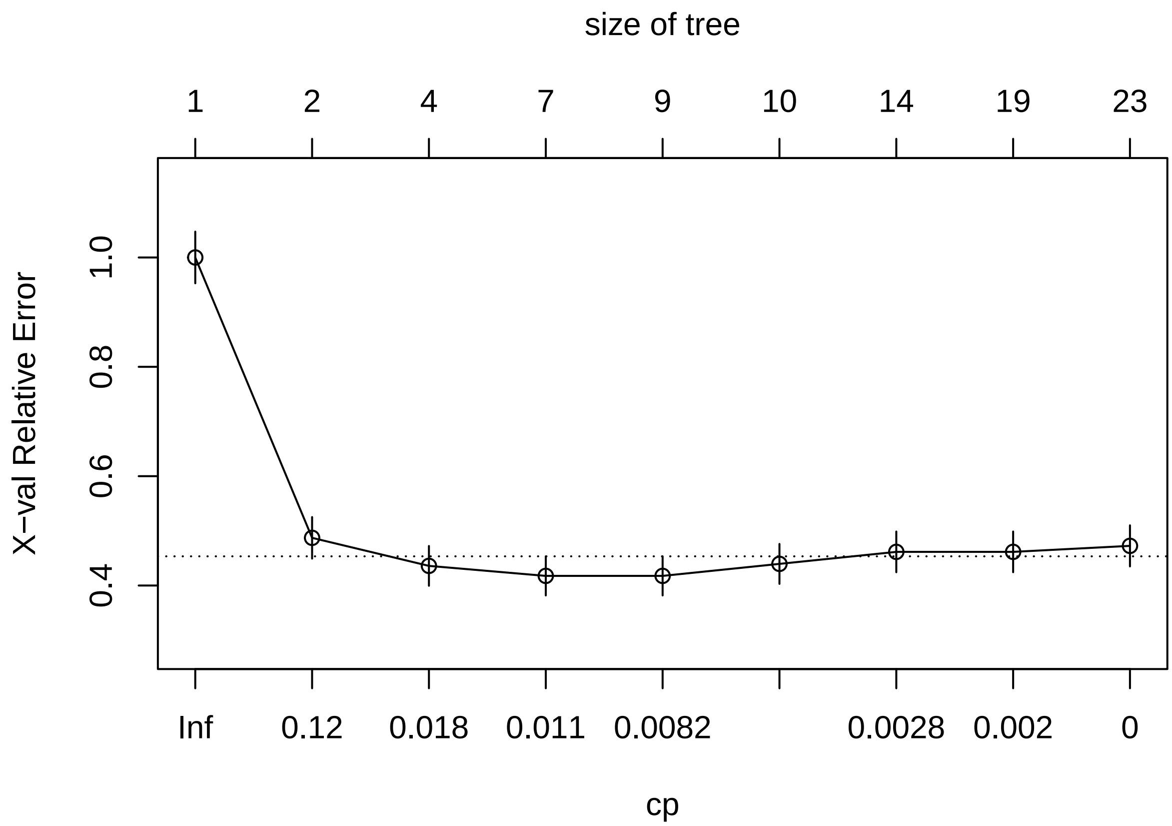
Use cross-validation to determine the tree size and create a plot of the final tree:

```
set.seed(1)
tree <- rpart(Purchase ~.,
              OJ,
              subset = rowTrain.oj,
              control = rpart.control(cp = 0))

cpTable <- printcp(tree)
```

```
##
## Classification tree:
## rpart(formula = Purchase ~ ., data = OJ, subset = rowTrain.oj,
##     control = rpart.control(cp = 0))
##
## Variables actually used in tree construction:
## [1] ListPriceDiff  LoyalCH        PriceCH        PriceDiff      SalePriceCH
## [6] SalePriceMM    STORE          StoreID        WeekofPurchase
##
## Root node error: 273/700 = 0.39
##
## n= 700
##
##          CP nsplit rel error  xerror     xstd
## 1 0.5384615      0   1.00000 1.00000 0.047270
## 2 0.0256410      1   0.46154 0.48718 0.038019
## 3 0.0122100      3   0.41026 0.43590 0.036404
## 4 0.0091575      6   0.37363 0.41758 0.035784
## 5 0.0073260      8   0.35531 0.41758 0.035784
## 6 0.0036630      9   0.34799 0.43956 0.036525
## 7 0.0021978     13   0.33333 0.46154 0.037233
## 8 0.0018315     18   0.32234 0.46154 0.037233
## 9 0.0000000     22   0.31502 0.47253 0.037575
```

```
# Size not consecutive
plotcp(tree)
```



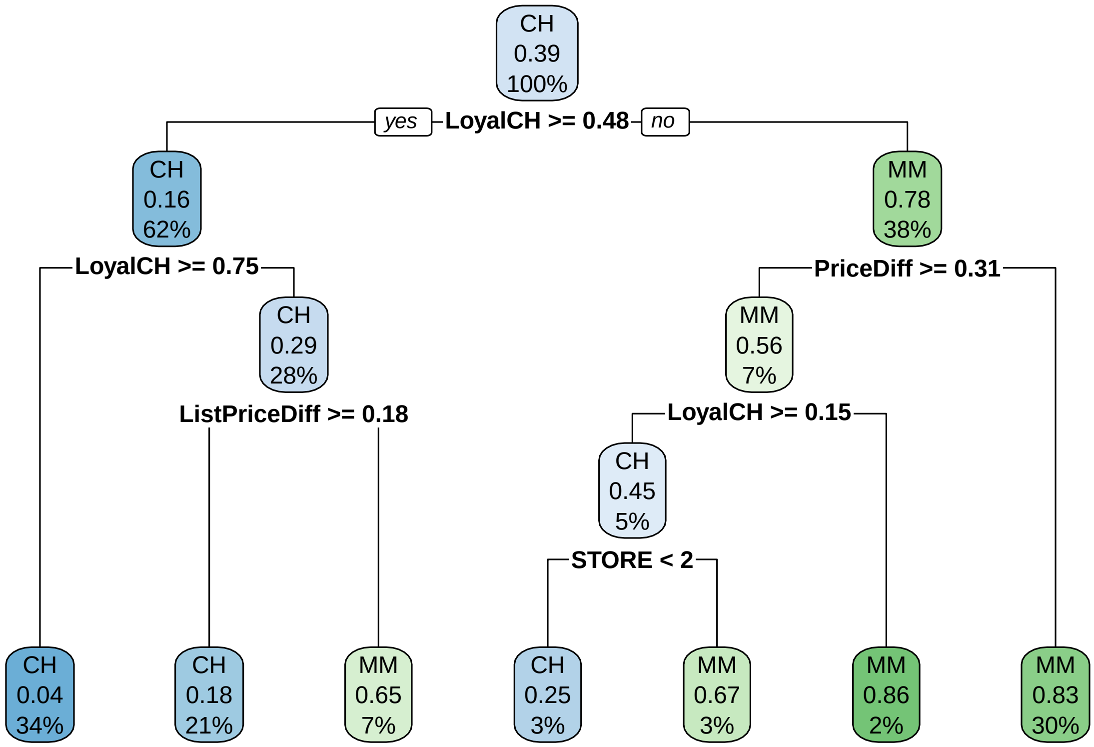

```
# size for min
minErr <- which.min(cpTable[,4])
cpTable[minErr, 1]
```

```
## [1] 0.009157509
```

The tree of 6 splits i.e. size 13 with $cp = 0.00916$ corresponds to the lowest cross-validation error. The dashed line above is the 1SE line, so a smaller tree with 3 splits i.e. size 7 has a cross-validation error below the line. Therefore, the tree size obtained by the minimum rule is different from the tree size obtained using the 1SE rule.

Final Tree

```
minErr <- which.min(cpTable[,4])
tree2 <- prune(tree, cp = cpTable[minErr, 1])
rpart.plot::rpart.plot(tree2)
```



(b) Perform boosting on the training data and report the variable importance.

```
set.seed(1)
gbmA.grid <- expand.grid(n.trees = c(2000,3000,4000,5000),
                         interaction.depth = 1:6,
                         shrinkage = c(0.0005,0.001,0.002),
                         n.minobsinnode = 1)
set.seed(1)
gbmA.fit <- train(Purchase ~ . ,
                  OJ,
                  subset = rowTrain.oj,
                  tuneGrid = gbmA.grid,
                  trControl = ctrl,
                  method = "gbm",
                  distribution = "adaboost",
                  metric = "ROC",
                  verbose = FALSE)

ggplot(gbmA.fit, highlight = TRUE)
```

Variale Importance

```
summary(gbmA.fit$finalModel, las = 2, cBars = 16, cex.names = 0.6)
```

Test Error Rate

```
gbmA.pred <- predict(gbmA.fit, newdata = OJ[-rowTrain.oj,], type = "prob")[,1]
```