p8106_hw4

Hao Zheng

4/12/2022

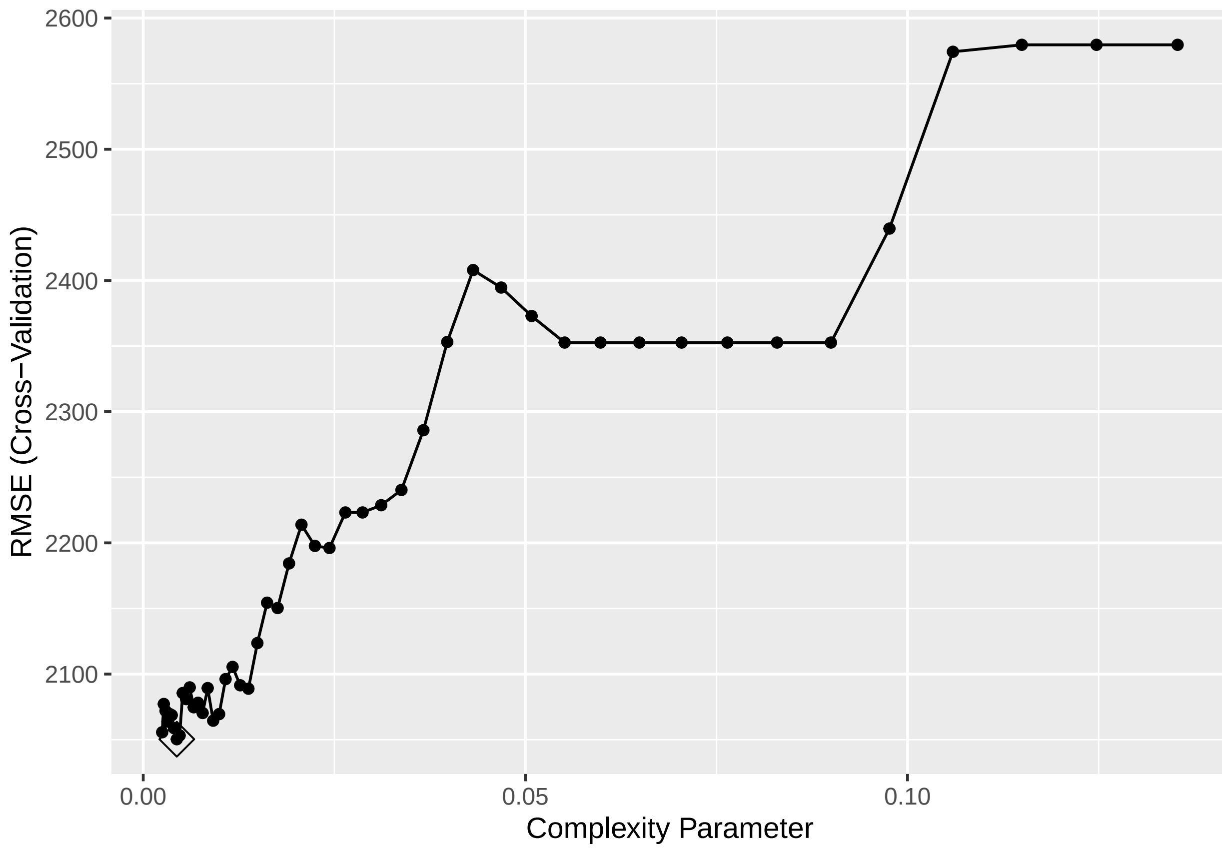
Problem 1 College Data

```
# Data Import
college_data =
  read.csv("./College.csv") %>%
  na.omit() %>%
  janitor::clean_names() %>%
  select(-college)

# Data Partition
set.seed(2022)
trRows <- createDataPartition(college_data$outstate,
                              p =.8,
                              list = F)

ctrl1 <- trainControl(method = "cv")
ctrl2 <- trainControl(method = "cv",
                      classProbs = TRUE,
                      summaryFunction = twoClassSummary)
ctrl3 <- trainControl(method = "cv",
                      classProbs = TRUE,
                      summaryFunction = twoClassSummary,
                      selectionFunction = "oneSE")
```

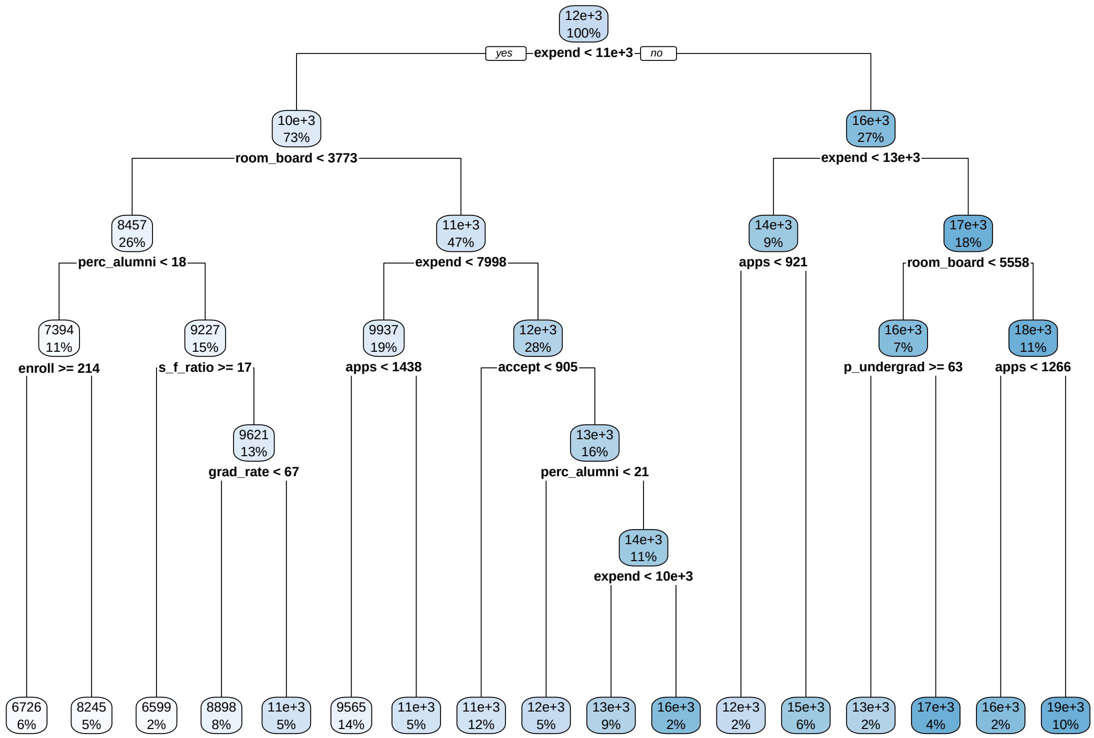
Question a) Build a Regression Tree

Build a regression tree on training data using caret based on cp value.

```
set.seed(2022)
rpart.fit <- train(outstate ~.,
                   college_data[trRows, ],
                   method = "rpart",
                   tuneGrid = data.frame(cp = exp(seq(-6,-2,length = 50))),
                   trControl = ctrl1)

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



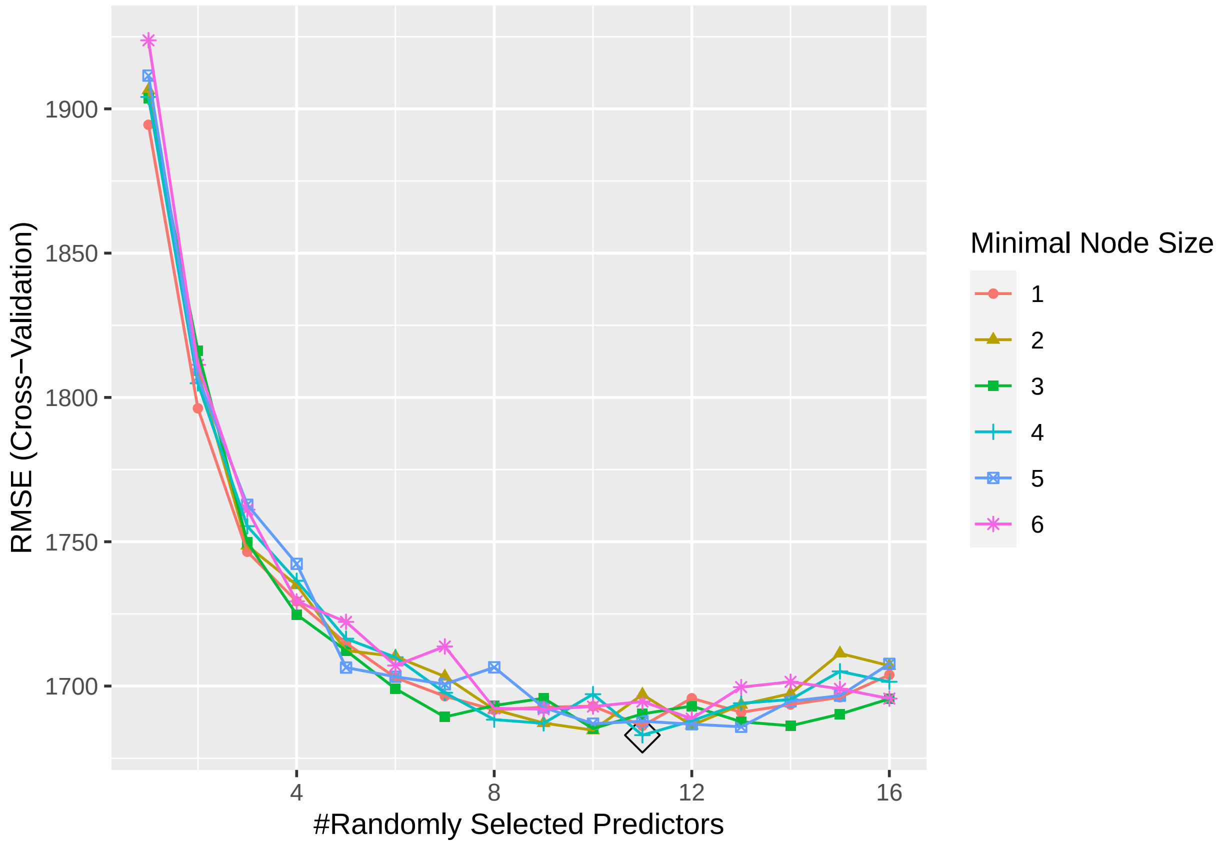

```
rpart.plot(rpart.fit$finalModel)
```



The pruned tree based on the optimal cp value is plotted as above. It's quite complex with 17 terminal nodes and 16 splits.

Question b) Random Forest

```r
rf.grid <- expand.grid(mtry = 1:16,
                       splitrule = "variance",
                       min.node.size = 1:6)

set.seed(2022)
rf.fit <- train(outstate ~ .,
                college_data[trRows,],
                method = "ranger",
                tuneGrid = rf.grid,
                trControl = ctrl1)

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

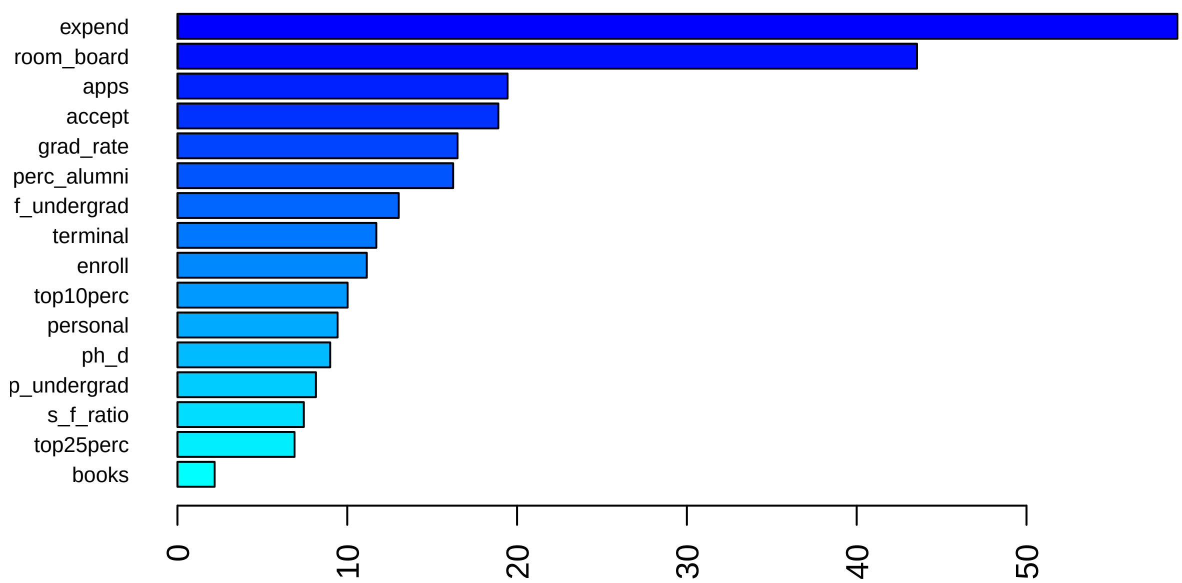



```
rf.fit$bestTune
```

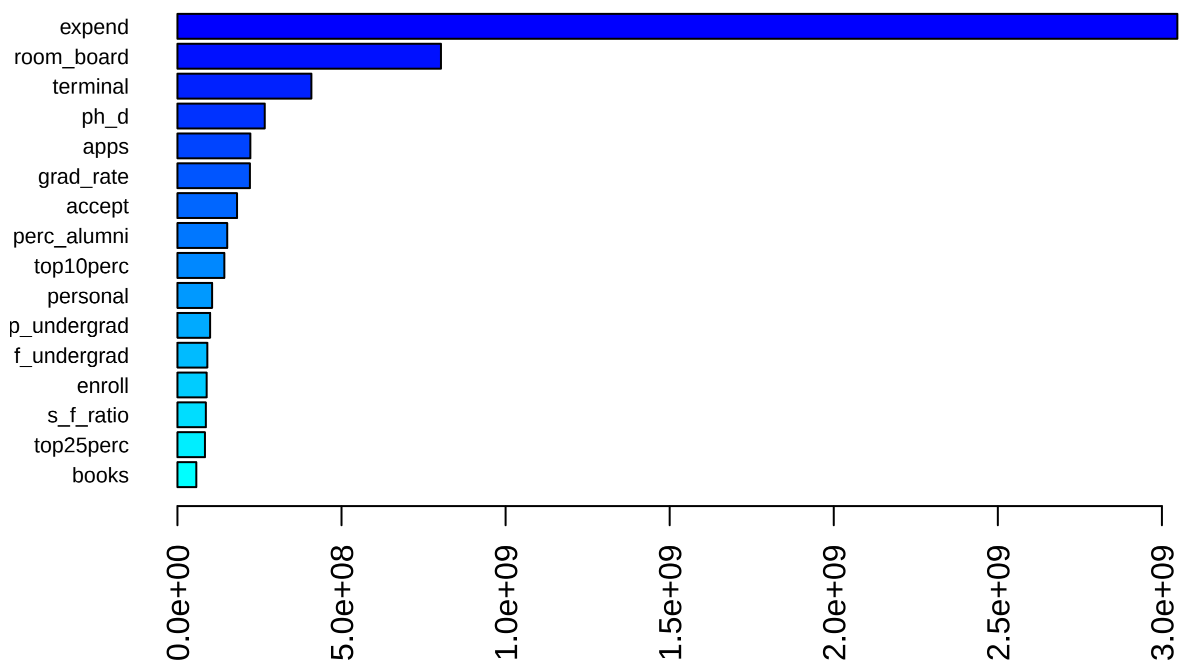
```
##    mtry splitrule min.node.size
## 64   11  variance             4
```

Using ranger method, we perform Random Forest algorithm with minimum node size 4 and 11 selected predictors.

```
# variable importance using permutation methods
set.seed(2022)
rf.final.per = ranger(outstate ~ .,
                      college_data[trRows,],
                      mtry = rf.fit$bestTune[[1]],
                      splitrule = "variance",
                      min.node.size = rf.fit$bestTune[[3]],
                      importance = "permutation",
                      scale.permutation.importance = TRUE)

barplot(sort(ranger::importance(rf.final.per), decreasing = FALSE),
        las = 2, horiz = TRUE, cex.names = 0.7,
        col = colorRampPalette(colors = c("cyan", "blue"))(16))
```




```
# variable importance using impurity methods
set.seed(2022)

rf.final.imp <- ranger(outstate ~ .,
                        college_data[trRows,],
                        mtry = rf.fit$bestTune[[1]],
                        splitrule = "variance",
                        min.node.size = rf.fit$bestTune[[3]],
                        importance = "impurity")

barplot(sort(ranger::importance(rf.final.imp), decreasing = FALSE),
        las = 2, horiz = TRUE, cex.names = 0.7,
        col = colorRampPalette(colors = c("cyan", "blue"))(16))
```




```
# Test error
pred.rf <- predict(rf.fit, newdata = college_data[-trRows,])
RMSE(pred.rf, college_data$outstate[-trRows])
```

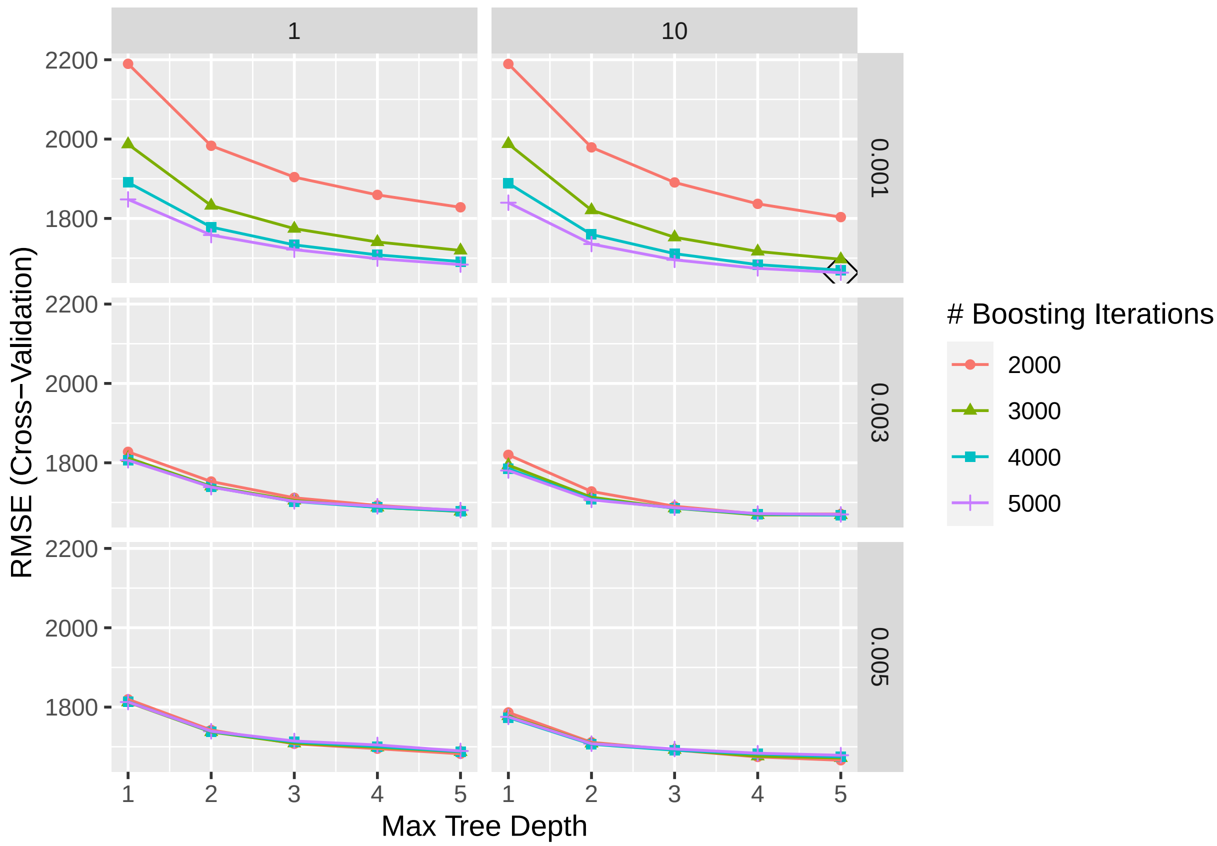
```
## [1] 1960.044
```

Using the permutation method, the most important predictors are `expend` and `room_board`. Using the impurity method, the predictor with the most importance are the same with the previous method. The test error for the random forest model is 1960.044.

Question c) Boosting

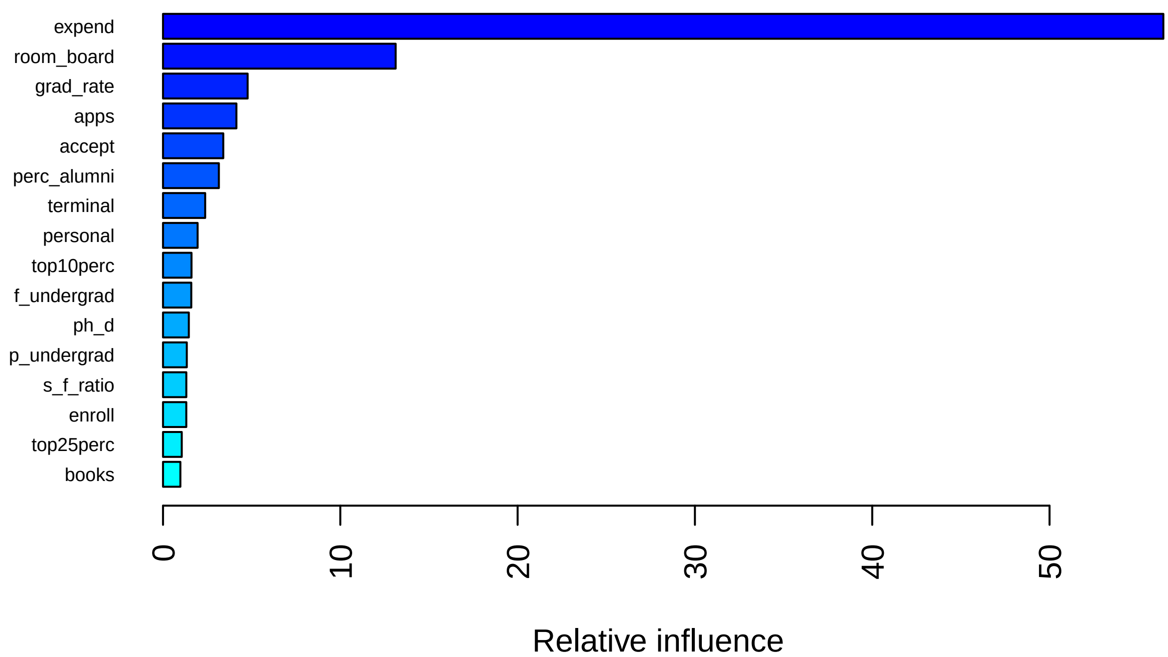
```
# gbm
gbm.grid <- expand.grid(n.trees = c(2000,3000,4000,5000),
                        interaction.depth = 1:5,
                        shrinkage = c(0.001,0.003,0.005),
                        n.minobsinnode = c(1,10))

set.seed(2022)
gbm.fit <- train(outstate ~.,
                 college_data[trRows,],
                 method = "gbm",
                 tuneGrid = gbm.grid,
```

```r
                  trControl = ctrl1,
                  verbose = FALSE)

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



We use the gradient boosting method implemented with gbm in caret package.

```r
# variable importance
summary(gbm.fit$finalModel, las = 2, cBars = 16, cex.names = 0.6)
```




```
##                     var    rel.inf
## expend           expend 56.4193398
## room_board   room_board 13.1211367
## grad_rate     grad_rate  4.7728099
## apps               apps  4.1391522
## accept           accept  3.3994923
## perc_alumni perc_alumni  3.1507046
## terminal       terminal  2.3784489
## personal       personal  1.9518607
## top10perc     top10perc  1.6070151
## f_undergrad f_undergrad  1.5954492
## ph_d               ph_d  1.4569400
## p_undergrad p_undergrad  1.3428155
## s_f_ratio     s_f_ratio  1.3157272
## enroll           enroll  1.3128713
## top25perc     top25perc  1.0573210
## books             books  0.9789158
```

```r
# Test error
pred.gbm <- predict(gbm.fit, newdata = college_data[-trRows,])
RMSE(pred.gbm, college_data$outstate[-trRows])
```

```
## [1] 1893.407
```

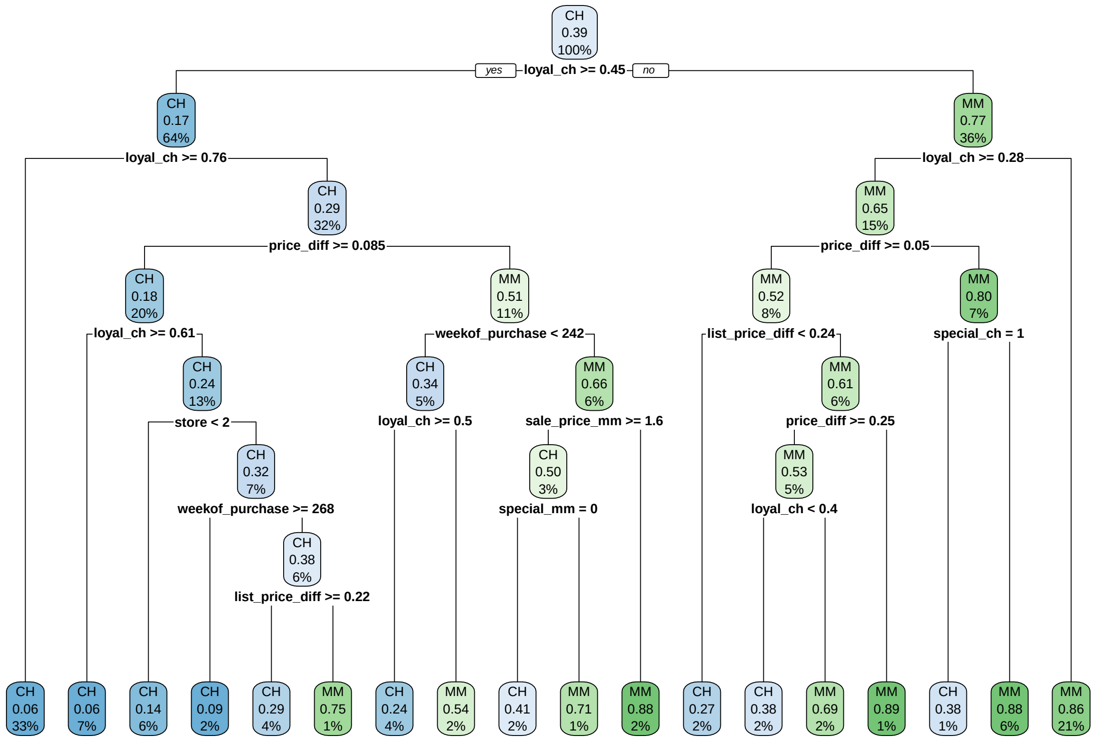
The most important variables for gradient boosting are still `expend` and `room_board`. The test error for boosting is 1893.407, which is smaller than the test error for random forest.

Problem 2 OJ Data

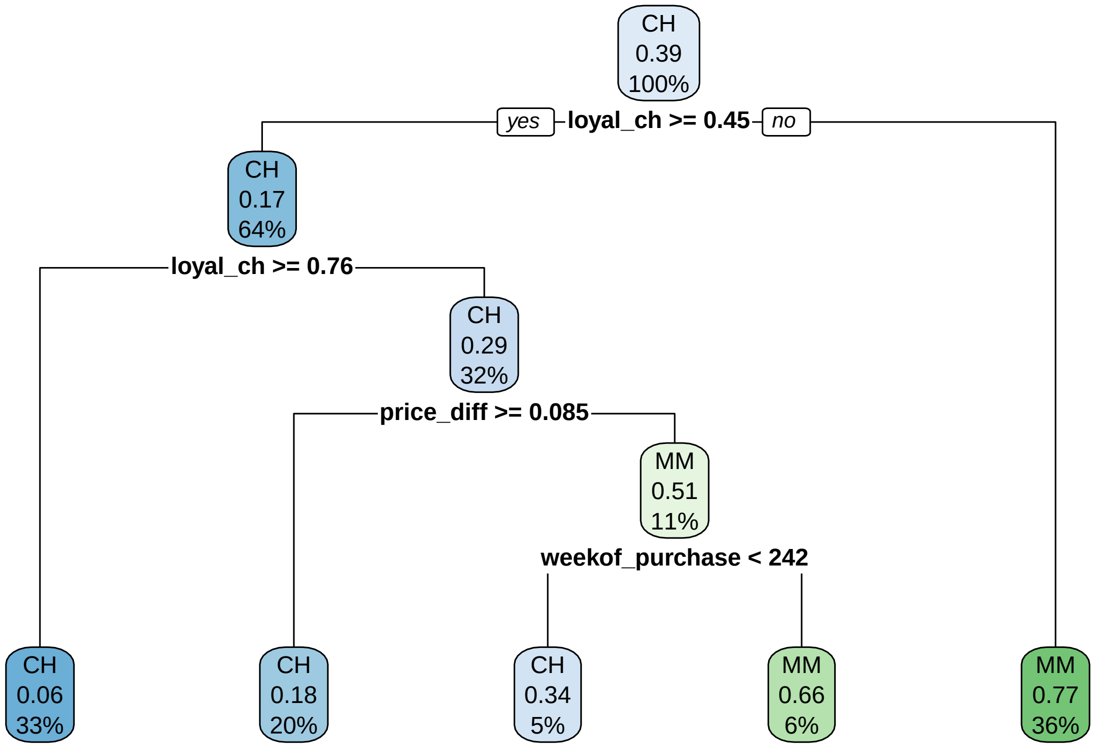
```
# Data
OJ_data =
  OJ %>%
  na.omit() %>%
  janitor::clean_names()

set.seed(2022)

rowTrain <- createDataPartition(y = OJ_data$purchase,
                                p = 700/1070,
                                list = FALSE)
```

Question a) Classification Tree

```
# classification selected using min MSE rule using CART
set.seed(2022)

rpart.fit2 = train(purchase ~ . ,
                   OJ_data,
                   subset = rowTrain,
                   method = "rpart",
                   tuneGrid = data.frame(cp = exp(seq(-6,-4,len = 100))),
                   trControl = ctrl2,
                   metric = "ROC")

rpart.plot(rpart.fit2$finalModel)
```



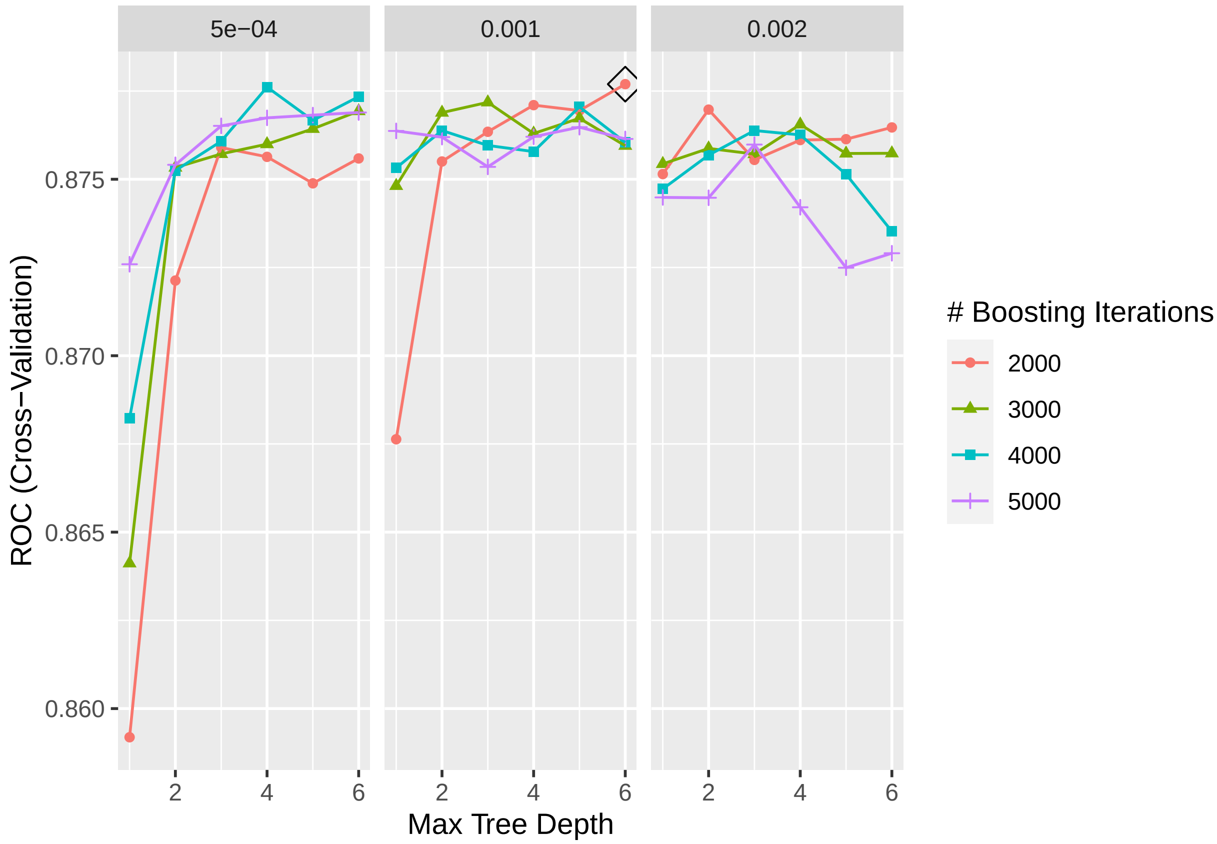

```r
# Classification tree selected using 1SE rule
set.seed(2022)

rpart.fit2.1se = train(purchase ~ . ,
                   OJ_data,
                   subset = rowTrain,
                   method = "rpart",
                   tuneGrid = data.frame(cp = exp(seq(-6,-4,len = 100))),
                   trControl = ctrl3,
                   metric = "ROC")

rpart.plot(rpart.fit2.1se$finalModel)
```



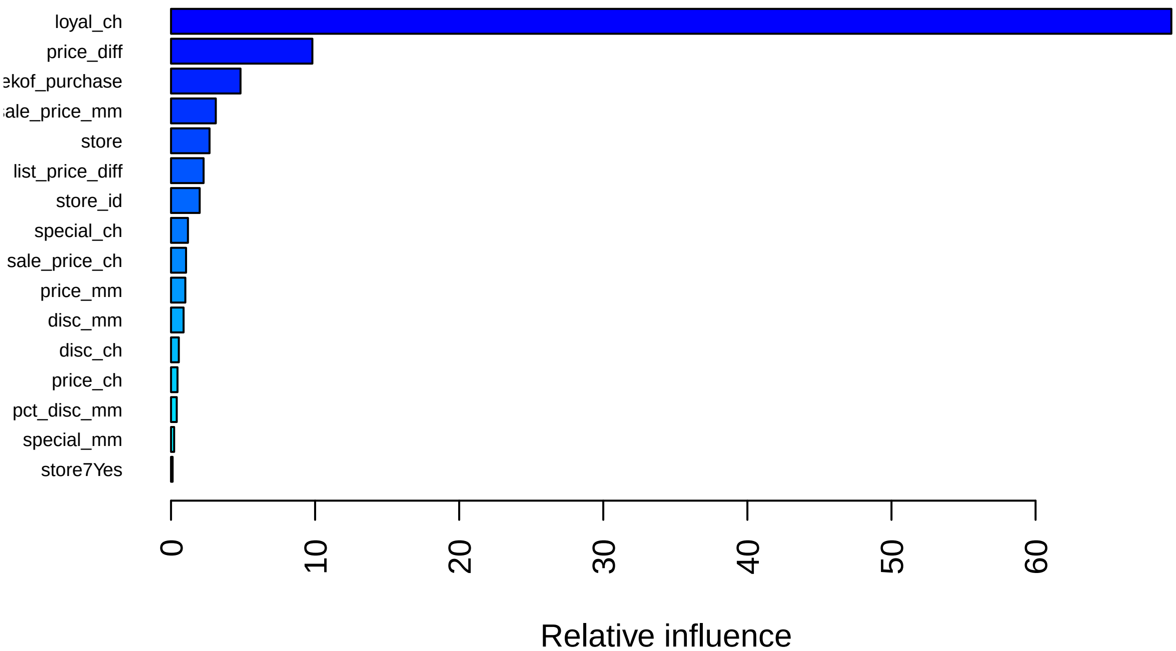
Here, we use rpart to build classification tree in order to predict response variable `purchase`. The tree size with the lowest cross-validation error is 18 with 17 splits. It's not the same with the tree size obtained using 1SE rule, which is 5 with only 4 splits.

Question b) Boosting

```
set.seed(2022)

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)

gbmA.fit <- train(purchase ~.,
                  OJ_data,
                  subset = rowTrain,
                  tuneGrid = gbmA.grid,
                  trControl = ctrl2,
                  method = "gbm",
                  distribution = "adaboost",
                  metric = "ROC",
                  verbose = FALSE)

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




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




```
##                              var    rel.inf
## loyal_ch                loyal_ch 69.4239421
## price_diff            price_diff  9.8103680
## weekof_purchase weekof_purchase  4.8237490
## sale_price_mm      sale_price_mm  3.1084009
## store                      store  2.6730398
## list_price_diff list_price_diff  2.2606130
## store_id                store_id  1.9899858
## special_ch            special_ch  1.1747910
## sale_price_ch      sale_price_ch  1.0434147
## price_mm                price_mm  0.9947896
## disc_mm                  disc_mm  0.8723593
## disc_ch                  disc_ch  0.5425042
## price_ch                price_ch  0.4499299
## pct_disc_mm          pct_disc_mm  0.3978840
## special_mm            special_mm  0.2191461
## store7Yes              store7Yes  0.1114350
## pct_disc_ch          pct_disc_ch  0.1036475
```

```
# RMSE
pred.gbm = predict(gbmA.fit, newdata = OJ_data[-rowTrain,])
test.error = mean(pred.gbm != OJ_data$purchase[-rowTrain]);test.error
```

```
## [1] 0.1490515
```

The most important predictor is `loyal_ch`, the second one is `price_diff`. The test error rate is 14.905%.