p8106_HW4_qz3366

Qing Zhou

2023-04-20

Question 1

Data preparation

In this exercise, we will build tree-based models using the College data. The dataset contains statistics for 565 US Colleges from a previous issue of US News and World Report. The response variable is the out-of-state tuition (Outstate).

```
# data import
college_df =
  read.csv("data/College.csv") %>%
  na.omit() %>%
  janitor::clean_names() %>%
  relocate("outstate", .after = "grad_rate") %>%
  select(-college)
```

Partition the dataset into two parts: training data (80%) and test data (20%).

```
set.seed(1)
# data partition
trainRows <- createDataPartition(y = college_df$outstate, p = 0.8, list = FALSE)

# training data
college_train = college_df[trainRows, ]

# testing data
college_test = college_df[-trainRows, ]

# create cross-validation objects
ctrl1 <- trainControl(method = "cv")
# for classification tree under the minimal MSE rule
ctrl2 <- trainControl(method = "cv",
                      classProbs = TRUE,
                      summaryFunction = twoClassSummary)
# for classification tree under the 1SE rule
ctrl3 <- trainControl(method = "cv",
                      classProbs = TRUE,
                      summaryFunction = twoClassSummary,
                      selectionFunction = "oneSE")
```

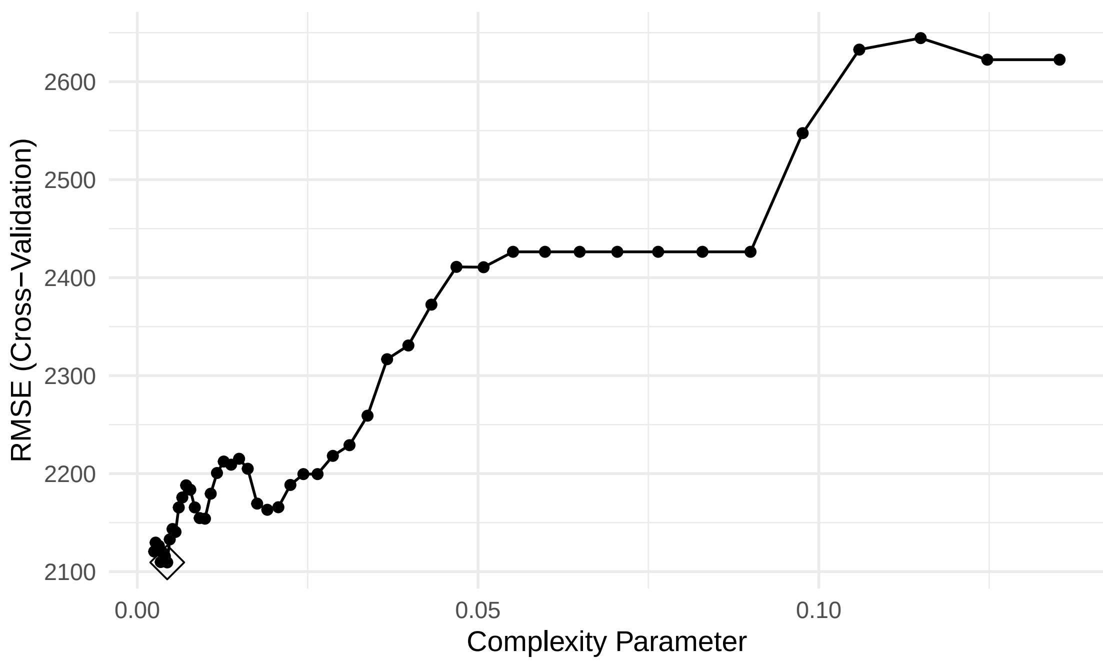
(a) Regression tree

Build a regression tree on the training data to predict the response. Create a plot of the tree.

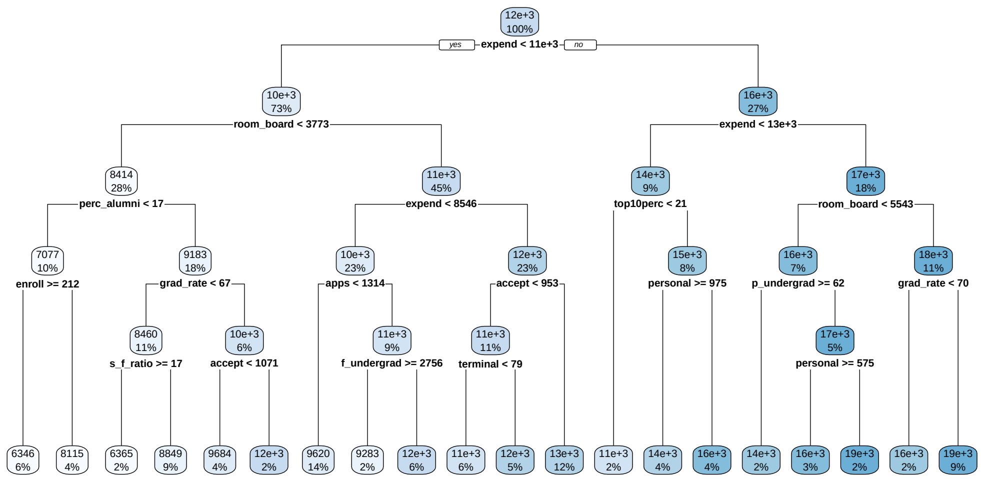
```
set.seed(1)

# build a regression tree on the training data
rpart.fit <- train(outstate ~ . ,
                   data = college_train,
                   method = "rpart",
                   tuneGrid = data.frame(cp = exp(seq(-6,-2, length = 50))),
                   trControl = ctrl1)
rpart.fit$bestTune
```

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

```
# plot of the complexity parameter
ggplot(rpart.fit, highlight = TRUE)
```




```
# create a plot of the tree
rpart.plot(rpart.fit$finalModel)
```

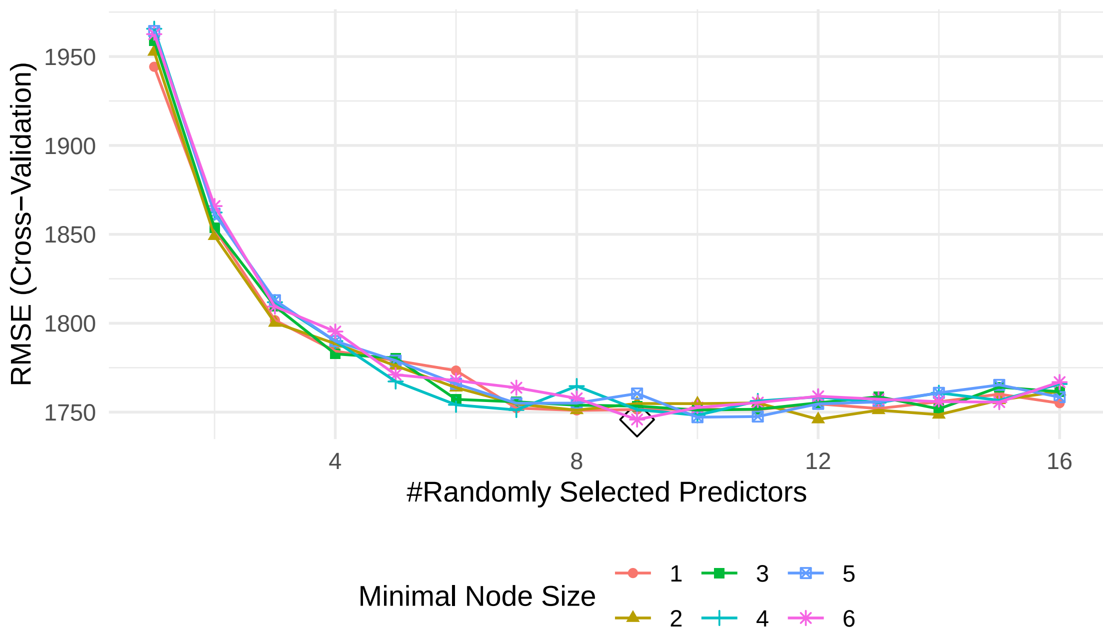


- The root node is `expend` over or under 11K.
- The optimal cp is 0.004389362.
- The pruned tree based on the optimal cp value is plotted as above. It's quite complicated with 20 terminal nodes and 19 splits.

(b) Random forest

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

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

set.seed(1)
# train a random forest model on the training data
rf.fit <- train(outstate ~ .,
               data = college_train,
                method = "ranger",
                tuneGrid = rf.grid,
                trControl = ctrl1)

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

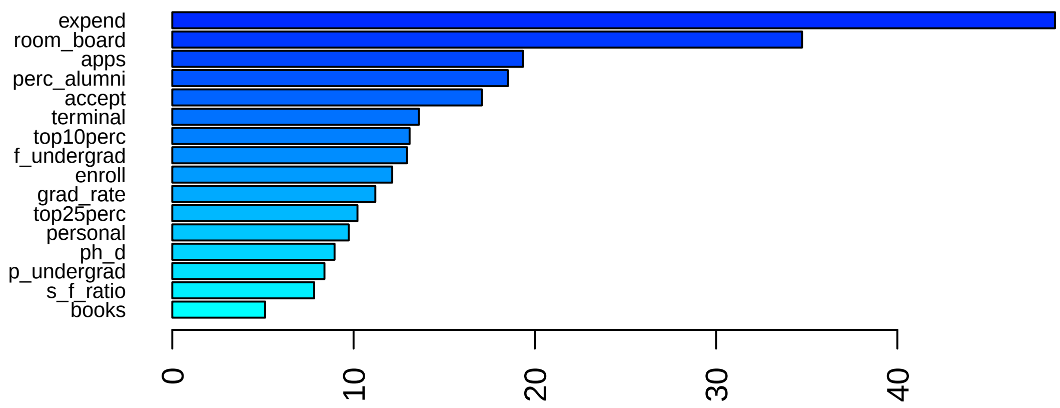



```
rf.fit$bestTune
```

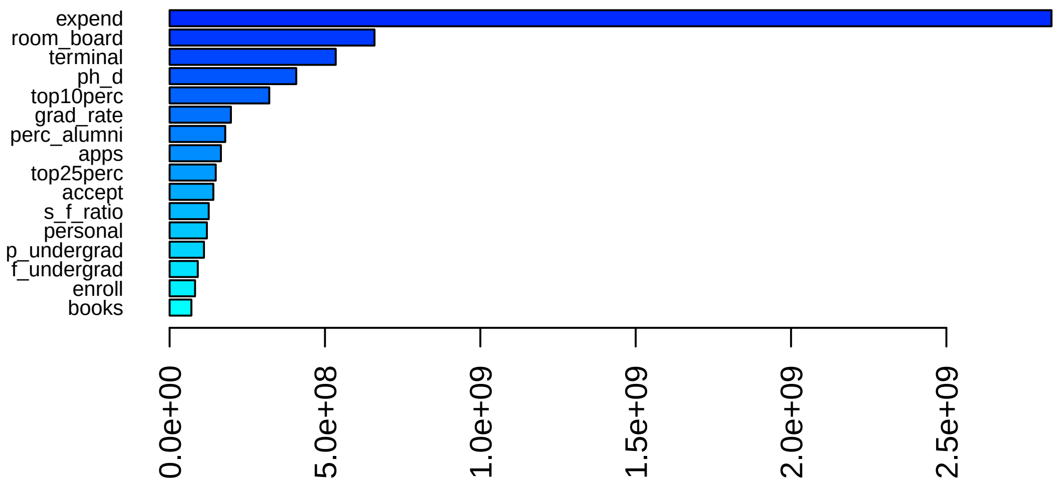
```
##    mtry splitrule min.node.size
## 54    9  variance             6
```

- Using ranger method, we perform Random Forest algorithm with minimum node size 6 and 9 selected predictors.
- Once a random forest model is trained, it is common to inquire about the variables that have the most predictive ability. Variables with a high degree of importance are instrumental in determining the outcome and their values can significantly affect the outcome. On the other hand, variables with low importance may be excluded from the model, which can simplify the model and improve its efficiency in terms of fitting and prediction.

```
# variable importance using permutation methods
set.seed(1)
rf.perm = ranger(outstate ~ .,
                      data = college_train,
                      mtry = rf.fit$bestTune[[1]],
                      splitrule = "variance",
                      min.node.size = rf.fit$bestTune[[3]],
                      importance = "permutation",
                      scale.permutation.importance = TRUE)
# report variable importance
barplot(sort(ranger::importance(rf.perm), decreasing = FALSE),
        las = 2, horiz = TRUE, cex.names = 0.7,
        col = colorRampPalette(colors = c("cyan", "blue"))(19))
```




```r
# variable importance using impurity methods
set.seed(1)
rf.impu <- ranger(outstate ~ .,
                  data = college_train,
                  mtry = rf.fit$bestTune[[1]],
                  splitrule = "variance",
                  min.node.size = rf.fit$bestTune[[3]],
                  importance = "impurity")
# report variable importance
barplot(sort(ranger::importance(rf.impu), decreasing = FALSE),
        las = 2, horiz = TRUE, cex.names = 0.7,
        col = colorRampPalette(colors = c("cyan", "blue"))(19))
```



- Calculate and graph variable importance using permutation and impurity metrics.
- The model indicated that the variables `expend` and `room-board` had the highest predictive power and their values were the most significant in determining the out-of-state tuition cost (outstate). This suggests that these variables play a crucial role in influencing the outstate tuition.

```r
# test error
pred.rf <- predict(rf.fit, newdata = college_test)
RMSE(pred.rf, college_test$outstate)
```

```
## [1] 1651.307
```

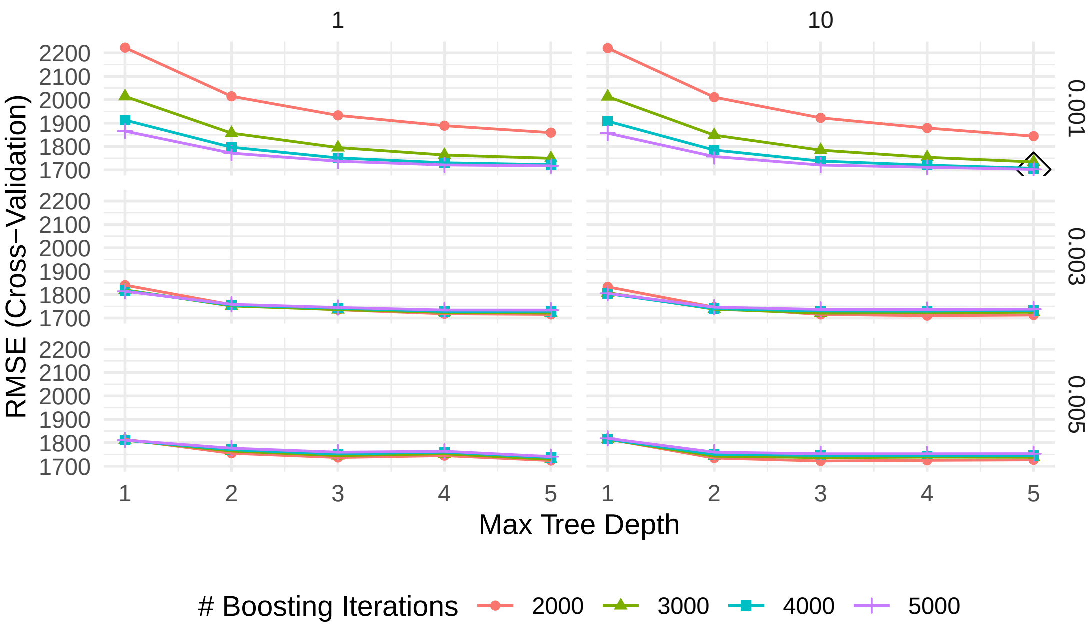
- The test error of the model is 1651.307

(c) Boosting

Perform boosting on the training data. Report the variable importance and the test error.

```r
# train model using gbm with grid of tuning parameters
bst.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(1)
bst.fit <- train(outstate ~.,
                 data = college_train,
                 method = "gbm",
                 tuneGrid = bst.grid,
                 trControl = ctrl1,
```

```
                  verbose = FALSE)

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

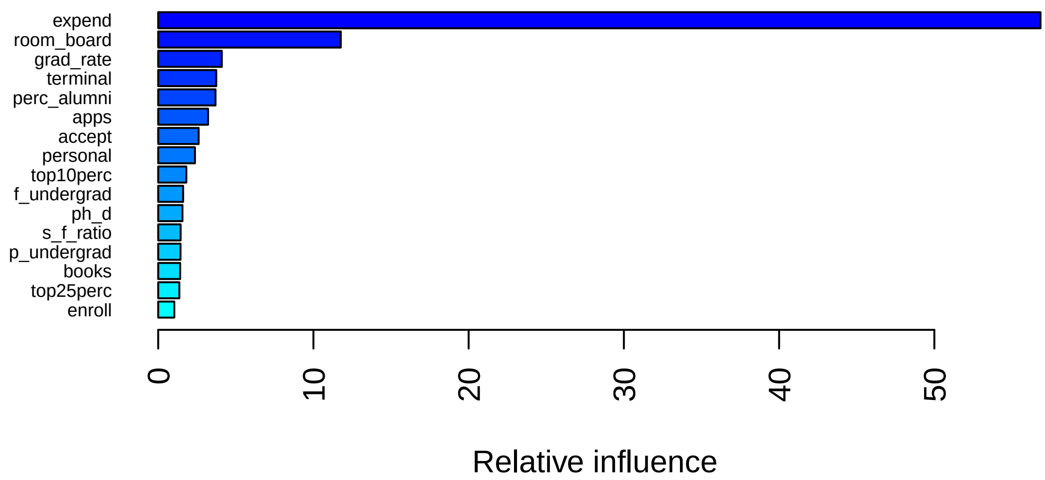



```
bst.fit$bestTune
```

```
##    n.trees interaction.depth shrinkage n.minobsinnode
## 40    5000                 5     0.001             10
```

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

```
# variable importance
summary(bst.fit$finalModel, las = 2, cBars = 19, cex.names = 0.6)
```




```
##                     var   rel.inf
## expend           expend 56.839618
## room_board   room_board 11.763945
## grad_rate     grad_rate  4.097538
## terminal       terminal  3.740792
## perc_alumni perc_alumni  3.692448
## apps               apps  3.212812
## accept           accept  2.606760
## personal       personal  2.368107
## top10perc     top10perc  1.812568
## f_undergrad f_undergrad  1.605638
## ph_d               ph_d  1.562622
## s_f_ratio     s_f_ratio  1.445236
## p_undergrad p_undergrad  1.436907
## books             books  1.415833
## top25perc     top25perc  1.361722
## enroll           enroll  1.037453
```

```r
# test error
pred.bst <- predict(bst.fit, newdata = college_test)
RMSE(pred.bst, college_test$outstate)
```

```
## [1] 1620.551
```

- The most important variables for gradient boosting are still `expend` and `room_board`. Other important variables include `grad_rate` and `terminal`. The test error for boosting is 1620.551, smaller than the test error for random forest.

Question 2

This problem involves the OJ data in the ISLR package. The data contains 1070 purchases where the customers either purchased Citrus Hill or Minute Maid Orange Juice. A number of characteristics of customers and products are recorded.

Data preparation

```
# data import
OJ =
  OJ %>%
  na.omit() %>%
  janitor::clean_names()
OJ$purchase <- factor(OJ$purchase, c("CH", "MM"))
```

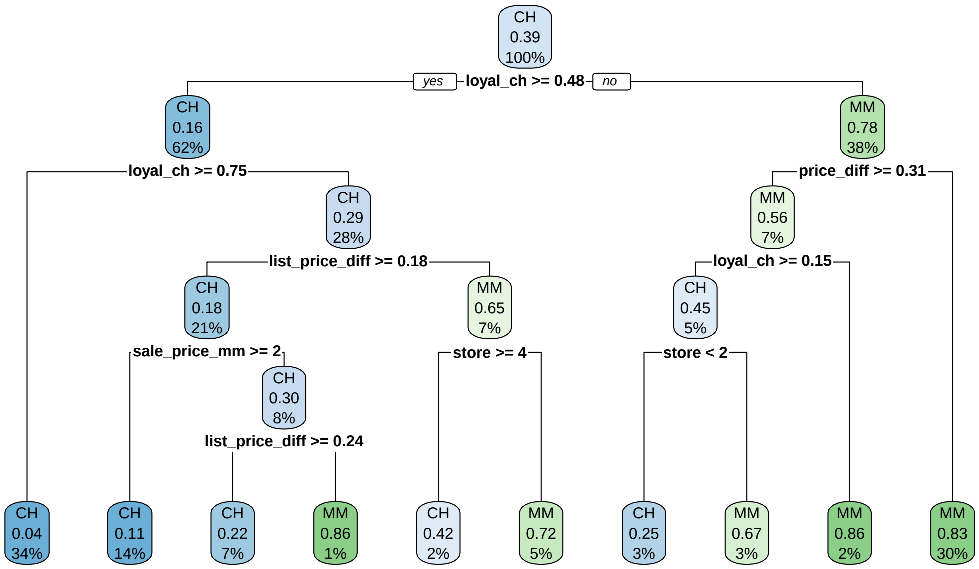
Create a training set containing a random sample of 700 observations, and a test set containing the remaining observations.

```
set.seed(1)
# data partition
trainRows2 <- createDataPartition(OJ$purchase,
                                  p = (699/1070),
                                  list = F)
```

(a) Classification Tree

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

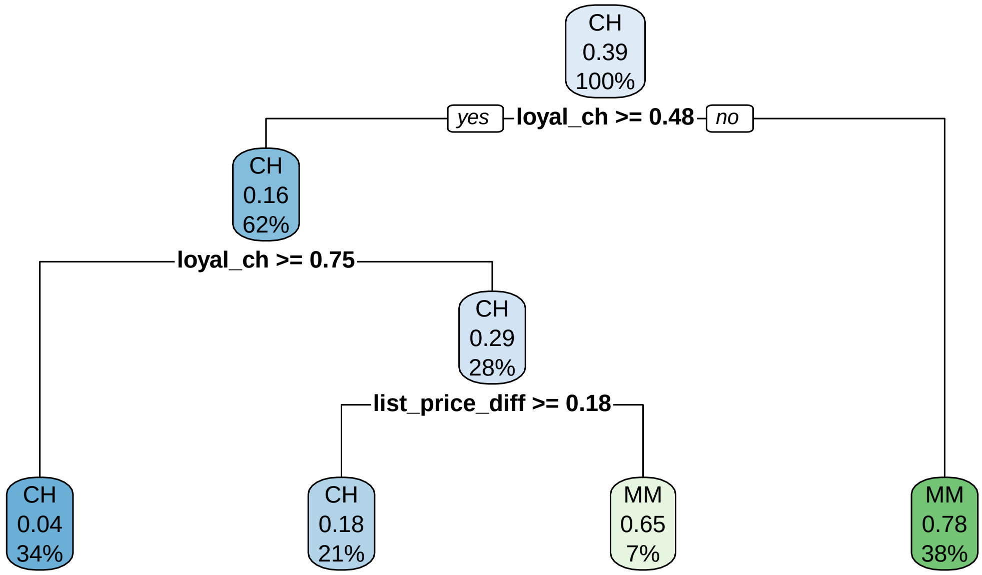
```
# classification selected by min MSE rule using CART
set.seed(1)

rpart.fit2 = train(purchase ~ . ,
                   OJ,
                   subset = trainRows2,
                   method = "rpart",
                   tuneGrid = data.frame(cp = exp(seq(-6,-4,len = 100))),
                   trControl = ctrl2,
                   metric = "ROC")

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




```r
rpart.fit2$bestTune$cp # report the best cp value
```

```
## [1] 0.00602938
```

```r
# classification tree selected by 1SE rule
set.seed(1)

rpart.fit2.1se = train(purchase ~ . ,
                    OJ,
                    subset = trainRows2,
                    method = "rpart",
                    tuneGrid = data.frame(cp = exp(seq(-6,-4,len = 100))),
                    trControl = ctrl3,
                    metric = "ROC")

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




```
rpart.fit2.1se$bestTune$cp # report the best cp value
```

```
## [1] 0.01831564
```

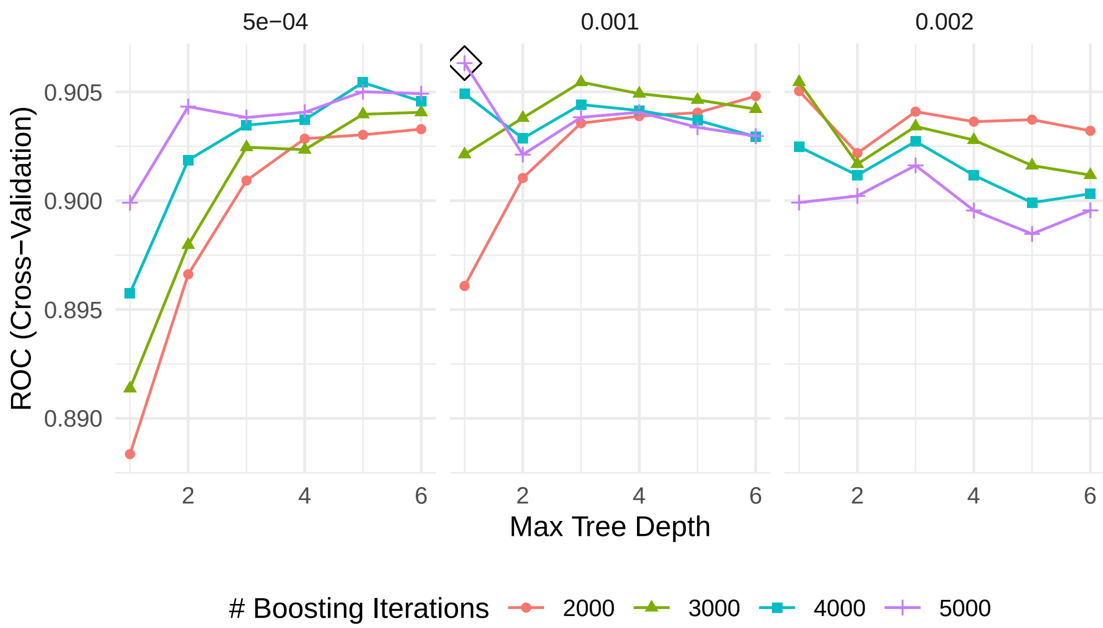
- We use rpart to build classification tree to predict the outcome `purchase`. The tree size with the lowest cross-validation error is 10 with 9 splits. It's DIFFERENT from the tree size obtained using 1SE rule, which is 4 with 3 splits. The latter is smaller.
- The tree with the lowest cross-validation error has cp = 0.00602938, while the tree obtained using the 1 SE rule has cp = 0.01831564.

(b) Boosting

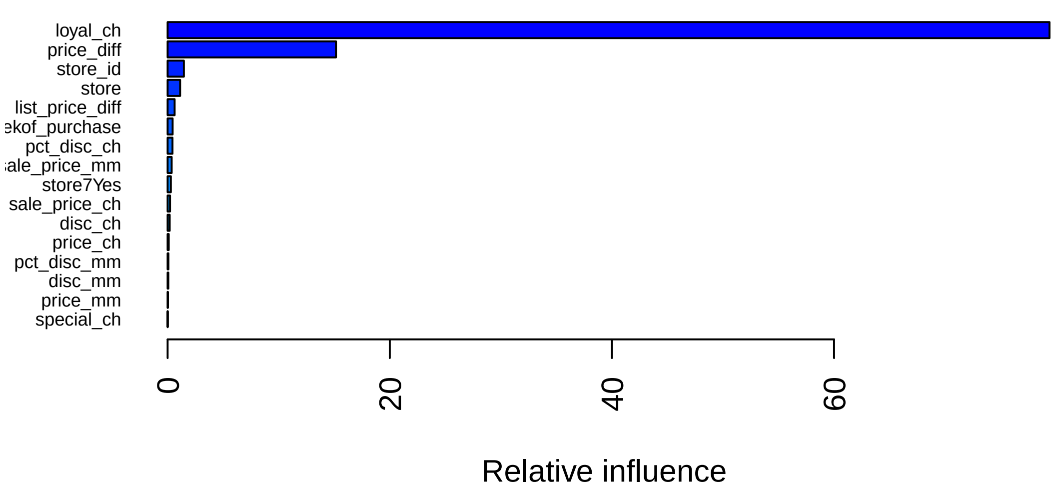
Perform boosting on the training data and report the variable importance and the test error rate.

```
set.seed(1)

gbm2.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)

gbm2.fit <- train(purchase ~.,
                  OJ,
                  subset = trainRows2,
                  tuneGrid = gbm2.grid,
                  trControl = ctrl2,
                  method = "gbm",
                  distribution = "adaboost",
                  metric = "ROC",
```

```
                    verbose = FALSE)

ggplot(gbm2.fit, highlight = TRUE)
```




```
# variance importance
summary(gbm2.fit$finalModel, las = 2, cBars = 16, cex.names = 0.6)
```




```
##                                var      rel.inf
```

```
## loyal_ch               loyal_ch 79.396857881
## price_diff           price_diff 15.158290356
## store_id               store_id  1.461757174
## store                     store  1.126546734
## list_price_diff list_price_diff  0.631757007
## weekof_purchase weekof_purchase  0.450817687
## pct_disc_ch         pct_disc_ch  0.440135026
## sale_price_mm     sale_price_mm  0.364518842
## store7Yes             store7Yes  0.290066119
## sale_price_ch     sale_price_ch  0.214712714
## disc_ch                 disc_ch  0.186703970
## price_ch               price_ch  0.100466110
## pct_disc_mm         pct_disc_mm  0.073980252
## disc_mm                 disc_mm  0.064440329
## price_mm               price_mm  0.022736525
## special_ch           special_ch  0.010343821
## special_mm           special_mm  0.005869453
```

```
# test error rate
pred.gbm2 = predict(gbm2.fit, newdata = OJ[-trainRows2,])
test.err.rate = mean(pred.gbm2 != OJ$purchase[-trainRows2])*100;
test.err.rate
```

```
## [1] 19.72973
```

- The most important predictor is `loyal_ch`, followed `price_diff`. The test error rate is 19.73%.