P8106 HW4

Brian Jo Hsuan Lee

3/31/2022

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(2022)

# 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
```

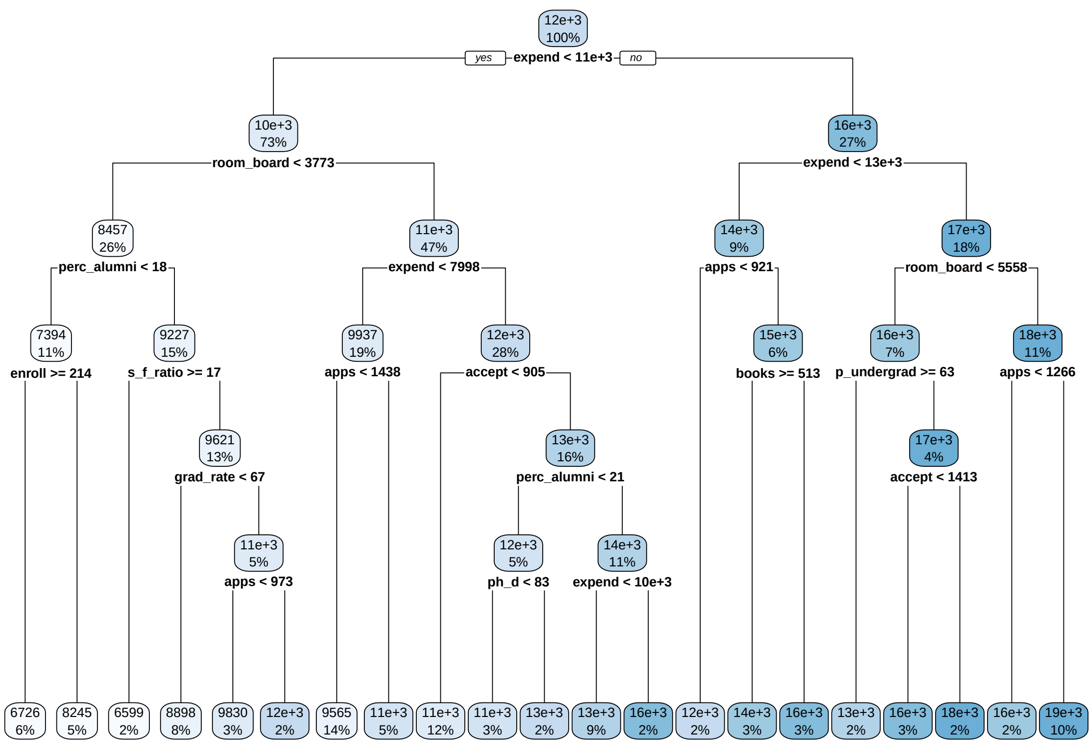
Set cross validation methods

```
# 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(2022)

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

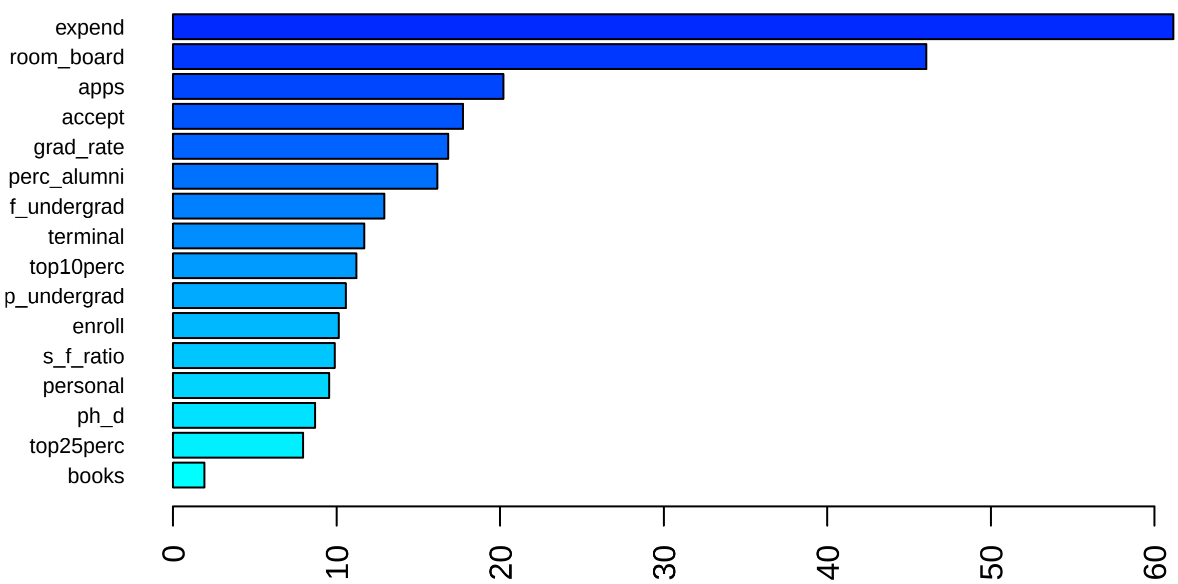
```
plot(ctree_fit$finalModel)

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)
```

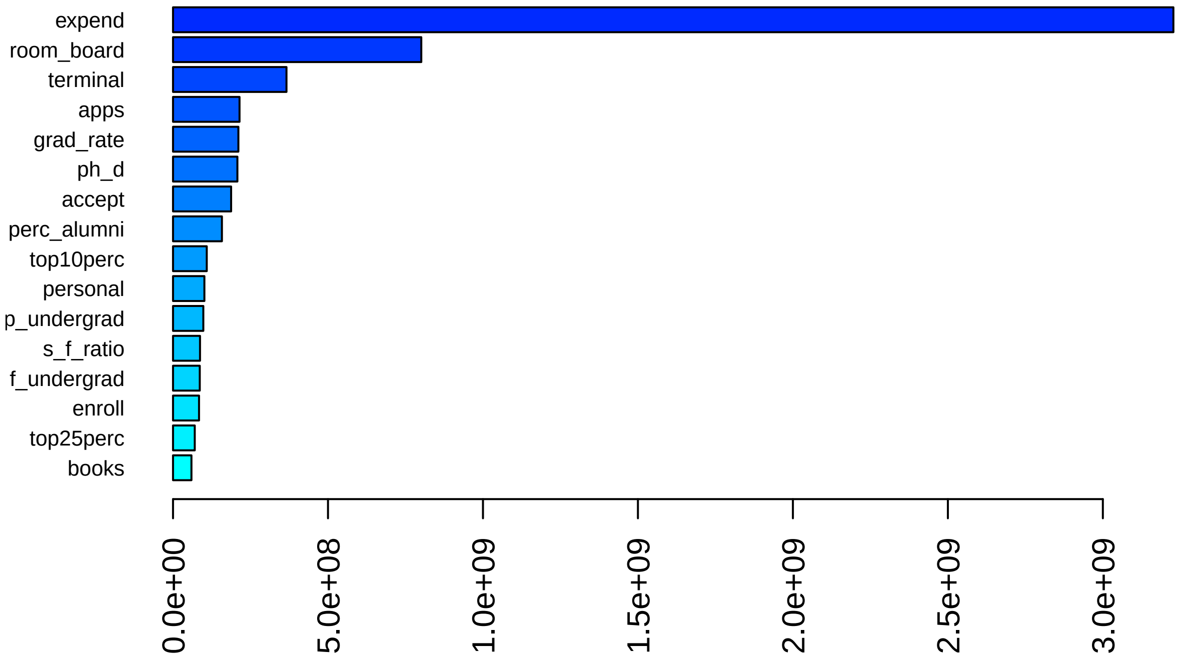
Calculate and graph variable importance using permutation and impurity metrics. Similarly, both evaluations suggest `expend` as the most important predictor for regressing out-of-state tuition, followed by `room-board`.

```
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")
```

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



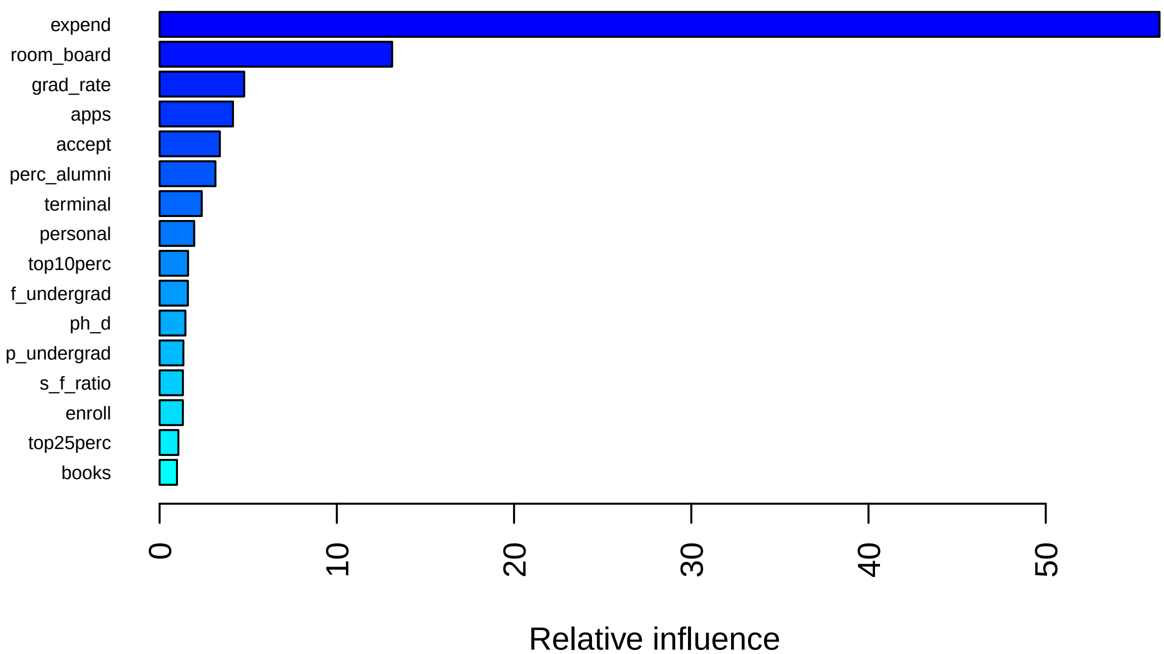
For the random forest model test error and its interpretation, see the end of part C).

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

```
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.

```
summary(gbm_fit$finalModel, las = 2, cBars = 19, 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
```

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	1976.002
boost	1893.407

```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    1174.037 1289.731 1326.377 1325.903 1364.887 1456.909    0
## boost 1218.270 1253.934 1273.515 1289.571 1304.006 1437.037    0
##
## RMSE
##           Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## rf    1486.876 1703.056 1770.978 1757.360 1846.989 1922.332    0
## boost 1555.524 1670.635 1695.745 1720.307 1779.423 1915.471    0
##
## Rsquared
##            Min.   1st Qu.    Median      Mean  3rd Qu.      Max. NA's
## rf    0.7269945 0.7545546 0.7816642 0.7781177 0.800610 0.8260480    0
## boost 0.7470747 0.7679953 0.7901223 0.7858472 0.806736 0.8140417    0
```

Problem 2: Citrus Hill or Minute Maid?

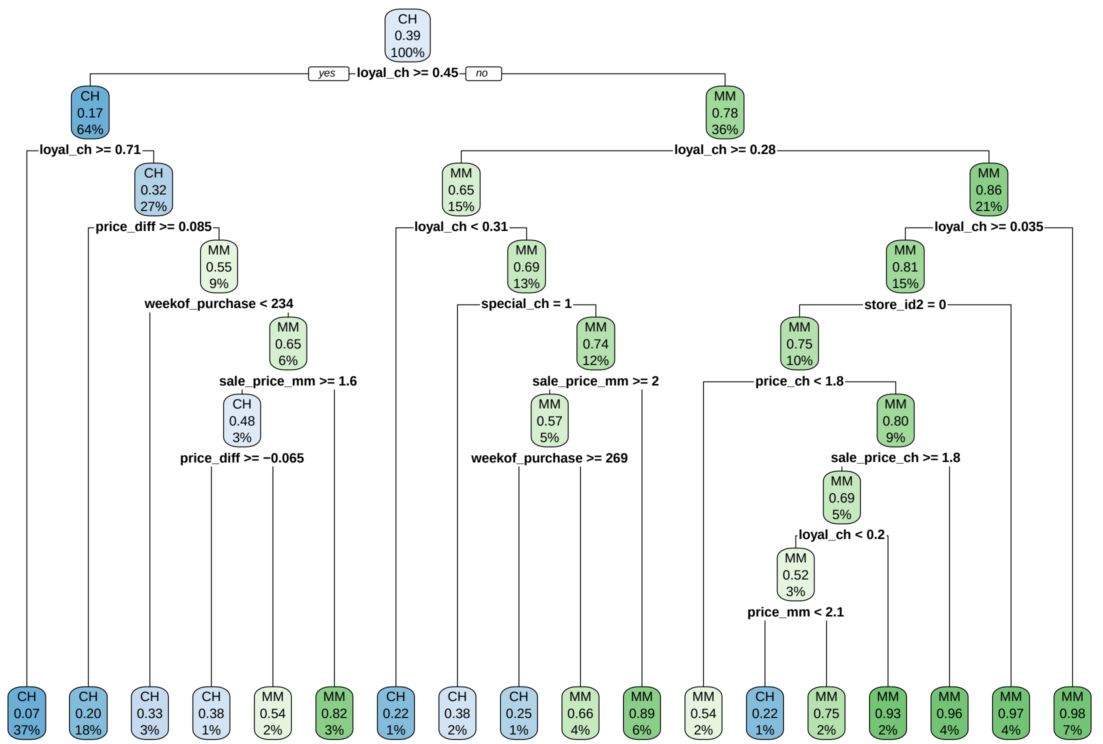
Load and split data into training and testing sets.

```
set.seed(2022)

data2 =
  OJ %>%
  janitor::clean_names() %>%
  mutate(
    purchase = factor(purchase),
    store_id = factor(store_id),
    store = factor(store)
  ) %>%
  drop_na()

# partition data into training and testing sets into randomized 4:1 splits
train_index2 = createDataPartition(y = data2$purchase, p = (699/1070), list = F)
train_data2 = data2[train_index2, ]
test_data2 = data2[-train_index2, ]

# testing set response for RMSE calculation
test_resp2 = test_data2$purchase
```

a) **Find a classification method with the lower validation error**

CART and CIT approaches under the minimal MSE Rule (code for CIT not evaluated)

```
set.seed(2022)

rpart_grid2 = data.frame(cp = exp(seq(-16,-4, length = 100)))

rpart_fit2 = train(purchase ~ . ,
                   data2,
                   subset = train_index2,
                   method = "rpart",
                   tuneGrid = rpart_grid2,
                   trControl = ctrl,
                   metric = "ROC")
# ggplot(rpart_fit2, highlight = TRUE)

rpart.plot(rpart_fit2$finalModel)
```

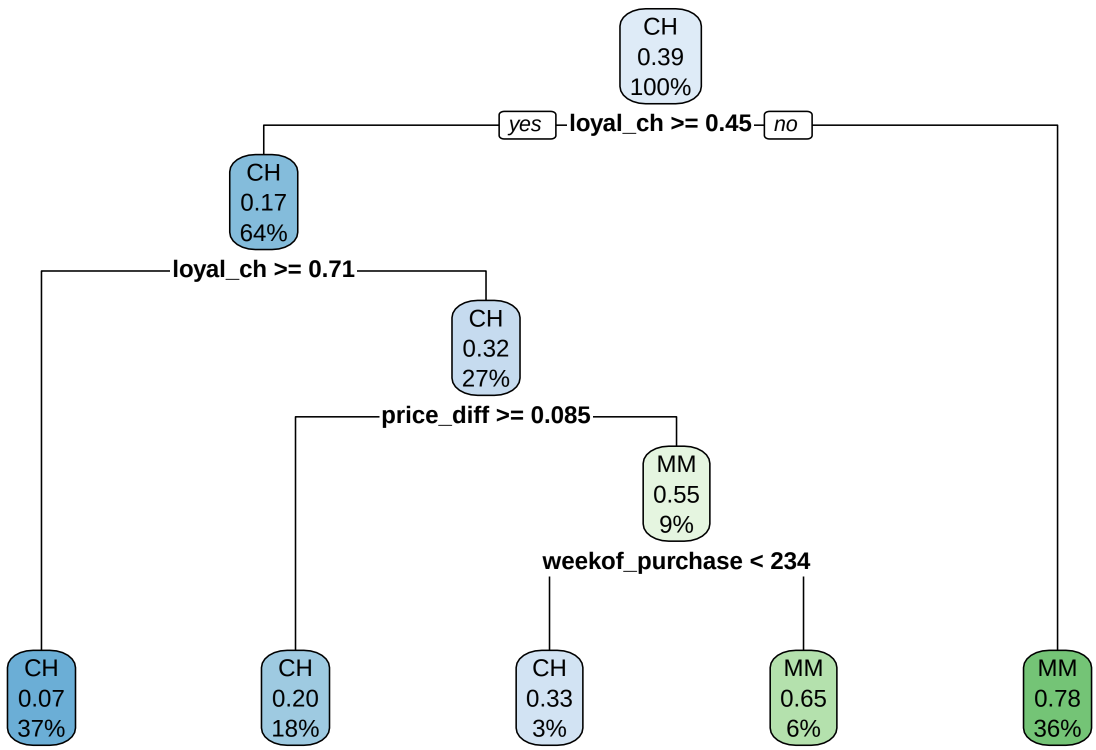



```
set.seed(2022)

ctree_grid2 = data.frame(mincriterion = 1-exp(seq(-3, 0, length = 100)))

ctree_fit2 = train(purchase ~ . ,
                   data2,
                   subset = train_index2,
                   method = "ctree",
                   tuneGrid = ctree_grid2,
                   trControl = ctrl)
ggplot(ctree_fit2, highlight = TRUE)
```

```
plot(ctree_fit2$finalModel)
```

CART and CIT approaches under the 1SE rule (code for CIT not evaluated)

```
set.seed(2022)
rpart_fit2_1se = train(purchase ~ . ,
                 data2,
                 subset = train_index2,
                 method = "rpart",
                 tuneGrid = rpart_grid2,
                 trControl = ctrl_1se)
# ggplot(rpart_fit2_1se, highlight = TRUE)

rpart.plot(rpart_fit2_1se$finalModel)
```

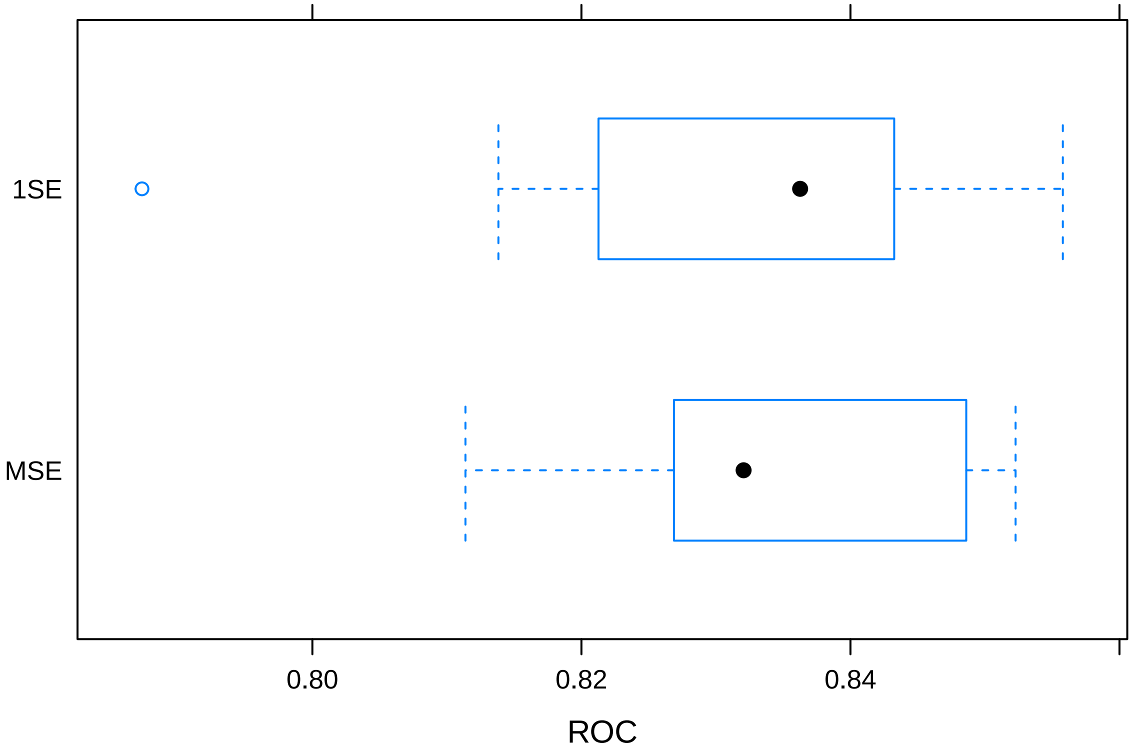



```
set.seed(2022)

ctree_fit2_1se = train(purchase ~ . ,
                       data2,
                       subset = train_index2,
                       method = "ctree",
                       tuneGrid = ctree_grid2,
                       trControl = ctrl_1se)
ggplot(ctree_fit2_1se, highlight = TRUE)

plot(ctree_fit2_1se$finalModel)
```

The 2 trees are differently sized, though they share similar splits. The optimized classification tree under the 1SE rule splits at all the thresholds at which the minimal MSE tree splits, but also 13 additional splits, creating a total of 18 terminal nodes instead of the 5 from the minimal MSE tree. As a result, there is a slight difference in cross-validation errors. In terms of ROC, even as they share a similarly sized IQR, the

1SE model has a lower spanning IQR and a higher mean.

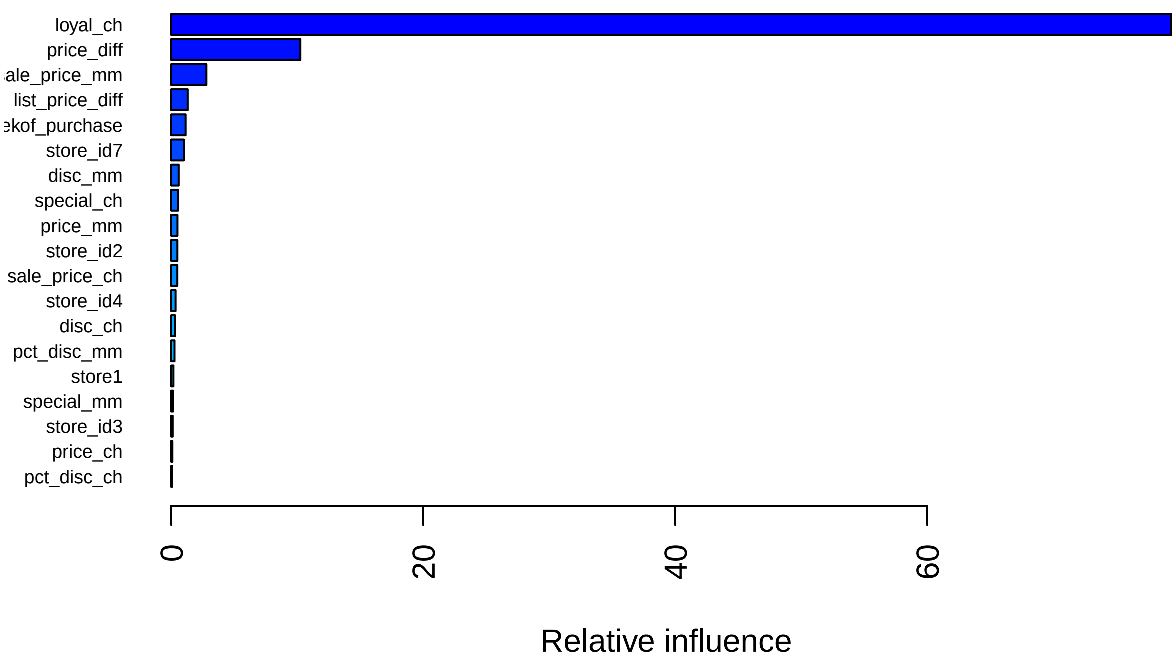
```
train_res = resamples(list(MSE = rpart_fit2, `1SE` = rpart_fit2_1se))
bwplot(train_res, metric = "ROC")
```



b) **Test error of a Adaboost classification tree**

Fit and graph predictor hierarchy by importance. `loyal_ch` and `price_diff` have a high relative influence on the classification compared to all other predictors, and the model performs at 0.926 AUC.

```
set.seed(2022)

gbm_grid2 = expand.grid(n.trees = c(2000, 3000, 4000, 5000),
                        interaction.depth = 1:6,
                        shrinkage = c(0.0005,0.001,0.002),
                        n.minobsinnode = c(1, 10))
gbm_fit2 = train(purchase ~ . ,
                 train_data2,
                 method = "gbm",
                 tuneGrid = gbm_grid2,
                 trControl = ctrl,
                 distribution = "adaboost",
                 metric = "ROC",
                 verbose = FALSE)
# ggplot(gbm_fit2, highlight = TRUE)

summary(gbm_fit2$finalModel, las = 2, cBars = 19, cex.names = 0.6)
```




```
##                               var     rel.inf
## loyal_ch                 loyal_ch 79.36353397
## price_diff             price_diff 10.25091089
## sale_price_mm       sale_price_mm  2.79276716
## list_price_diff list_price_diff  1.30828874
## weekof_purchase weekof_purchase  1.14322384
## store_id7               store_id7  1.00596275
## disc_mm                   disc_mm  0.59346157
## special_ch             special_ch  0.54820977
## price_mm                 price_mm  0.49791363
## store_id2               store_id2  0.49041715
```

```
## sale_price_ch     sale_price_ch  0.48655500
## store_id4             store_id4  0.34701457
## disc_ch                 disc_ch  0.30711537
## pct_disc_mm         pct_disc_mm  0.26289109
## store1                   store1  0.17494661
## special_mm           special_mm  0.15517502
## store_id3             store_id3  0.11650020
## price_ch               price_ch  0.09248357
## pct_disc_ch         pct_disc_ch  0.06262910
## store7Yes             store7Yes  0.00000000
## store2                   store2  0.00000000
## store3                   store3  0.00000000
## store4                   store4  0.00000000
```

```
gbm_pred = predict(gbm_fit2, newdata = test_data2, type = "prob")[,1]
gbm_roc = roc(test_resp2, gbm_pred)
```

```
## Setting levels: control = CH, case = MM
```

```
## Setting direction: controls > cases
```

```
plot(gbm_roc, col = 1)
legend("bottomright",
       legend = paste0("Adaboost: ", round(gbm_roc$auc[1], 3)),
       col = 1, lwd = 2)
```