STAA 577: HW7

Your Name

Problem 1

```r
data(OJ)
set.seed(577)
n <- nrow(OJ)
tr_index <- sample(seq_len(n), size = 800, replace = F)
OJtrain <- OJ[tr_index, ]
OJtest <- OJ[-tr_index, ]
```

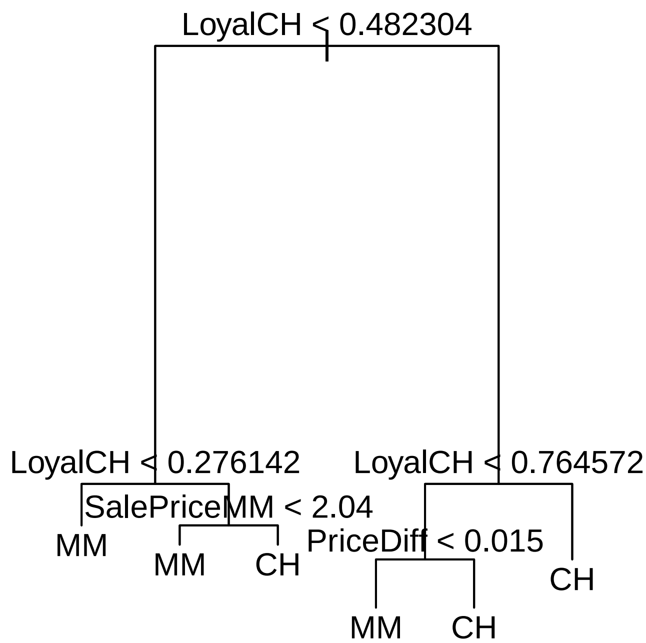
no output needed

Problem 1b

```
##
## Classification tree:
## tree(formula = Purchase ~ ., data = OJtrain)
## Variables actually used in tree construction:
## [1] "LoyalCH"     "SalePriceMM" "PriceDiff"
## Number of terminal nodes:  6
## Residual mean deviance:  0.7588 = 602.5 / 794
## Misclassification error rate: 0.1638 = 131 / 800
```

insert answer

Problem 1c



insert answer

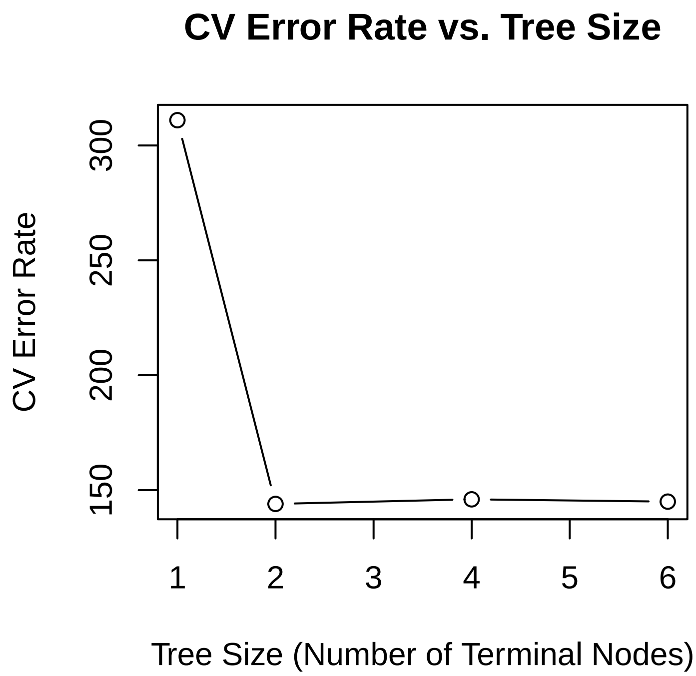
Problem 1d

```
##     oj_pred
##       CH  MM
##   CH 134  30
##   MM  24  82
```

```
## [1] 0.2
```

Test error is .2

Problem 1e




```
## [1] 2
```

tree size 2 has the lowest error rate

Problem 1f

no output is needed

Problem 1g

```
## Training Error Rate (Unpruned Tree): 0.1638
```

```
## Training Error Rate (Pruned Tree): 0.1762
```

- Training error for full tree: .1638
- Training error for pruned tree: .1762

Problem 1h

```
## [1] 0.2333
```

```
## [1] 0.2
```

- Test error for full tree: .2
- Test error for pruned tree: .2333

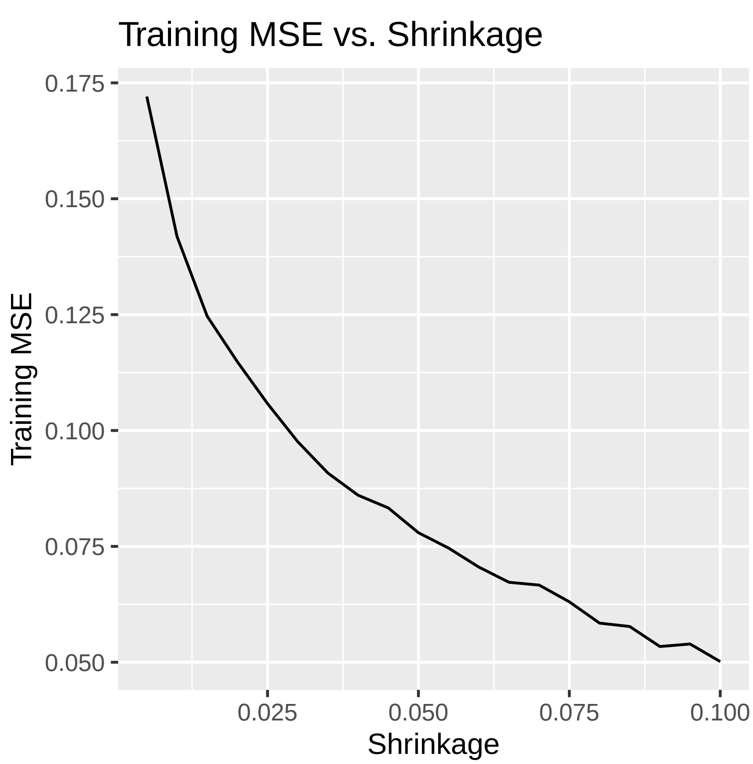
Problem 2a

Nothing to report

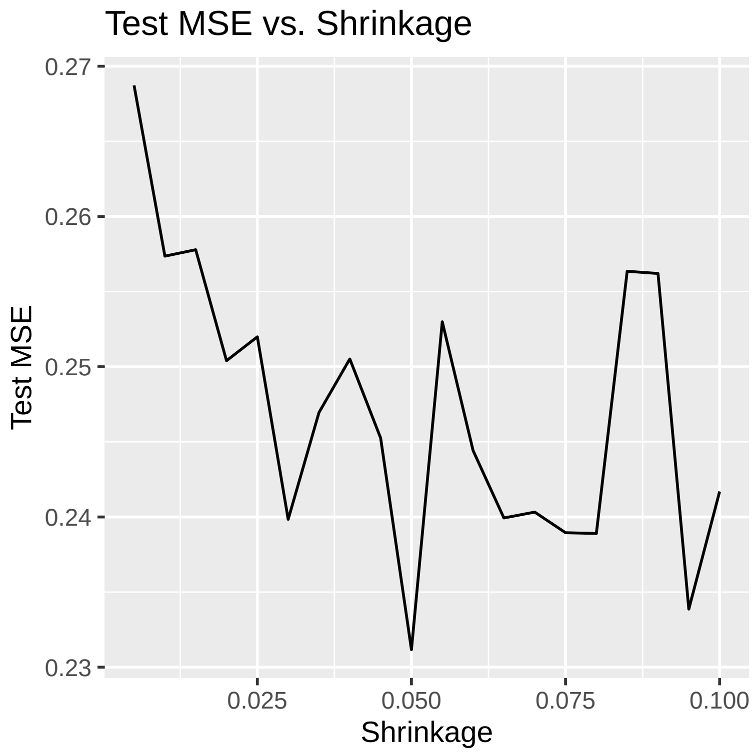
Problem 2b

no output to report

Problem 2c

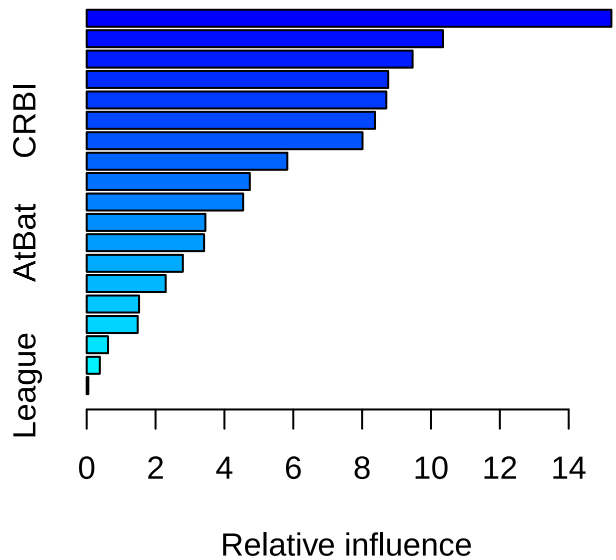


Problem 2d




```
##
## Optimal lambda: 0.05
```

Problem 2e




```
##                var     rel.inf
## CAtBat       CAtBat 15.24022191
## CHits         CHits 10.34858112
## CWalks       CWalks  9.46601826
## CRuns         CRuns  8.75659634
## PutOuts     PutOuts  8.70145002
## CRBI           CRBI  8.37371811
## Years         Years  8.01096804
## Walks         Walks  5.82616246
## Hits           Hits  4.73823746
## CHmRun       CHmRun  4.54399401
## RBI             RBI  3.44740626
## AtBat         AtBat  3.40962732
## HmRun         HmRun  2.79316846
## Assists     Assists  2.29404133
## Runs           Runs  1.52040819
## Errors       Errors  1.48151576
## NewLeague NewLeague  0.61977849
## Division   Division  0.38174580
## League       League  0.04636065
```

C At Bat, hits and Walks seem to be the top 3. Generally anything related to getting on base is important to salary which makes sense.

Problem 2f

```
## Test MSE for Bagging: 0.1879
```

- test MSE is .1879

Problem 2g

```
## Test MSE for Random Forest: 0.1953
```

- test MSE is .1953

Problem 3

```
##
## Test MSE for Linear Regression: 25.2694

## Test MSE for Boosting: 15.1895

## Test MSE for Bagging: 9.6815

## Test MSE for Random Forest: 11.4519
```

Bagging is the most effective in this case with a MSE of 9.6815

Problem 4

- Include both sketches. You can draw it with R (not easy) or just draw the sketch by hand.

Problem 5

```
## Majority vote classification = Red

## Average probability classification = Green
```

if we use majroity voting its red but average probability is green.

Appendix

```
library(knitr)
# install the tidyverse library (do this once) install.packages('tidyverse')
library(tidyverse)
# set chunk and figure default options
knitr::opts_chunk$set(echo = FALSE, message = FALSE, warning = FALSE, fig.width = 4,
    fig.height = 4, tidy = TRUE)
library(tidyverse)  # data manip
library(ISLR)  # data
library(GGally)  # pairs plots
library(gam)  # gam models
library(leaps)  # regression subsets
library(tree)  # trees
library(randomForest)  # random forest
library(gbm)  # boosting
library(glmnet)  # lasso
library(ISLR)
data(OJ)
set.seed(577)
n <- nrow(OJ)
tr_index <- sample(seq_len(n), size = 800, replace = F)
OJtrain <- OJ[tr_index, ]
OJtest <- OJ[-tr_index, ]
# prob 1b
library(tree)
oj_tree <- tree(Purchase ~ ., data = OJtrain)
summary(oj_tree)

# problem 1c
plot(oj_tree)
text(oj_tree, pretty = 0)

# problem 1d
oj_pred <- predict(oj_tree, newdata = OJtest, type = "class")
conf_mat <- table(OJtest$Purchase, oj_pred)

test_error_rate_unpruned <- (conf_mat[1, 2] + conf_mat[2, 1])/sum(conf_mat)

conf_mat
test_error_rate_unpruned


set.seed(577)
cv_oj <- cv.tree(oj_tree, FUN = prune.misclass)
plot(cv_oj$size, cv_oj$dev, type = "b", xlab = "Tree Size (Number of Terminal Nodes)",
    ylab = "CV Error Rate", main = "CV Error Rate vs. Tree Size")
optimal_size <- cv_oj$size[which.min(cv_oj$dev)]
```

```r
optimal_size


# problem f
pruned_oj <- prune.misclass(oj_tree, best = optimal_size)

# prob1g
train_pred_unpruned <- predict(oj_tree, newdata = OJtrain, type = "class")
conf_mat_train_unpruned <- table(OJtrain$Purchase, train_pred_unpruned)
train_error_unpruned <- (conf_mat_train_unpruned[1, 2] + conf_mat_train_unpruned[2,
    1])/sum(conf_mat_train_unpruned)
cat("Training Error Rate (Unpruned Tree):", round(train_error_unpruned, 4), "\n")

train_pred_pruned <- predict(pruned_oj, newdata = OJtrain, type = "class")
conf_mat_train_pruned <- table(OJtrain$Purchase, train_pred_pruned)
train_error_pruned <- (conf_mat_train_pruned[1, 2] + conf_mat_train_pruned[2, 1])/sum(conf_mat_train_pru
cat("Training Error Rate (Pruned Tree):", round(train_error_pruned, 4), "\n")
# prob1h
test_pred_pruned <- predict(pruned_oj, newdata = OJtest, type = "class")
conf_mat_test_pruned <- table(OJtest$Purchase, test_pred_pruned)
test_error_pruned <- (conf_mat_test_pruned[1, 2] + conf_mat_test_pruned[2, 1])/sum(conf_mat_test_pruned
round(test_error_pruned, 4)

test_error_rate_unpruned

# prob 2a
Hitters <- Hitters[complete.cases(Hitters), ]
Hitters$lsalary <- log(Hitters$Salary)
Hitters <- Hitters[, !(names(Hitters) %in% "Salary")]

set.seed(577)
n <- nrow(Hitters)
tr_index <- sample(seq_len(n), size = 200, replace = F)
Htrain <- Hitters[tr_index, ]
Htest <- Hitters[-tr_index, ]
# prob 2c
lambda_seq <- seq(0.005, 0.1, length.out = 20)
train_mse <- numeric(length(lambda_seq))
test_mse <- numeric(length(lambda_seq))

for (i in 1:length(lambda_seq)) {
    boost_mod <- gbm(lsalary ~ ., data = Htrain, distribution = "gaussian", n.trees = 1000,
        shrinkage = lambda_seq[i], verbose = FALSE)
    train_pred <- predict(boost_mod, newdata = Htrain, n.trees = 1000)
    test_pred <- predict(boost_mod, newdata = Htest, n.trees = 1000)
    train_mse[i] <- mean((Htrain$lsalary - train_pred)^2)
    test_mse[i] <- mean((Htest$lsalary - test_pred)^2)
}

# Plot training MSE vs. lambda
df_train <- data.frame(lambda = lambda_seq, MSE = train_mse)

ggplot(df_train, aes(x = lambda, y = MSE)) + geom_line() + labs(title = "Training MSE vs. Shrinkage",
```

```r
    x = "Shrinkage", y = "Training MSE")

# prob 2d
df_test <- data.frame(lambda = lambda_seq, MSE = test_mse)
ggplot(df_test, aes(x = lambda, y = MSE)) + geom_line() + labs(title = "Test MSE vs. Shrinkage",
    x = "Shrinkage", y = "Test MSE")

# Identify and print the lambda value that minimizes the test MSE
optimal_lambda <- lambda_seq[which.min(test_mse)]
cat("\nOptimal lambda:", optimal_lambda, "\n")
# problem 2e
boost_mod_final <- gbm(lsalary ~ ., data = Htrain, distribution = "gaussian", n.trees = 1000,
    shrinkage = optimal_lambda, verbose = FALSE)
print(summary(boost_mod_final))
# prob 2f
set.seed(577)
bag_mod <- randomForest(lsalary ~ ., data = Htrain, mtry = ncol(Htrain) - 1, ntree = 500)
bag_pred <- predict(bag_mod, newdata = Htest)
bag_test_mse <- mean((Htest$lsalary - bag_pred)^2)
cat("Test MSE for Bagging:", round(bag_test_mse, 4), "\n")
# problem 2g
set.seed(577)
rf_mod <- randomForest(lsalary ~ ., data = Htrain, ntree = 500)
rf_pred <- predict(rf_mod, newdata = Htest)
rf_test_mse <- mean((Htest$lsalary - rf_pred)^2)
cat("Test MSE for Random Forest:", round(rf_test_mse, 4), "\n")
# problem 3

set.seed(577)
library(MASS)

n_boston <- nrow(Boston)
train_index <- sample(seq_len(n_boston), size = round(0.7 * n_boston), replace = FALSE)
BostonTrain <- Boston[train_index, ]
BostonTest <- Boston[-train_index, ]

# Linear
lm_mod <- lm(medv ~ ., data = BostonTrain)
lm_pred <- predict(lm_mod, newdata = BostonTest)
lm_test_mse <- mean((BostonTest$medv - lm_pred)^2)
cat("\nTest MSE for Linear Regression:", round(lm_test_mse, 4), "\n")

# Boosting
boost_boston <- gbm(medv ~ ., data = BostonTrain, distribution = "gaussian", n.trees = 1000,
    shrinkage = 0.01, verbose = FALSE)
boost_pred <- predict(boost_boston, newdata = BostonTest, n.trees = 1000)
boost_test_mse <- mean((BostonTest$medv - boost_pred)^2)
cat("Test MSE for Boosting:", round(boost_test_mse, 4), "\n")

# Baggin
bag_boston <- randomForest(medv ~ ., data = BostonTrain, mtry = ncol(BostonTrain) -
    1, ntree = 500)
bag_pred_boston <- predict(bag_boston, newdata = BostonTest)
```

```r
bag_test_mse_boston <- mean((BostonTest$medv - bag_pred_boston)^2)
cat("Test MSE for Bagging:", round(bag_test_mse_boston, 4), "\n")

# Random Forrest
rf_boston <- randomForest(medv ~ ., data = BostonTrain, ntree = 500)
rf_pred_boston <- predict(rf_boston, newdata = BostonTest)
rf_test_mse_boston <- mean((BostonTest$medv - rf_pred_boston)^2)
cat("Test MSE for Random Forest:", round(rf_test_mse_boston, 4), "\n")


# problem 5
p <- c(0.1, 0.15, 0.2, 0.2, 0.55, 0.6, 0.6, 0.65, 0.7, 0.75)

num_red <- sum(p > 0.5)
majority_vote <- ifelse(num_red > length(p)/2, "Red", "Green")

cat("Majority vote classification =", majority_vote, "\n")

p_avg <- mean(p)
avg_prob_class <- ifelse(p_avg > 0.5, "Red", "Green")

cat("Average probability classification =", avg_prob_class, "\n")
```