ETC3250 2017 - Lab 10

Tree-based methods

Souhaib Ben Taieb

4 October 2017

Purpose

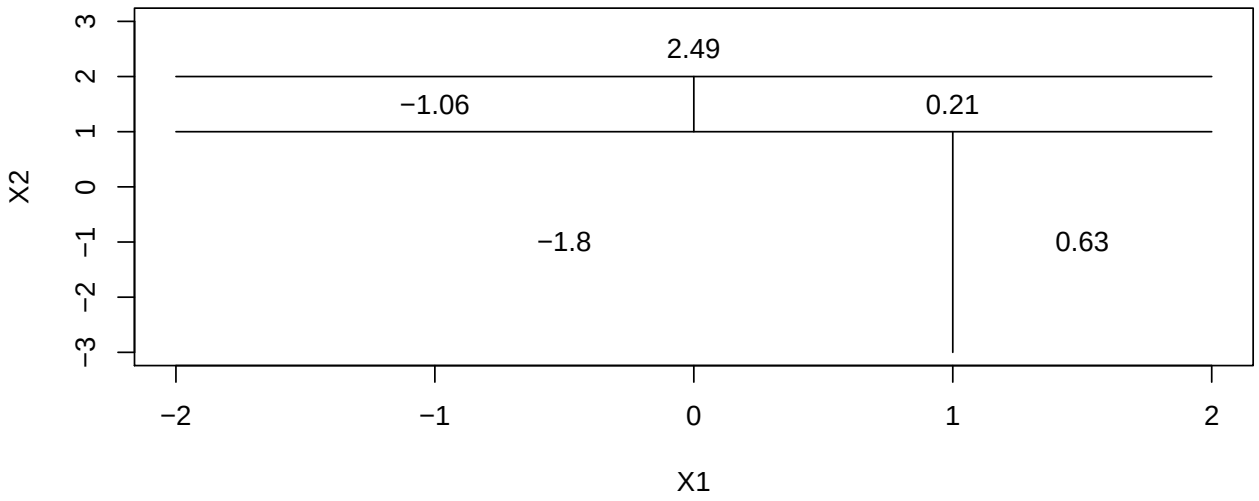
The goal of this lab is to understand tree-based methods.

Exercice 1

ISLR Section 8.4, exercise 4(b). Use functions *plot*, *lines* and *text*.

ISLR Section 8.4, exercise 4(b).

```
plot(NA, NA, type = "n", xlim = c(-2, 2), ylim = c(-3, 3), xlab = "X1", ylab = "X2")

# X2 < 1
lines(x = c(-2, 2), y = c(1, 1))
# X1 < 1 with X2 < 1
lines(x = c(1, 1), y = c(-3, 1))
text(x = (-2 + 1)/2, y = -1, labels = c(-1.8))
text(x = 1.5, y = -1, labels = c(0.63))

# X2 < 2 with X2 >= 1
lines(x = c(-2, 2), y = c(2, 2))
text(x = 0, y = 2.5, labels = c(2.49))

# X1 < 0 with X2<2 and X2>=1
lines(x = c(0, 0), y = c(1, 2))
text(x = -1, y = 1.5, labels = c(-1.06))
text(x = 1, y = 1.5, labels = c(0.21))
```



Exercice 2

Read and run the code in Section 8.3 of ISLR.

Assignment

Question 1

ISLR Section 8.4, exercise 9.

(a)

```
library(ISLR)
set.seed(1986)

n <- nrow(OJ)
id.train <- sample(n, 800)
OJ.train <- OJ[id.train, ]
OJ.test  <- OJ[-id.train, ]
```

(b)

```
library(tree)
oj.tree <- tree(Purchase ~ ., data = OJ.train)
summary(oj.tree)
#
# Classification tree:
# tree(formula = Purchase ~ ., data = OJ.train)
# Variables actually used in tree construction:
# [1] "LoyalCH"   "PriceDiff" "SpecialCH"
# Number of terminal nodes:  9
# Residual mean deviance:  0.7265 = 574.6 / 791
# Misclassification error rate: 0.15 = 120 / 800
```

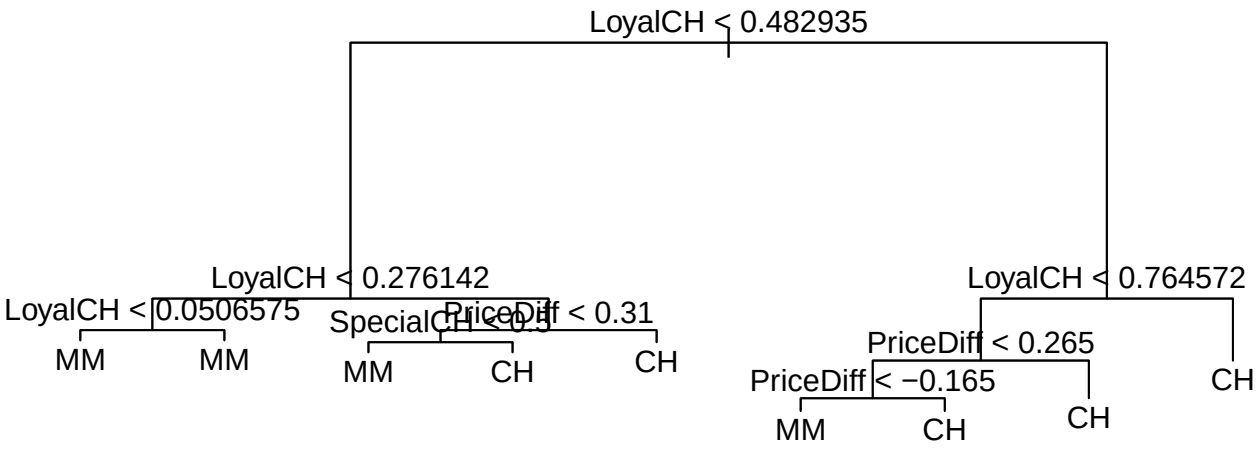
(c)

```
oj.tree
# node), split, n, deviance, yval, (yprob)
#       * denotes terminal node
#
#  1) root 800 1070.00 CH ( 0.61000 0.39000 )
#    2) LoyalCH < 0.482935 304  335.10 MM ( 0.24013 0.75987 )
#      4) LoyalCH < 0.276142 171  119.30 MM ( 0.11111 0.88889 )
#        8) LoyalCH < 0.0506575 63   10.27 MM ( 0.01587 0.98413 ) *
#        9) LoyalCH > 0.0506575 108   97.32 MM ( 0.16667 0.83333 ) *
#      5) LoyalCH > 0.276142 133  179.60 MM ( 0.40602 0.59398 )
#       10) PriceDiff < 0.31 97  120.00 MM ( 0.30928 0.69072 )
```

```
#         20) SpecialCH < 0.5 83   91.66 MM ( 0.24096 0.75904 ) *
#         21) SpecialCH > 0.5 14   16.75 CH ( 0.71429 0.28571 ) *
#       11) PriceDiff > 0.31 36   45.83 CH ( 0.66667 0.33333 ) *
#    3) LoyalCH > 0.482935 496  441.60 CH ( 0.83669 0.16331 )
#      6) LoyalCH < 0.764572 239  292.50 CH ( 0.69874 0.30126 )
#       12) PriceDiff < 0.265 145  199.50 CH ( 0.55172 0.44828 )
#         24) PriceDiff < -0.165 38   45.73 MM ( 0.28947 0.71053 ) *
#         25) PriceDiff > -0.165 107  139.20 CH ( 0.64486 0.35514 ) *
#       13) PriceDiff > 0.265 94   49.83 CH ( 0.92553 0.07447 ) *
#      7) LoyalCH > 0.764572 257   78.01 CH ( 0.96498 0.03502 ) *
```

(d)

```
plot(oj.tree)
text(oj.tree, pretty = 0)
```



(e)

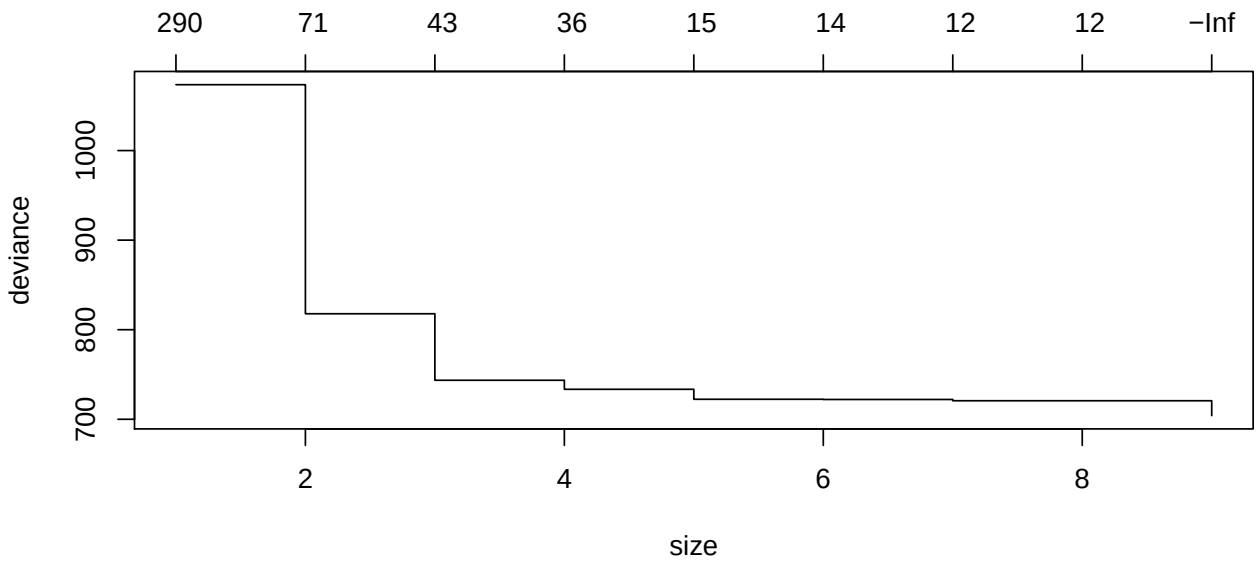
```
oj.pred <- predict(oj.tree, OJ.test, type = "class")
table(OJ.test$Purchase, oj.pred)
#     oj.pred
#       CH  MM
#   CH 147  18
#   MM  30  75
mean(OJ.test$Purchase != oj.pred)
# [1] 0.1777778
```

(f)

```
cv.oj <- cv.tree(oj.tree, FUN = prune.tree)
```

(g)

```
plot(cv.oj)
```



(h)

```
print(cv.oj$size[which.min(cv.oj$dev)])
# [1] 9
```

(i)

```
oj.pruned <- prune.tree(oj.tree, best = cv.oj$size[which.min(cv.oj$dev)])
```

(j)

```
summary(oj.pruned)
#
# Classification tree:
# tree(formula = Purchase ~ ., data = OJ.train)
# Variables actually used in tree construction:
# [1] "LoyalCH"   "PriceDiff" "SpecialCH"
# Number of terminal nodes:  9
# Residual mean deviance:  0.7265 = 574.6 / 791
# Misclassification error rate: 0.15 = 120 / 800
summary(oj.tree)
#
# Classification tree:
# tree(formula = Purchase ~ ., data = OJ.train)
```

```
# Variables actually used in tree construction:
# [1] "LoyalCH"   "PriceDiff" "SpecialCH"
# Number of terminal nodes:  9
# Residual mean deviance:  0.7265 = 574.6 / 791
# Misclassification error rate: 0.15 = 120 / 800
```

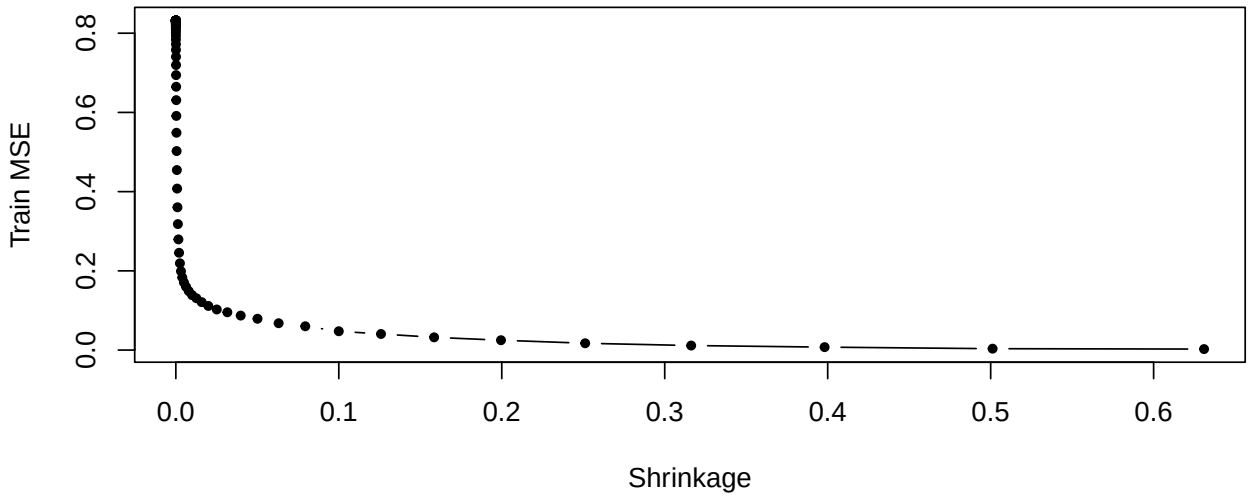
(k)

```
oj.pred <- predict(oj.tree, OJ.test, type = "class")
mean(OJ.test$Purchase != oj.pred)
# [1] 0.17777778

oj.pred_pruned <- predict(oj.pruned, OJ.test, type = "class")
mean(OJ.test$Purchase != oj.pred_pruned)
# [1] 0.17777778
```

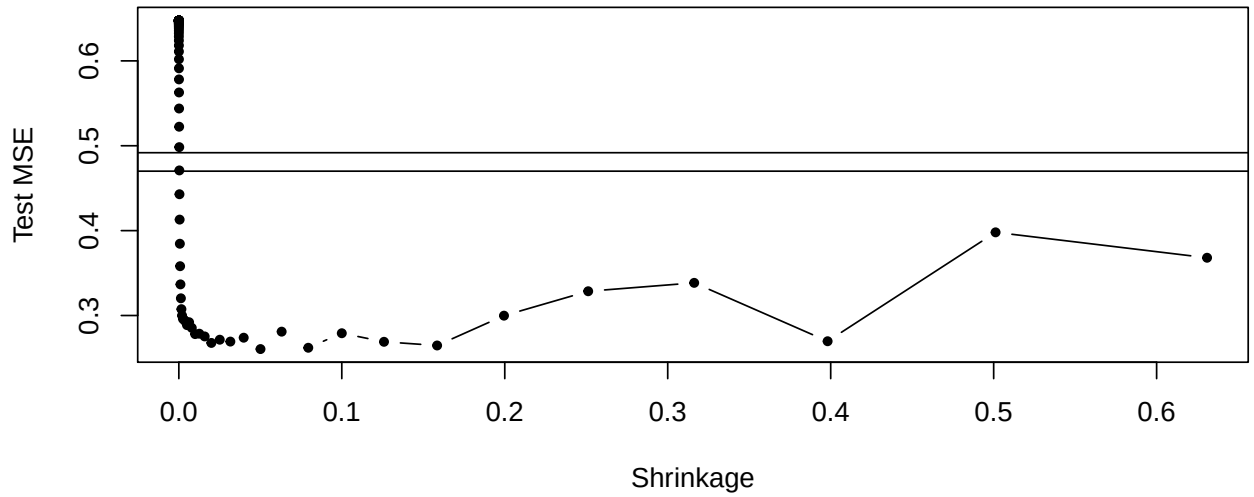
Question 2

ISLR Section 8.4, exercise 10.

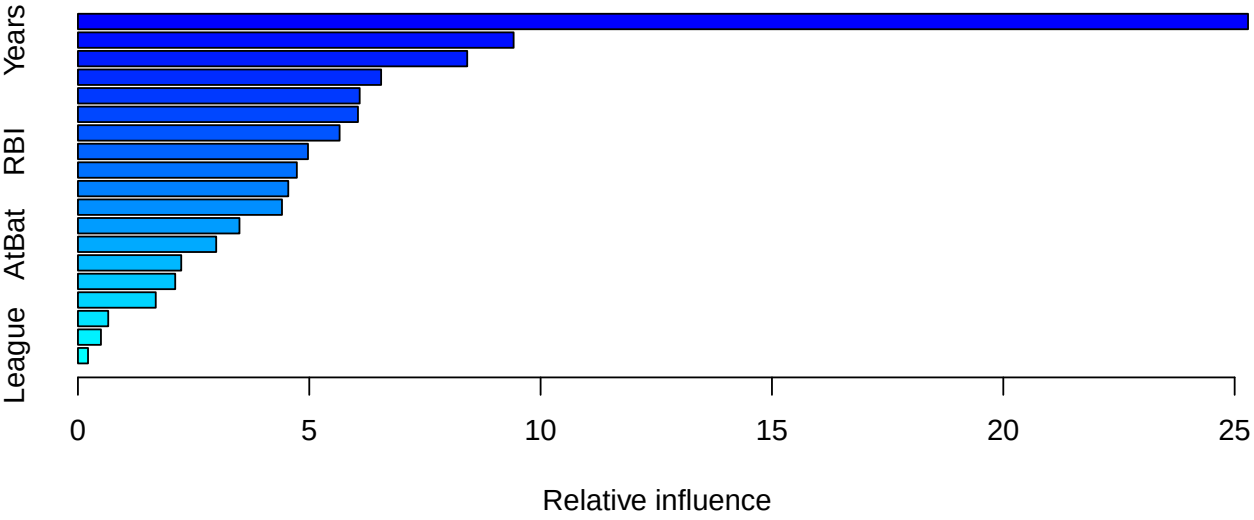
```
DT <- Hitters[-which(is.na(Hitters$Salary)), ]
DT$Salary = log(DT$Salary)

id.train <- seq(200)
DT.train <- DT[id.train, ]
DT.test  <- DT[-id.train, ]

library(gbm)
set.seed(1986)
lambdas <- 10^seq(-10, -0.2, by = 0.1)
length.lambdas <- length(lambdas)
train.errors <- rep(NA, length.lambdas)
test.errors <- rep(NA, length.lambdas)
for (i in seq_along(lambdas)) {
    boost.hitters = gbm(Salary ~ ., data = DT.train, distribution = "gaussian",
        n.trees = 1000, shrinkage = lambdas[i])
    train.pred <- predict(boost.hitters, DT.train, n.trees = 1000)
    test.pred <- predict(boost.hitters, DT.test, n.trees = 1000)
    train.errors[i] <- mean((DT.train$Salary - train.pred)^2)
    test.errors[i] <- mean((DT.test$Salary - test.pred)^2)
}

plot(lambdas, train.errors, type = "b", xlab = "Shrinkage", ylab = "Train MSE", pch = 20)
```




```
plot(lambdas, test.errors, type = "b", xlab = "Shrinkage", ylab = "Test MSE",   pch = 20)


#
min(test.errors)
# [1] 0.2604598

lm.fit <- lm(Salary ~ ., data = DT.train)
lm.pred <- predict(lm.fit, DT.test)
test.MSE.lm <- mean((DT.test$Salary - lm.pred)^2)

library(glmnet)
x <- model.matrix(Salary ~ ., data = DT.train)
y <- DT.train$Salary
x.test <- model.matrix(Salary ~ ., data = DT.test)
lasso.fit <- glmnet(x, y, alpha = 1)
lasso.pred <- predict(lasso.fit, s = 0.01, newx = x.test)
test.MSE.reg <- mean((DT.test$Salary - lasso.pred)^2)

plot(lambdas, test.errors, type = "b", xlab = "Shrinkage", ylab = "Test MSE",   pch = 20)
abline(h = c(test.MSE.lm, test.MSE.reg))
```




```
#
boost.best <- gbm(Salary ~ ., data = DT.train, distribution = "gaussian",
    n.trees = 1000, shrinkage = lambdas[which.min(test.errors)])
summary(boost.best)
```




```
#                var    rel.inf
# CAtBat       CAtBat 25.2892300
# Years         Years  9.4165481
# CWalks       CWalks  8.4137365
# Walks         Walks  6.5527099
# CHits         CHits  6.0895530
# PutOuts     PutOuts  6.0522227
# CRBI           CRBI  5.6557992
# RBI             RBI  4.9724988
# Hits           Hits  4.7315686
# CHmRun       CHmRun  4.5463486
# Assists     Assists  4.4102872
# HmRun         HmRun  3.4913940
```

```
# AtBat         AtBat  2.9888757
# Runs           Runs  2.2330914
# Errors       Errors  2.1029732
# CRuns         CRuns  1.6821558
# Division   Division  0.6551103
# NewLeague NewLeague  0.4975751
# League       League  0.2183218

#
library(randomForest)
rf.hitters <- randomForest(Salary ~ ., data = DT.train, ntree = 500, mtry = ncol(DT.train) -1)
rf.pred <- predict(rf.hitters, DT.test)
mean((DT.test$Salary - rf.pred)^2)
# [1] 0.2241872
```

TURN IN

- Your `.Rmd` file (which should knit without errors and without assuming any packages have been pre-loaded)
- Your Word (or pdf) file that results from knitting the Rmd.
- DUE: 8 October 11:55pm (late submissions not allowed), loaded into moodle