CART Tree Tutorial

## Regression Trees

* First we use ISLR Hitters dataset

library(ISLR)  
data(Hitters)

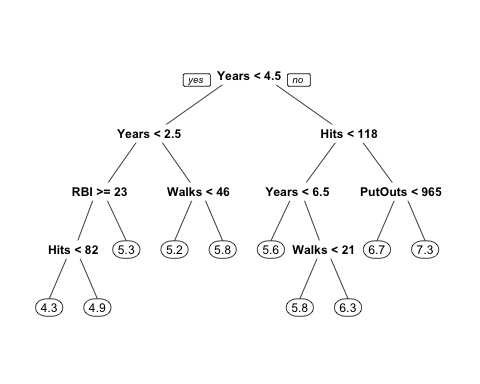
* Load rpart library and create train/test split

library(rpart)  
library(rpart.plot)  
Hitters <- na.omit(Hitters) #Remove NA for demo  
Hitters$Salary <- log(Hitters$Salary)  
set.seed(200)  
hitter.split <- sample(1:nrow(Hitters), size=nrow(Hitters) \* 0.7)  
h.train <- Hitters[hitter.split,]  
h.test <- Hitters[-hitter.split,]

### CART Tree (no pruning)

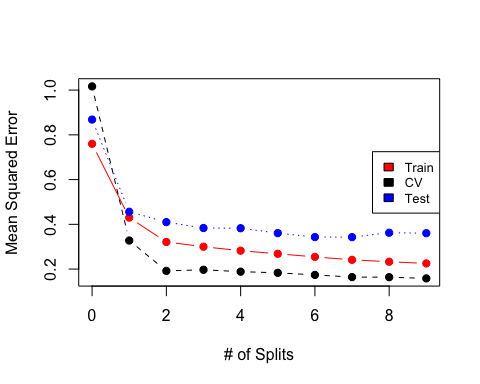
* Grow a large tree

example.cart <- rpart(formula = Salary ~ Years + Hits + RBI + PutOuts + Runs + Walks, data = h.train, method = "anova", control = rpart.control(minbucket = 6))  
prp(example.cart, roundint = FALSE)



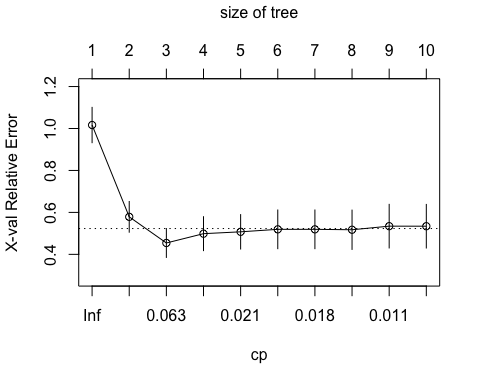
# Get train/CV/test error on different levels of alpha

cp.param <- example.cart$cptable  
train.mse <- double(10)  
cv.mse <- double(10)  
test.mse <- double(10)  
  
for (i in 1:10) {  
 alpha <- cp.param[i, 'CP']  
 train.mse[i] <- mean((h.train$Salary - predict(prune(example.cart, cp=alpha), newdata = h.train))^2)  
 cv.mse[i] <- cp.param[i, 'xerror'] \* cp.param[i, 'rel error']  
 test.mse[i] <- mean((h.test$Salary - predict(prune(example.cart, cp=alpha), newdata = h.test))^2)  
}  
  
matplot(cp.param[,'nsplit'], cbind(train.mse, cv.mse, test.mse), pch=19, col=c("red", "black", "blue"), type="b", ylab="Mean Squared Error", xlab="# of Splits")  
legend("right", c('Train', 'CV', 'Test') ,col=seq\_len(3),cex=0.8,fill=c("red", "black", "blue"))



* Looking at the plot of CV values only, a tree of size 3 looks best

plotcp(example.cart)



* We can see the regions of the 3 split tree

library(ggplot2)  
region <- ifelse(h.train$Years < 2.5, 'Region 4', ifelse(h.train$Years < 4.5, 'Region 1', ifelse(h.train$Hits < 117.5, 'Region 2', 'Region 3')))  
ggplot(data=h.train) +   
 geom\_point(aes(x=Years, y=Hits, color=Salary, shape=region)) +  
 geom\_vline(xintercept = 4.5) +  
 geom\_vline(xintercept =2.5) +  
 geom\_segment(x = 4.5, y= 117.5, xend=30, yend=117.5) +  
 theme\_minimal()



### Bagged Trees

* Bagging is equivalent to Random Forest when the mtry parameter is set to the number of dimensions (p)

library(randomForest)

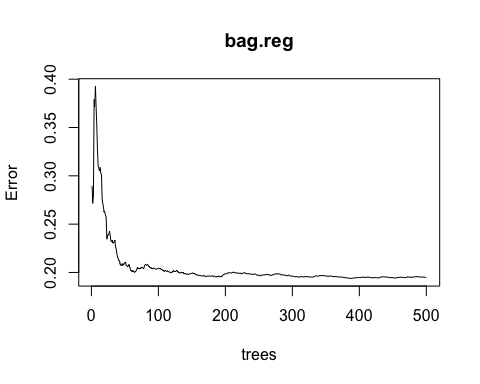
## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##   
## Attaching package: 'randomForest'

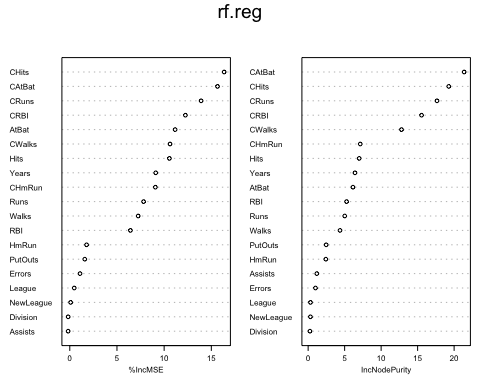
## The following object is masked from 'package:ggplot2':  
##   
## margin

set.seed(1)  
bag.reg <- randomForest(Salary~., data=h.train, mtry=ncol(h.train) - 1, importance=TRUE)  
plot(bag.reg)

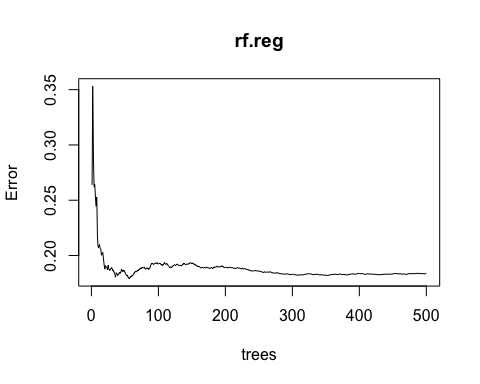


### Random Forest

set.seed(1)  
# setting mtry to sqrt(p) is a rule of thumb, this number can be set by  
# k fold CV as well  
rf.reg <- randomForest(Salary~., data=h.train, mtry=round(sqrt(ncol(h.train) - 1)), importance=TRUE)  
#importance(rf.reg)  
varImpPlot(rf.reg, cex=0.5)

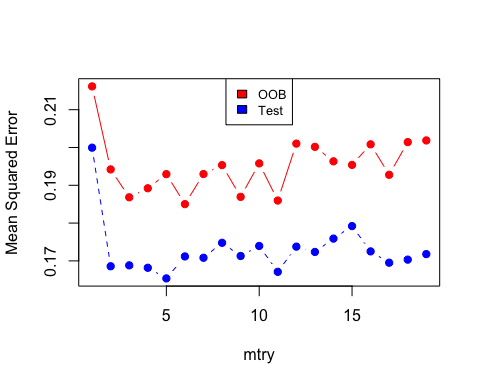


plot(rf.reg)



* ntree of around 225 looks good
* Try choosing mtry by plotting the OOB error

p <- ncol(h.train) - 1  
oob.error <- double(p) #initialize empty vector  
test.error <- double(p)  
set.seed(1)  
  
for(m in 1:p) {  
 fit <- randomForest(Salary ~ ., data=h.train, mtry=m, ntree=225)  
 oob.error[m] <- fit$mse[225]  
 test.error[m] <- mean((h.test$Salary - predict(fit, newdata=h.test))^2)  
}  
  
matplot(1:p, cbind(oob.error, test.error), pch=19, col=c("red", "blue"), type="b", ylab="Mean Squared Error")  
legend("top", c('OOB', 'Test') ,col=seq\_len(2),cex=0.8,fill=c("red", "blue"))



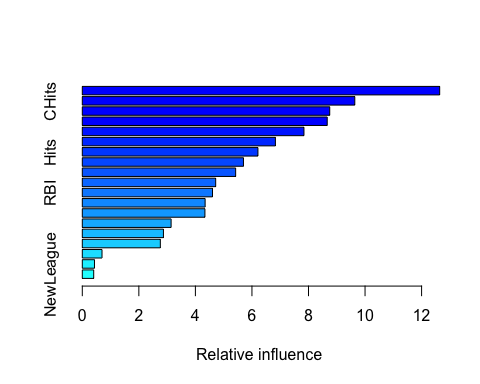
### Boosting

library(gbm)

## Warning: package 'gbm' was built under R version 3.5.2

## Loaded gbm 2.1.5

set.seed(1)  
reg.boost = gbm(Salary ~ ., data = h.train, n.trees = 5000, distribution = "gaussian", shrinkage = 0.01)  
summary(reg.boost)



## var rel.inf  
## CAtBat CAtBat 12.6396654  
## CHits CHits 9.6375303  
## CRuns CRuns 8.7510321  
## CRBI CRBI 8.6619247  
## PutOuts PutOuts 7.8395009  
## Walks Walks 6.8293835  
## Hits Hits 6.2118303  
## Years Years 5.6998201  
## CWalks CWalks 5.4237194  
## HmRun HmRun 4.7175858  
## RBI RBI 4.6050276  
## CHmRun CHmRun 4.3438790  
## AtBat AtBat 4.3369376  
## Assists Assists 3.1408294  
## Errors Errors 2.8696911  
## Runs Runs 2.7600580  
## League League 0.6972765  
## Division Division 0.4307474  
## NewLeague NewLeague 0.4035608

par(mfrow=c(2,2))  
plot(reg.boost, i="Walks")

plot(reg.boost, i="CHits")

plot(reg.boost, i="CRBI")

plot(reg.boost, i="Years")

## Classification Trees

* Now we use Heart dataset

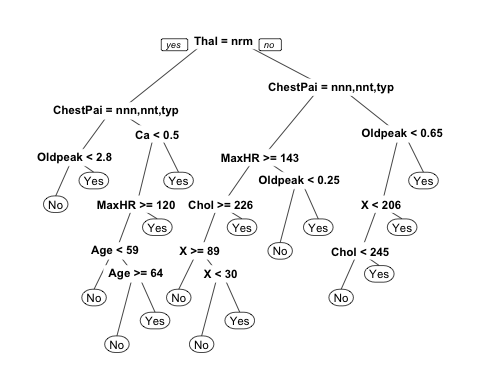
Heart <-read.csv('https://www-bcf.usc.edu/~gareth/ISL/Heart.csv')

* Create train/test split

Heart <- na.omit(Heart) #Remove NA for demo  
set.seed(490)  
heart.split <- sample(1:nrow(Heart), size=nrow(Heart) \* 0.7)  
heart.train <- Heart[heart.split,]  
heart.test <- Heart[-heart.split,]

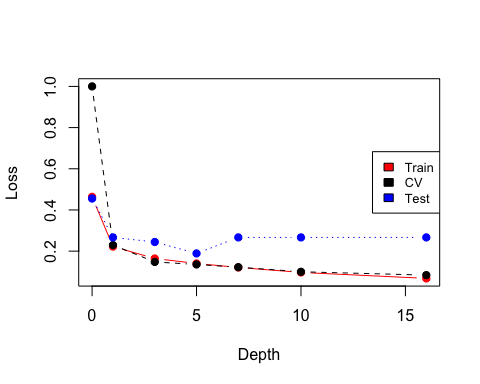
### CART Tree (no pruning)

#Create tree model  
class.cart <- rpart(formula = AHD ~ ., data = heart.train, method = "class", control = rpart.control(minbucket = 2, xval = 10))  
prp(class.cart, roundint = FALSE)



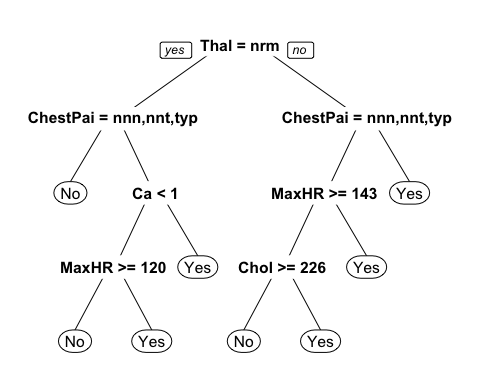
# Pruned CART Tree

cp.class.param <- class.cart$cptable  
train.acc <- double(7)  
cv.acc <- double(7)  
test.acc <- double(7)  
  
for (i in 1:nrow(cp.class.param)) {  
 alpha <- cp.class.param[i, 'CP']  
 train.cm <- table(heart.train$AHD, predict(prune(class.cart, cp=alpha), newdata = heart.train, type='class'))  
 train.acc[i] <- 1-sum(diag(train.cm))/sum(train.cm)  
 cv.acc[i] <- cp.class.param[i, 'xerror'] \* cp.class.param[i, 'rel error']  
 test.cm <- table(heart.test$AHD, predict(prune(class.cart, cp=alpha), newdata = heart.test, type='class'))  
 test.acc[i] <- 1-sum(diag(test.cm))/sum(test.cm)  
}  
  
matplot(cp.class.param[,'nsplit'], cbind(train.acc, cv.acc, test.acc), pch=19, col=c("red", "black", "blue"), type="b", ylab="Loss", xlab="Depth")  
legend("right", c('Train', 'CV', 'Test') ,col=seq\_len(3),cex=0.8,fill=c("red", "black", "blue"))



* Tree size 5 with 8 leaf nodes looks good based on CV

prune.class.trees <- prune(class.cart, cp=cp.class.param[5,'CP'])  
prp(prune.class.trees)

 \* Calulate Test Error

conf.mat.tree <- table(heart.test$AHD, predict(prune.class.trees, type = 'class', newdata = heart.test))  
conf.mat.tree

##   
## No Yes  
## No 38 11  
## Yes 13 28

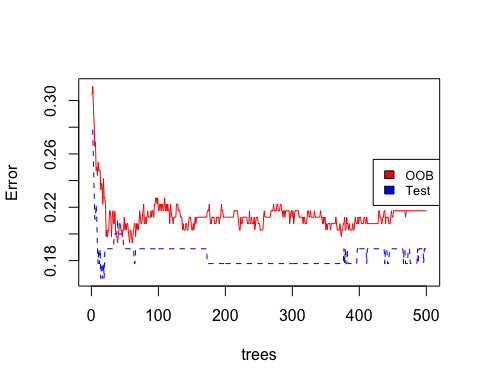
acc <- sum(diag(conf.mat.tree))/sum(conf.mat.tree)  
acc

## [1] 0.7333333

### Bagged Trees

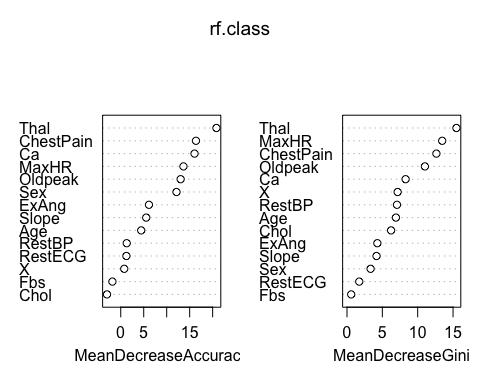
* Bagging is equivalent to Random Forest when the mtry parameter is set to the number of dimensions (p)

library(randomForest)  
set.seed(1)  
bag.class <- randomForest(AHD~., data=heart.train, mtry=ncol(heart.train) - 1, importance=TRUE, xtest=heart.test[,-15], ytest=heart.test$AHD)  
#plot(bag.class)  
err <- bag.class$err.rate[,1]  
bag.err <- cbind(err, bag.class$test$err.rate[,1])  
  
colnames(bag.err) <- c("OOB", "Test")  
matplot(1:bag.class$ntree, bag.err, type = "l", xlab="trees", ylab="Error", col = c("red", "blue"))  
legend("right", c('OOB', 'Test') ,col=seq\_len(2),cex=0.8,fill=c("red", "blue"))

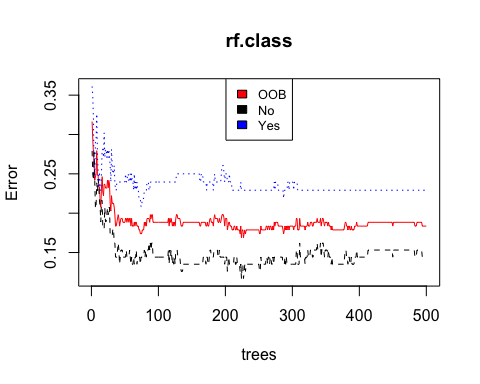


### Random Forest

set.seed(1)  
# setting mtry to sqrt(p) is a rule of thumb, this number can be set by  
# k fold CV as well  
rf.class <- randomForest(AHD~., data=heart.train,  
 mtry=round(sqrt(ncol(heart.train) - 1)), importance=TRUE)  
#importance(rf.class)  
varImpPlot(rf.class)

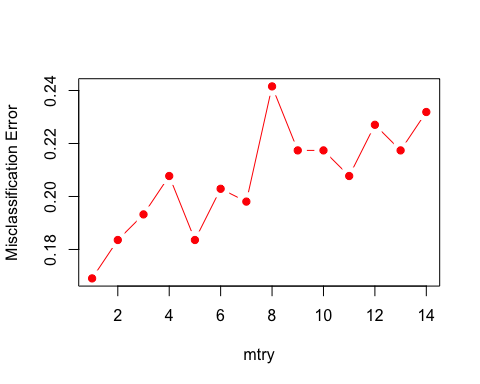


plot(rf.class, col=c("red", "black", "blue"))  
legend("top", colnames(rf.class$err.rate) ,col=seq\_len(3),cex=0.8,fill=c("red", "black", "blue"))



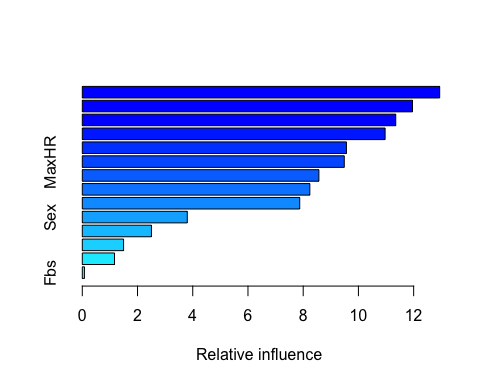
* ntree of around 175 looks good
* Try choosing mtry by plotting the OOB error

p <- ncol(heart.train) - 1  
oob.error.class <- double(p) #initialize empty vector  
set.seed(1)  
  
for(m in 1:p) {  
 fit <- randomForest(AHD ~ ., data=heart.train, mtry=m, ntree=175)  
 conf.mat <- fit$err.rate[175]  
 oob.error.class[m] <- fit$err.rate[175, 'OOB']  
}  
  
matplot(1:p, oob.error.class, pch=19, col="red", type="b", ylab="Misclassification Error", xlab="mtry")



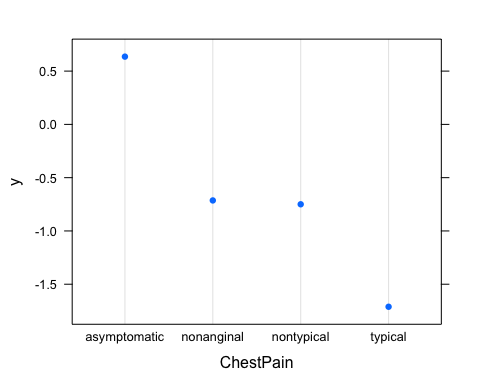
### Boosting

library(gbm)  
set.seed(1)  
#format y for gbm, must be 0/1  
AHD.0.1 <- ifelse(heart.train$AHD == 'Yes', 1, 0)  
class.boost = gbm(AHD.0.1 ~ . - AHD, data = heart.train, n.trees = 5000, distribution = "adaboost", shrinkage = 0.01)  
summary(class.boost)



## var rel.inf  
## ChestPain ChestPain 12.94269943  
## Thal Thal 11.95846003  
## RestBP RestBP 11.35167287  
## X X 10.96619306  
## Oldpeak Oldpeak 9.56593700  
## MaxHR MaxHR 9.48725353  
## Chol Chol 8.56820631  
## Ca Ca 8.24025603  
## Age Age 7.87417514  
## Sex Sex 3.80200228  
## Slope Slope 2.50436289  
## ExAng ExAng 1.49528278  
## RestECG RestECG 1.16677201  
## Fbs Fbs 0.07672663

par(mfrow=c(2,2))  
plot(class.boost, i="ChestPain")



plot(class.boost, i="Chol")

plot(class.boost, i="X")

plot(class.boost, i="Ca")