Bagging and Random Forests

YOUR NAME

This is from James, Witten, Hastie, and Ibshirani (2021) section 8.3 beginning on page 353.

```r
library(tree)
```

```
## Warning: package 'tree' was built under R version 4.4.1
```

```r
library(ISLR2)
data(Boston, package = 'ISLR2')
# These were created in the decision_trees lab
set.seed(1)
train <- sample(1:nrow(Boston), nrow(Boston) / 2)
boston.test <- Boston[-train, "medv"]
```

Here we apply bagging and random forests to the `Boston` data, using the `randomForest` package in `R`. The exact results obtained in this section may depend on the version of `R` and the version of the `randomForest` package installed on your computer. Recall that bagging is simply a special case of a random forest with $m = p$. Therefore, the `randomForest()` function can be used to perform both random forests and bagging. We perform bagging as follows:

```r
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.4.1
```

```
## randomForest 4.7-1.2
```

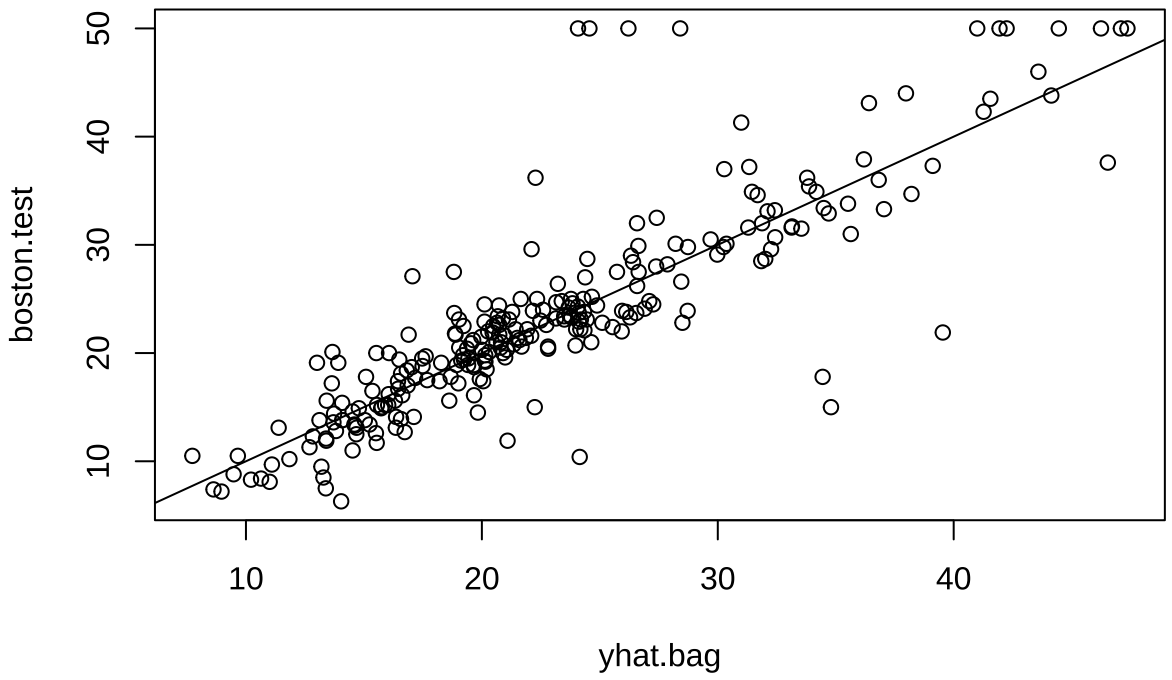
```
## Type rfNews() to see new features/changes/bug fixes.
```

```r
set.seed(1)
bag.boston <- randomForest(medv ~ ., data = Boston,
    subset = train, mtry = 12, importance = TRUE)
bag.boston
```

```
##
## Call:
##  randomForest(formula = medv ~ ., data = Boston, mtry = 12, importance = TRUE,      subset = train)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 12
##
##           Mean of squared residuals: 11.40162
##                     % Var explained: 85.17
```

The argument `mtry = 12` indicates that all 12 predictors should be considered for each split of the tree—in other words, that bagging should be done. How well does this bagged model perform on the test set?

```r
yhat.bag <- predict(bag.boston, newdata = Boston[-train, ])
plot(yhat.bag, boston.test)
abline(0, 1)
```



```r
mean((yhat.bag - boston.test)^2)
```

```
## [1] 23.41916
```

The test set MSE associated with the bagged regression tree is 23.42, about two-thirds of that obtained using an optimally-pruned single tree. We could change the number of trees grown by `randomForest()` using the `ntree` argument:

```r
bag.boston <- randomForest(medv ~ ., data = Boston,
    subset = train, mtry = 12, ntree = 25)
yhat.bag <- predict(bag.boston, newdata = Boston[-train, ])
mean((yhat.bag - boston.test)^2)
```

```
## [1] 25.75055
```

Growing a random forest proceeds in exactly the same way, except that we use a smaller value of the `mtry` argument. By default, `randomForest()` uses $p/3$ variables when building a random forest of regression trees, and $\sqrt{p}$ variables when building a random forest of classification trees. Here we use `mtry = 6`.

```
set.seed(1)
rf.boston <- randomForest(medv ~ ., data = Boston,
    subset = train, mtry = 6, importance = TRUE)
yhat.rf <- predict(rf.boston, newdata = Boston[-train, ])
mean((yhat.rf - boston.test)^2)
```

```
## [1] 20.06644
```

The test set MSE is 20.07; this indicates that random forests yielded an improvement over bagging in this case.

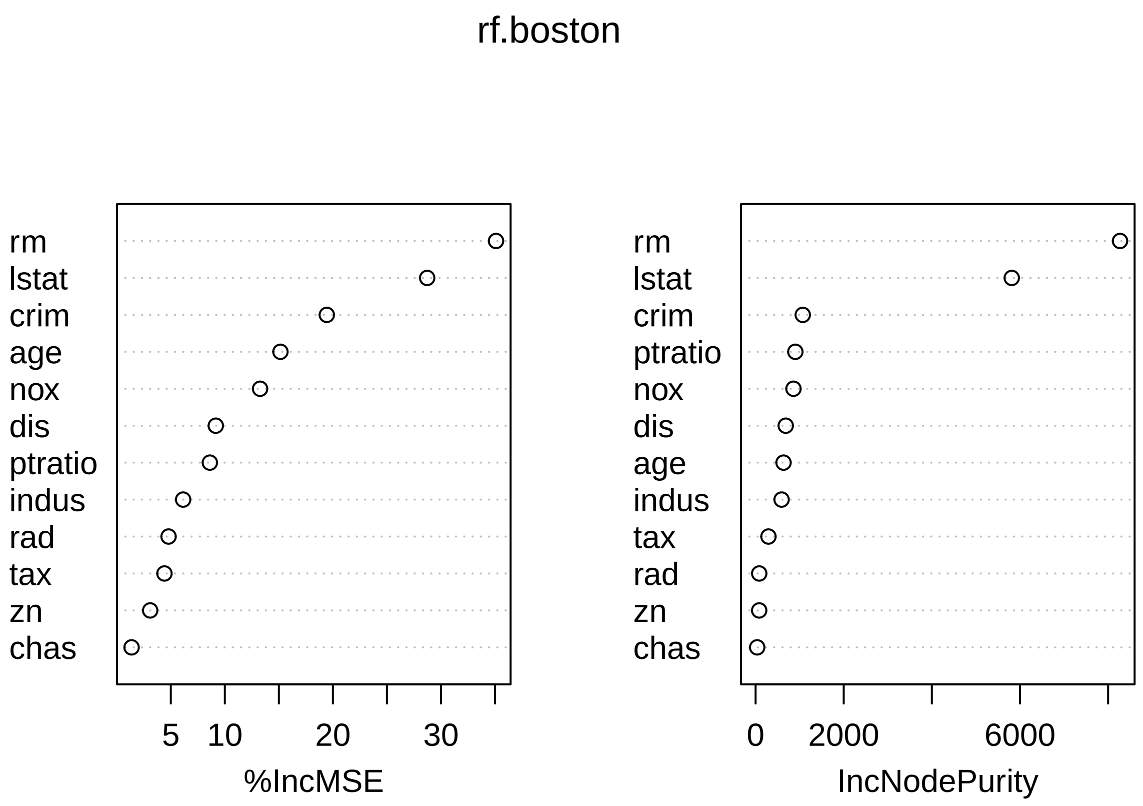
Using the `importance()` function, we can view the importance of each variable.

```
importance(rf.boston)
```

```
##            %IncMSE IncNodePurity
## crim    19.435587    1070.42307
## zn       3.091630      82.19257
## indus    6.140529     590.09536
## chas     1.370310      36.70356
## nox     13.263466     859.97091
## rm      35.094741    8270.33906
## age     15.144821     634.31220
## dis      9.163776     684.87953
## rad      4.793720      83.18719
## tax      4.410714     292.20949
## ptratio  8.612780     902.20190
## lstat   28.725343    5813.04833
```

Two measures of variable importance are reported. The first is based upon the mean decrease of accuracy in predictions on the out of bag samples when a given variable is permuted. The second is a measure of the total decrease in node impurity that results from splits over that variable, averaged over all trees (this was plotted in Figure 8.9). In the case of regression trees, the node impurity is measured by the training RSS, and for classification trees by the deviance. Plots of these importance measures can be produced using the `varImpPlot()` function.

```
varImpPlot(rf.boston)
```



The results indicate that across all of the trees considered in the random forest, the wealth of the community (`lstat`) and the house size (`rm`) are by far the two most important variables.

Boosting

Here we use the `gbm` package, and within it the `gbm()` function, to fit boosted regression trees to the `Boston` data set. We run `gbm()` with the option `distribution = "gaussian"` since this is a regression problem; if it were a binary classification problem, we would use `distribution = "bernoulli"`. The argument `n.trees = 5000` indicates that we want 5000 trees, and the option `interaction.depth = 4` limits the depth of each tree.

```
library(gbm)
```

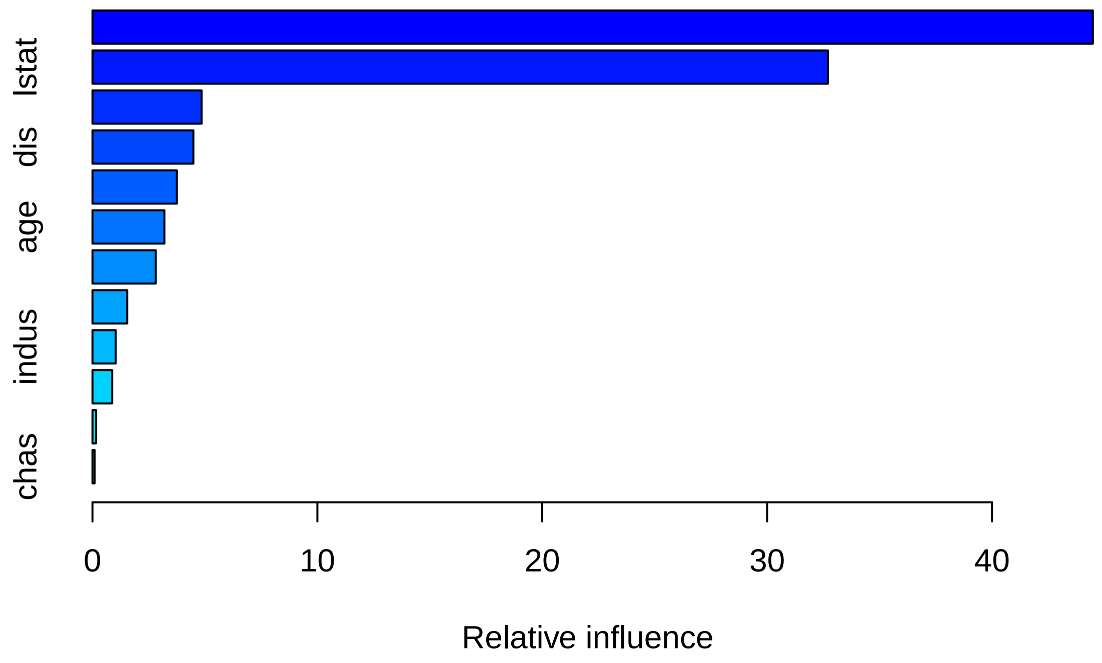
```
## Loaded gbm 2.2.2
```

```
## This version of gbm is no longer under development. Consider transitioning to gbm3, https://github.c
```

```
set.seed(1)
boost.boston <- gbm(medv ~ ., data = Boston[train, ],
    distribution = "gaussian", n.trees = 5000,
    interaction.depth = 4)
```

The `summary()` function produces a relative influence plot and also outputs the relative influence statistics.

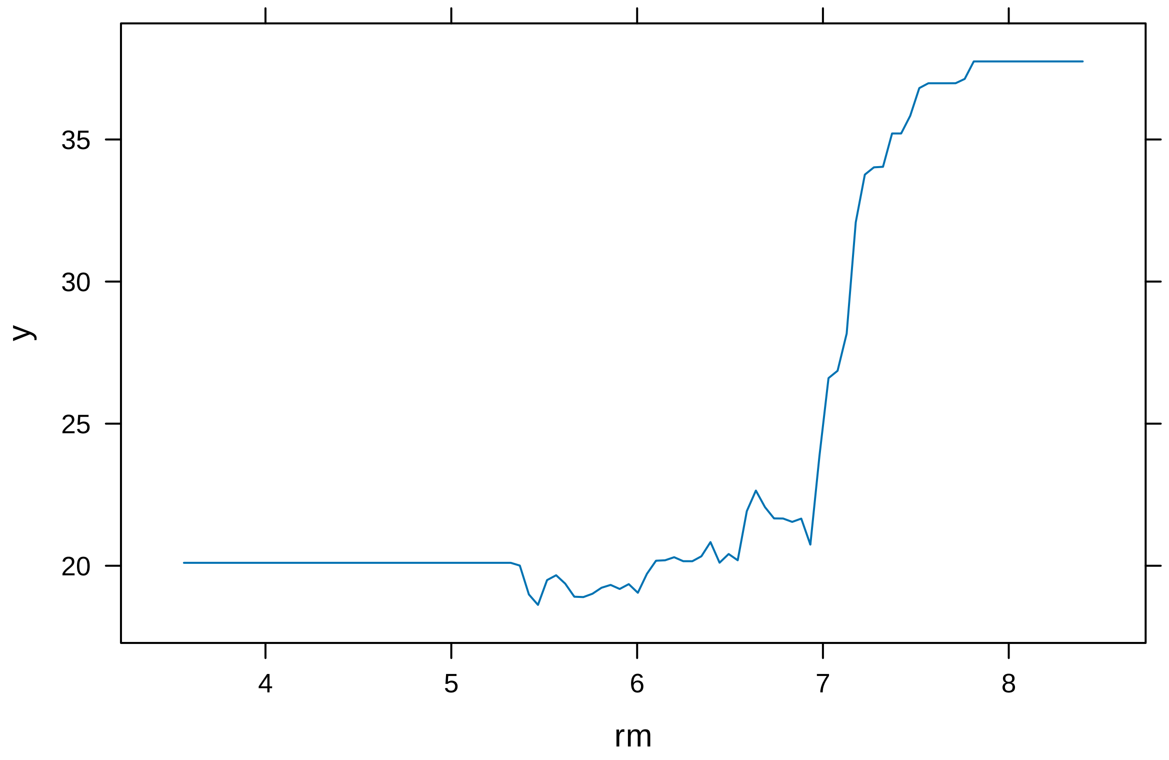
```r
summary(boost.boston)
```



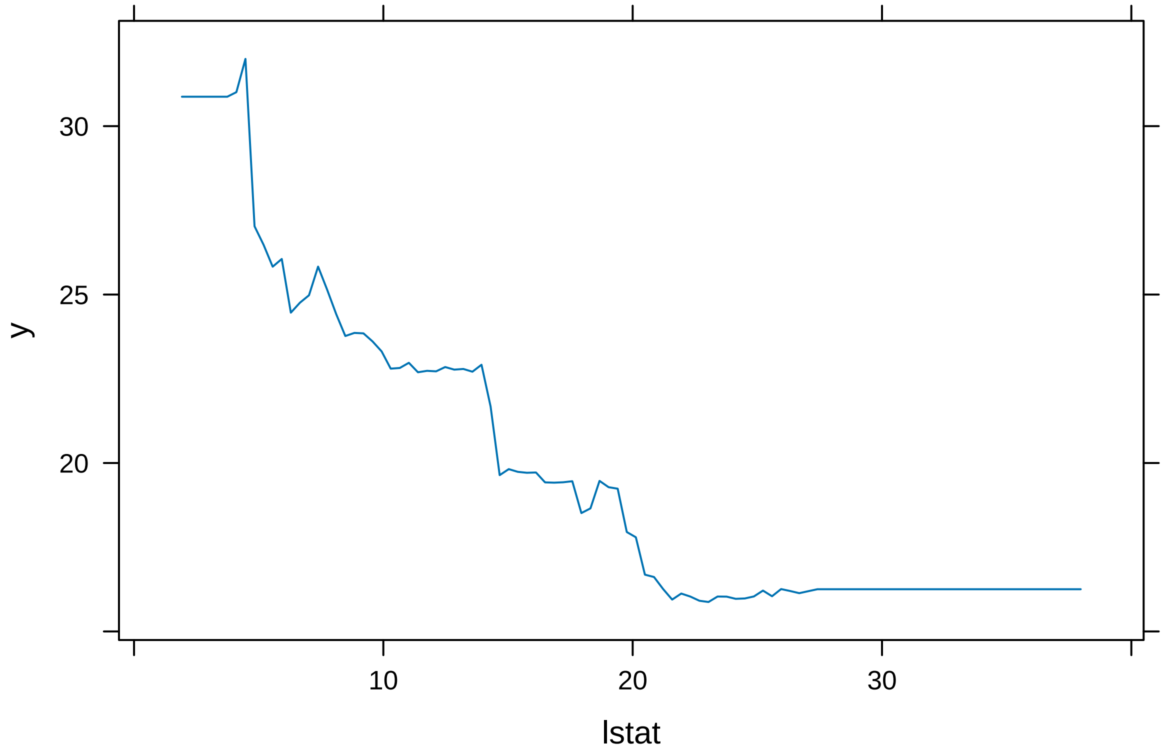
```
##               var     rel.inf
## rm             rm 44.48249588
## lstat       lstat 32.70281223
## crim         crim  4.85109954
## dis           dis  4.48693083
## nox           nox  3.75222394
## age           age  3.19769210
## ptratio ptratio  2.81354826
## tax           tax  1.54417603
## indus       indus  1.03384666
## rad           rad  0.87625748
## zn             zn  0.16220479
## chas         chas  0.09671228
```

We see that `lstat` and `rm` are by far the most important variables. We can also produce *partial dependence plots* for these two variables. These plots illustrate the marginal effect of the selected variables on the response after *integrating* out the other variables. In this case, as we might expect, median house prices are increasing with `rm` and decreasing with `lstat`.

```r
plot(boost.boston, i = "rm")
```



```r
plot(boost.boston, i = "lstat")
```



We now use the boosted model to predict `medv` on the test set:

```
yhat.boost <- predict(boost.boston,
    newdata = Boston[-train, ], n.trees = 5000)
mean((yhat.boost - boston.test)^2)
```

```
## [1] 18.39057
```

The test MSE obtained is 18.39: this is superior to the test MSE of random forests and bagging. If we want to, we can perform boosting with a different value of the shrinkage parameter λ in (8.10). The default value is 0.001, but this is easily modified. Here we take $\lambda = 0.2$.

```
boost.boston <- gbm(medv ~ ., data = Boston[train, ],
    distribution = "gaussian", n.trees = 5000,
    interaction.depth = 4, shrinkage = 0.2, verbose = F)
yhat.boost <- predict(boost.boston,
    newdata = Boston[-train, ], n.trees = 5000)
mean((yhat.boost - boston.test)^2)
```

```
## [1] 16.54778
```

In this case, using $\lambda = 0.2$ leads to a lower test MSE than $\lambda = 0.001$.

Bayesian Additive Regression Trees

In this section we use the `BART` package, and within it the `gbart()` function, to fit a Bayesian additive regression tree model to the `Boston` housing data set. The `gbart()` function is designed for quantitative outcome variables. For binary outcomes, `lbart()` and `pbart()` are available.

To run the `gbart()` function, we must first create matrices of predictors for the training and test data. We run BART with default settings.

```
library(BART)
```

```
## Loading required package: nlme
```

```
## Warning: package 'nlme' was built under R version 4.4.1
```

```
## Loading required package: survival
```

```
## Warning: package 'survival' was built under R version 4.4.1
```

```
x <- Boston[, 1:12]
y <- Boston[, "medv"]
xtrain <- x[train, ]
ytrain <- y[train]
xtest <- x[-train, ]
ytest <- y[-train]
set.seed(1)
bartfit <- gbart(xtrain, ytrain, x.test = xtest)
```

```
## *****Calling gbart: type=1
## *****Data:
## data:n,p,np: 253, 12, 253
## y1,yn: 0.213439, -5.486561
## x1,x[n*p]: 0.109590, 20.080000
## xp1,xp[np*p]: 0.027310, 7.880000
## *****Number of Trees: 200
## *****Number of Cut Points: 100 ... 100
## *****burn,nd,thin: 100,1000,1
## *****Prior:beta,alpha,tau,nu,lambda,offset: 2,0.95,0.795495,3,3.71636,21.7866
## *****sigma: 4.367914
## *****w (weights): 1.000000 ... 1.000000
## *****Dirichlet:sparse,theta,omega,a,b,rho,augment: 0,0,1,0.5,1,12,0
## *****printevery: 100
##
## MCMC
## done 0 (out of 1100)
## done 100 (out of 1100)
## done 200 (out of 1100)
## done 300 (out of 1100)
## done 400 (out of 1100)
## done 500 (out of 1100)
## done 600 (out of 1100)
## done 700 (out of 1100)
## done 800 (out of 1100)
## done 900 (out of 1100)
## done 1000 (out of 1100)
## time: 2s
## trcnt,tecnt: 1000,1000
```

Next we compute the test error.

```
yhat.bart <- bartfit$yhat.test.mean
mean((ytest - yhat.bart)^2)
```

```
## [1] 15.91912
```

On this data set, the test error of BART is lower than the test error of random forests and boosting.

Now we can check how many times each variable appeared in the collection of trees.

```
ord <- order(bartfit$varcount.mean, decreasing = T)
bartfit$varcount.mean[ord]
```

```
##     nox   lstat     rad      rm     tax ptratio    chas     age   indus      zn
##  22.973  21.653  21.638  20.725  20.021  19.615  19.283  19.278  19.073  15.576
##     dis    crim
##  13.800  11.607
```

On Your Own

1. In the lab, a classification tree was applied to the Carseats data set after converting Sales into a qualitative response variable. Now we will seek to predict Sales using regression trees and related approaches, treating the response as a quantitative variable.

(a) Split the data set into a training set and a test set.

(b) Fit a regression tree to the training set. Plot the tree, and interpret the results. What test MSE do you obtain?

(c) Use cross-validation in order to determine the optimal level of tree complexity. Does pruning the tree improve the test MSE?

(d) Use the bagging approach in order to analyze this data. What test MSE do you obtain? Use the importance() function to determine which variables are most important.

(e) Use random forests to analyze this data. What test MSE do you obtain? Use the importance() function to determine which variables are most important. Describe the effect of m, the number of variables considered at each split, on the error rate obtained.

(f) Now analyze the data using BART, and report your results.

2. This question uses the Caravan data set.

```r
data(Caravan, package = 'ISLR2')
```

(a) Create a training set consisting of the first 1,000 observations, and a test set consisting of the remaining observations.

(b) Fit a boosting model to the training set with Purchase as the response and the other variables as predictors. Use 1,000 trees, and a shrinkage value of 0.01. Which predictors appear to be the most important?

(c) Use the boosting model to predict the response on the test data. Predict that a person will make a purchase if the estimated probability of purchase is greater than 20 %. Form a confusion matrix. What fraction of the people predicted to make a purchase do in fact make one? How does this compare with the results obtained from applying KNN or logistic regression to this data set?