Chapter 8 Problem 12

Andira Putri

Apply boosting, bagging, and random forests to a data set of your own choice. Be sure to fit the models on a training set and to evaluate their performance on a test set. How accurate are the results compared to simple methods like linear or logistic regression? Which of these approaches yields the best performance?

I will apply the three tree-based methods to the `Boston` data set, just because I've become friendly with it in the past week :) I want to study crime rate, so `crim` will be my response.

First, I fit a linear regression model to the data.

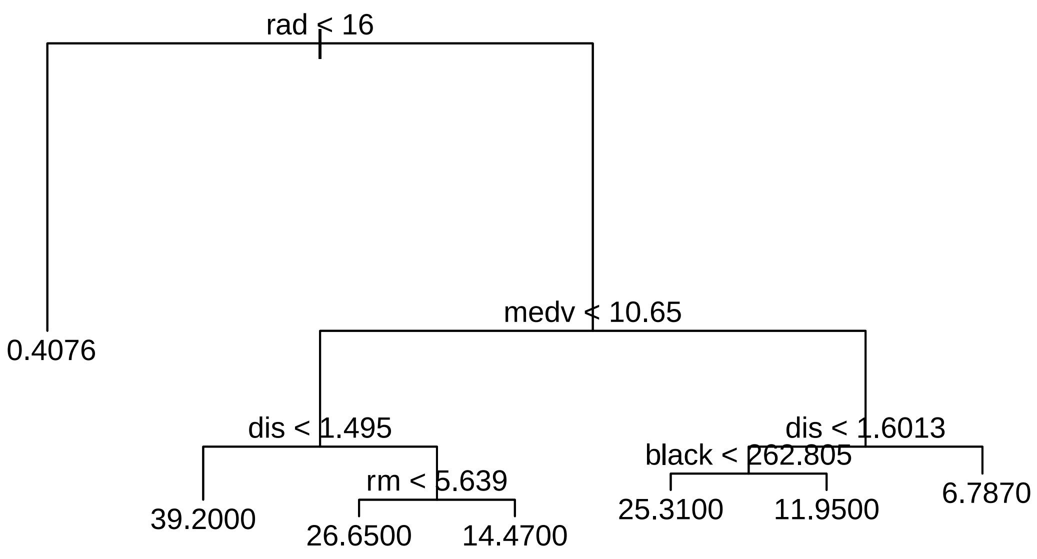
```r
library(MASS)
data(Boston)
#split the data into training and test set
set.seed(1)
train=sample(506,400) #400 obs for training
train.set=Boston[train,]
test.set=Boston[-train,] #106 obs for testing
#fit lin regression model
lin.reg=lm(crim~.,data=train.set)
lin.pred=predict(lin.reg,test.set)
#MSE
mean((lin.pred-test.set$crim)^2)
```

```
## [1] 44.68449
```

Our least squares linear regression MSE is 44.68449. Let's compare this with our decision trees.

Now, I want to generate a classification tree without applying boosting, bagging, or random forests.

```r
#generate classification tree
library(tree)
tree=tree(crim~.,train.set)
plot(tree)
text(tree,pretty=0,cex=0.9)
```



```r
#make predictions using this tree
tree.pred=predict(tree,test.set)
#compute test error with MSE formula
mean((tree.pred-test.set$crim)^2)
```

```
## [1] 46.66706
```

We get the above tree as a prediction tool for `crim`. To explain this tree, say that our accessibility to radial highways `rad` is higher than 16. Then, we go down to study median value of homes `medv`. If it is less than 10.65 ($10,650), we go down the left branch. If our distance to the Boston employment centers `dis` is 1.495 (miles I assume), then our predicted crime rate `crim` is 39.2000. Our regression tree has an MSE of 46.66706 which is slightly smaller than linear regression.

These simple regression trees have many advantages. It is easy to explain, resembles human decision making, can be displayed visually, and can easily handle qualitative predictors without dummy variables.

Of course, they have disadvantages including:

- These simple regression trees generally do not have the same level of predictive accuracy as other approaches.
- They can be very non-robust, meaning small changes in data lead to large changes in prediction

Methods like bagging, random forests, and boosting seek to override these issues.

Bagging

Decision trees suffer from high variance. Averaging a set of observations reduces that variance. The algorithm for bagging is:

1. Bootstrap by taking repeated samples from the training set.
2. Do this B times. Generate B different bootstrapped training data sets, or B different trees.
3. Train method on the b-th bootstrapped training set in order to get $\hat{f}^{*b}(x)$..
4. Average all the predictions to get $\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x)$.

Using a large B value is not dangerous for overfitting- just make sure it is sufficiently large.

```
library(randomForest)
```

```
## randomForest 4.6-14

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

```
bag=randomForest(crim~.,train.set,mtry=13,importance=TRUE)
#mtry=13 b/c there are 13 predictors to consider
importance(bag)
```

```
##            %IncMSE IncNodePurity
## zn       9.8283785      5.354508
## indus    2.7713580     34.166836
## chas    -1.8316728      2.875688
## nox      4.6676200    338.891993
## rm       4.5879720   2069.152677
## age      5.9491613    619.068043
## dis      0.6159651   4942.345027
## rad     22.5043518   7303.230822
## tax      2.4142350     84.395019
## ptratio  5.2738844     51.606447
## black   -2.6018659   1277.765993
## lstat    9.5443030   1889.008966
## medv     9.8467579   9779.840981
```

```
#Computing MSE
bag.predict=predict(bag,test.set) #yhat
mean((bag.predict-test.set$crim)^2)
```

```
## [1] 41.10399
```

Our test MSE from bagging, 41.10399, is much smaller than that of the simple regression tree, denoting a significant impact using this method. We're getting somewhere!

Random Forests

Random forests aim to decorrelate the trees. Random forests is almost like bagging except a random sample of m predictors is chosen as split candidates from a full set of predictors (when m=p, it's just bagging). m is typically the square root of p. The reason why predictors are chosen at random is to avoid always choosing the strongest predictors. Choosing the strongest predictors all the time leads to trees looking the same, predictions that are highly correlated, and a small reduction of variance. Random forests override these disadvantages which are typical in bagging.

```
rf=randomForest(crim~.,train.set,mtry=4,importance=TRUE)
#mtry=4, so I only consider 4 randomly-chosen predictors per split
importance(rf)
```

```
##            %IncMSE IncNodePurity
## zn       8.1983122      1.778191
## indus    7.7956475    789.990882
## chas     0.4538274    21.415115
## nox      6.5862180  1679.022415
## rm       4.4377401  2048.773523
## age      4.4508302  1271.247553
## dis      4.8809539  4490.140688
## rad     16.6795053  4230.755944
## tax     10.2579530  2784.741803
## ptratio  7.1416019   274.937389
## black    1.1744646  1226.450412
## lstat    8.6377363  2469.996638
## medv     7.4664157  6145.704651
```

```r
rf.predict=predict(rf,test.set)
mean((rf.predict-test.set$crim)^2)
```

```
## [1] 40.08297
```

So far, random forest has given us our lowest test MSE of 40.08297.

Boosting

Boosting works like bagging except trees are grown sequentially and does not involve bootstrapping. Boosting fits a tree using residuals rather than an outcome Y, and it is dependent on previous trees.

```r
set.seed(1)
library(gbm)
```

```
## Loaded gbm 2.1.4
```

```r
boost=gbm(crim~.,train.set,distribution="gaussian",n.trees=500,interaction.depth=4)
boost.pred=predict(boost,test.set,n.trees=500)
mean((boost.pred-test.set$crim)^2)
```

```
## [1] 41.61091
```

Our boosting test MSE is 41.61091, which is higher than our MSE from bagging. Random forest gave us the best tree model!