DSE6011 - Module 7 - Ch 8 Exercises

Nathan Monges

2024-08-17

```r
library(tree)
library(randomForest)
```

```
## randomForest 4.7-1.1

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

```r
library(gbm)
```

```
## Warning: package 'gbm' was built under R version 4.3.3

## Loaded gbm 2.2.2

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

```r
library(tidymodels)
```

```
## -- Attaching packages -------------------------------------- tidymodels 1.2.0 --

## v broom        1.0.5      v recipes      1.0.10
## v dials        1.2.1      v rsample      1.2.1
## v dplyr        1.1.4      v tibble       3.2.1
## v ggplot2      3.5.0      v tidyr        1.3.1
## v infer        1.0.7      v tune         1.2.1
## v modeldata    1.4.0      v workflows    1.1.4
## v parsnip      1.2.1      v workflowsets 1.1.0
## v purrr        1.0.2      v yardstick    1.3.1

## Warning: package 'modeldata' was built under R version 4.3.3

## -- Conflicts ----------------------------------------- tidymodels_conflicts() --
## x dplyr::combine()  masks randomForest::combine()
## x purrr::discard()  masks scales::discard()
## x dplyr::filter()   masks stats::filter()
## x dplyr::lag()      masks stats::lag()
## x ggplot2::margin() masks randomForest::margin()
## x recipes::step()   masks stats::step()
## * Dig deeper into tidy modeling with R at https://www.tmwr.org
```

```r
library(BART)
```

```
## Warning: package 'BART' was built under R version 4.3.3

## Loading required package: nlme

##
## Attaching package: 'nlme'

## The following object is masked from 'package:dplyr':
```

```
##
##     collapse

## Loading required package: survival
```

```
library(ISLR2)
library(MASS)
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:ISLR2':
##
##     Boston

## The following object is masked from 'package:dplyr':
##
##     select
```

```
library(glmnet)
```

```
## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

## Loaded glmnet 4.1-8
```

Exercise 8

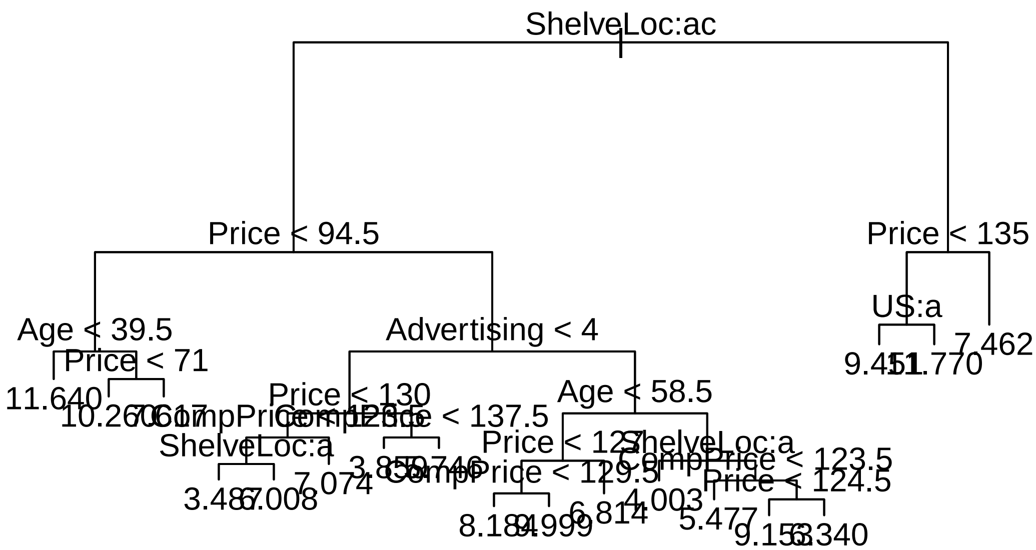
(a) Split the data set into a training set and a test set.

```
set.seed(1)
train <- sample(1:nrow(Carseats), 200)
test <- Carseats[-train, ]
```

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

```
tree.carseats <- tree(Sales ~ ., Carseats, subset = train)
tree.pred <- predict(tree.carseats, test)

plot(tree.carseats)
text(tree.carseats)
```




```r
sales.test <- Carseats[-train, "Sales"]
mean((tree.pred - sales.test)^2) #test mse
```

```
## [1] 4.922039
```

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

```r
cv.sales <- cv.tree(tree.carseats) #18 t nodes gives lowest dev
cv.sales
```

```
## $size
##  [1] 18 17 16 15 14 13 12 11 10  8  7  6  5  4  3  2  1
##
## $dev
##  [1]  831.3437  852.3639  868.6815  862.3400  862.3400  893.4641  911.2580
##  [8]  950.2691  955.2535 1039.1241 1066.6899 1125.0894 1205.5806 1273.2889
## [15] 1302.8607 1349.9273 1620.4687
##
## $k
##  [1]      -Inf  16.99544  20.56322  25.01730  25.57104  28.01938  30.36962
##  [8]  31.56747  31.80816  40.75445  44.44673  52.57126  76.21881  99.59459
## [15] 116.69889 159.79501 337.60153
##
## $method
## [1] "deviance"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```r
prune.sales <- prune.tree(tree.carseats, best = 18)
prune.pred <- predict(prune.sales, test)
mean((prune.pred - sales.test)^2)
```

```
## [1] 4.922039
```

Cross validation in the tree shows that the optimla level of tree complexity is with 18, terminal nodes, which is the same as the baseline tree in the previos example. These results give the same 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.

```
set.seed(2)
sales.bag <- randomForest(Sales ~ ., Carseats, subset = train,
                          mtry = 11, importance = TRUE)
```

```
## Warning in randomForest.default(m, y, ...): invalid mtry: reset to within valid
## range
```

```
yhat.bag <- predict(sales.bag, newdata = test)
mean((yhat.bag - sales.test)^2)
```

```
## [1] 2.586535
```

```
importance(sales.bag)
```

```
##                %IncMSE IncNodePurity
## CompPrice   25.7444826     170.66690
## Income       4.7699543      90.24851
## Advertising 12.3934240     104.10313
## Population  -2.0752891      59.47636
## Price       55.2142251     505.28250
## ShelveLoc   47.2885836     376.66846
## Age         17.0744805     156.14233
## Education    2.1151607      45.21764
## Urban       -0.9618328       8.82340
## US           5.3624960      18.75708
```

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.

```
set.seed(3)
sales.rf <- randomForest(Sales ~ ., Carseats, subset = train,
                          mtry = 4, importance = TRUE) #using p/3 predictors
yhat.rf <- predict(sales.rf, newdata = test)
mean((yhat.rf - sales.test)^2)
```

```
## [1] 2.767523
```

```
importance(sales.rf)
```

```
##                %IncMSE IncNodePurity
## CompPrice   17.6331422     156.63313
## Income       4.3070432     123.34376
## Advertising 10.5447183     105.29972
## Population  -3.0173206      85.60279
## Price       41.8071333     434.89549
## ShelveLoc   39.2121869     319.15366
## Age         13.4325285     170.17609
## Education    0.6734260      62.70388
## Urban       -0.8419159      13.75924
## US           5.8004774      29.65185
```

Random forests seems to be giving less acurate results compare to bagging in this scenerio with an MSE of 2.76 whereas bagging gave a test MSE of 2.58. In the rf regression I used m = 4 predictors for the random forests to subset as for regressiion m = p/3. Adjusting m affects the bias-variance tradeoff where a lower m value may increase the trees diversity of predictors which may result in underfittingm but our test MSE shows that the model seems to be doing pretty well at m = p/3.

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

```
library(BART)
x <- Carseats[, 1:11]
y <- Carseats[, "Sales"]
xtrain <- x[train, ]
ytrain <- y[train]

xtest <- x[-train, ]
ytest <- y[-train]
set.seed(1)
bartfit.sales <- gbart(xtrain, ytrain, x.test = xtest)
```

```
## *****Calling gbart: type=1
## *****Data:
## data:n,p,np: 200, 15, 200
## y1,yn: 2.781850, 1.091850
## x1,x[n*p]: 10.360000, 1.000000
## xp1,xp[np*p]: 11.220000, 1.000000
## *****Number of Trees: 200
## *****Number of Cut Points: 100 ... 1
## *****burn,nd,thin: 100,1000,1
## *****Prior:beta,alpha,tau,nu,lambda,offset: 2,0.95,0.273474,3,6.80272e-30,7.57815
## *****sigma: 0.000000
## *****w (weights): 1.000000 ... 1.000000
## *****Dirichlet:sparse,theta,omega,a,b,rho,augment: 0,0,1,0.5,1,15,0
## *****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
```

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

```
## [1] 0.1603478
```

Exercise 9

a) Create a training set containing a random sample of 800 observations, and a test set containing the remaining observations.

```
attach(OJ)

set.seed(1)
```

```r
train <- sample(1:nrow(OJ), 800)
oj_train <- OJ[train, ]

oj_test <- OJ[-train, ]
```

b) Fit a tree to the training data, with Purchase as the response and the other variables as predictors. Use the summary() function to produce summary statistics about the tree, and describe the results obtained. What is the training error rate? How many terminal nodes does the tree have?

```r
tree.oj <- tree(Purchase ~ ., OJ, subset = train)

summary(tree.oj)
```

```
##
## Classification tree:
## tree(formula = Purchase ~ ., data = OJ, subset = train)
## Variables actually used in tree construction:
## [1] "LoyalCH"       "PriceDiff"     "SpecialCH"     "ListPriceDiff"
## [5] "PctDiscMM"
## Number of terminal nodes:  9
## Residual mean deviance:  0.7432 = 587.8 / 791
## Misclassification error rate: 0.1588 = 127 / 800
```

The training error rate is 0.1588 and there are 9 terminal nodes in the tree model.

c) Type in the name of the tree object in order to get a detailed text output. Pick one of the terminal nodes, and interpret the information displayed.

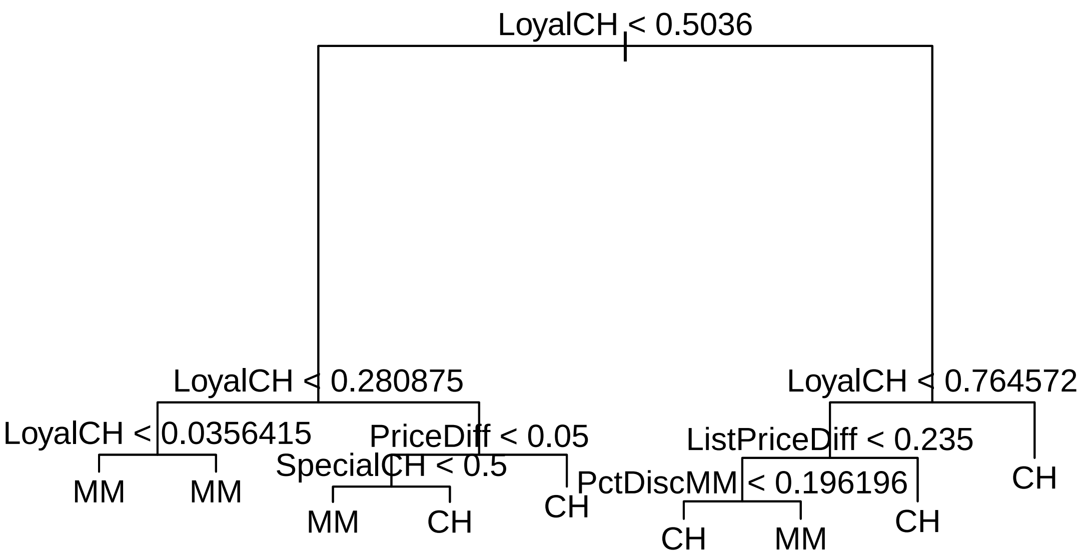
```r
tree.oj
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 800 1073.00 CH ( 0.60625 0.39375 )
##    2) LoyalCH < 0.5036 365  441.60 MM ( 0.29315 0.70685 )
##      4) LoyalCH < 0.280875 177  140.50 MM ( 0.13559 0.86441 )
##        8) LoyalCH < 0.0356415 59   10.14 MM ( 0.01695 0.98305 ) *
##        9) LoyalCH > 0.0356415 118  116.40 MM ( 0.19492 0.80508 ) *
##      5) LoyalCH > 0.280875 188  258.00 MM ( 0.44149 0.55851 )
##       10) PriceDiff < 0.05 79   84.79 MM ( 0.22785 0.77215 )
##         20) SpecialCH < 0.5 64   51.98 MM ( 0.14062 0.85938 ) *
##         21) SpecialCH > 0.5 15   20.19 CH ( 0.60000 0.40000 ) *
##       11) PriceDiff > 0.05 109  147.00 CH ( 0.59633 0.40367 ) *
##    3) LoyalCH > 0.5036 435  337.90 CH ( 0.86897 0.13103 )
##      6) LoyalCH < 0.764572 174  201.00 CH ( 0.73563 0.26437 )
##       12) ListPriceDiff < 0.235 72   99.81 MM ( 0.50000 0.50000 )
##         24) PctDiscMM < 0.196196 55   73.14 CH ( 0.61818 0.38182 ) *
##         25) PctDiscMM > 0.196196 17   12.32 MM ( 0.11765 0.88235 ) *
##       13) ListPriceDiff > 0.235 102   65.43 CH ( 0.90196 0.09804 ) *
##      7) LoyalCH > 0.764572 261   91.20 CH ( 0.95785 0.04215 ) *
```

From the information displayed in the tree mode, the second terminal node split is LoyalCH < 0.5036 with 365 observation in the branch. The majority class is "MM" with 70.69% of observation being "MM",

d) Create a plot of the tree, and interpret the results.

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



e) Predict the response on the test data, and produce a confusion matrix comparing the test labels to the predicted test labels. What is the test error rate?

```r
oj.pred <- predict(tree.oj, oj_test, type = "class")
table(oj.pred, oj_test$Purchase)
```

```
##
## oj.pred  CH  MM
##      CH 160  38
##      MM   8  64
```

```r
(160 + 64) / 270
```

```
## [1] 0.8296296
```

f) Apply the cv.tree() function to the training set in order to determine the optimal tree size.

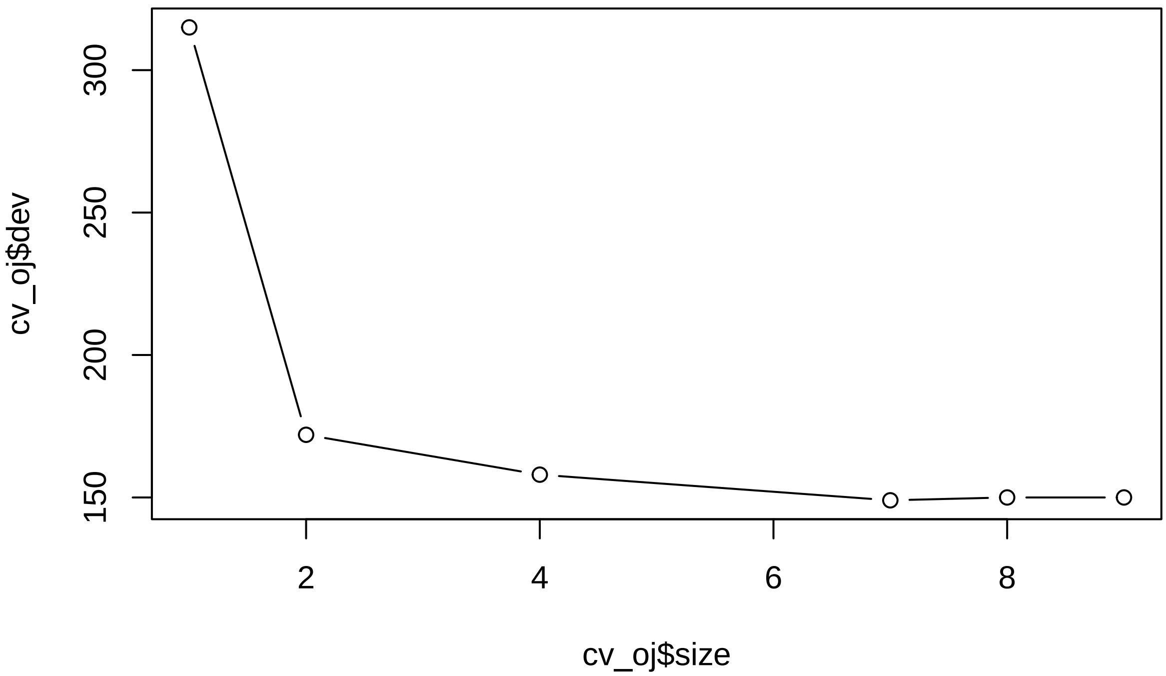
```r
cv_oj <- cv.tree(tree.oj, FUN = prune.misclass)
cv_oj
```

```
## $size
## [1] 9 8 7 4 2 1
##
## $dev
## [1] 150 150 149 158 172 315
##
## $k
## [1]       -Inf   0.000000   3.000000   4.333333  10.500000 151.000000
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

From the cross validation on the oj.tree model it is evident that the optima tree size is one with 7 terminal nodes since it has the lwoest deviance of 149.

g) Produce a plot with tree size on the x-axis and cross-validated classification error rate on the y-axis.

```
plot(cv_oj$size,  cv_oj$dev, type = "b")
```



h) Which tree size corresponds to the lowest cross-validated classification error rate?

A tree with 7 terminal nodes gives the lowest crossvalidated error rate.

i) Produce a pruned tree corresponding to the optimal tree size obtained using cross-validation. If cross-validation does not lead to selection of a pruned tree, then create a pruned tree with five terminal nodes.

```
prune.oj <- prune.tree(tree.oj, best = 7)
prune.pred.oj <- predict(prune.oj, oj_test, type = "class")
table(prune.pred.oj, oj_test$Purchase)
```

```
##
## prune.pred.oj  CH  MM
##            CH 160  36
##            MM   8  66
```

```
(160 + 66) / 270
```

```
## [1] 0.837037
```

j) Compare the training error rates between the pruned and unpruned trees. Which is higher?

The pruned tree gives slightly more accurate predictions with a test MSE of 0.83 which is greater than the unpruned tree with a test MSE of 0.82.

k) Compare the test error rates between the pruned and unpruned trees. Which is higher?

The pruned tree gives slightly more accurate predictions with a test MSE of 0.83 which is greater than the unpruned tree with a test MSE of 0.82.

Exercise 10

a) Remove the observations for whom the salary information is unknown, and then log-transform the salaries.

```
Hitters <- Hitters

Hitters <- Hitters[!is.na(Hitters$Salary), ]

Hitters$log_sal <- log(Hitters$Salary)
```

b) Create a training set consisting of the first 200 observations, and a test set consisting of the remaining observations.

```
set.seed(2)
hitters.train <- Hitters[1:200, ]
hitters.test <- Hitters[201:nrow(Hitters), ]
```

c) Perform boosting on the training set with 1,000 trees for a range of values of the shrinkage parameter . Produce a plot with different shrinkage values on the x-axis and the corresponding training set MSE on the y-axis.

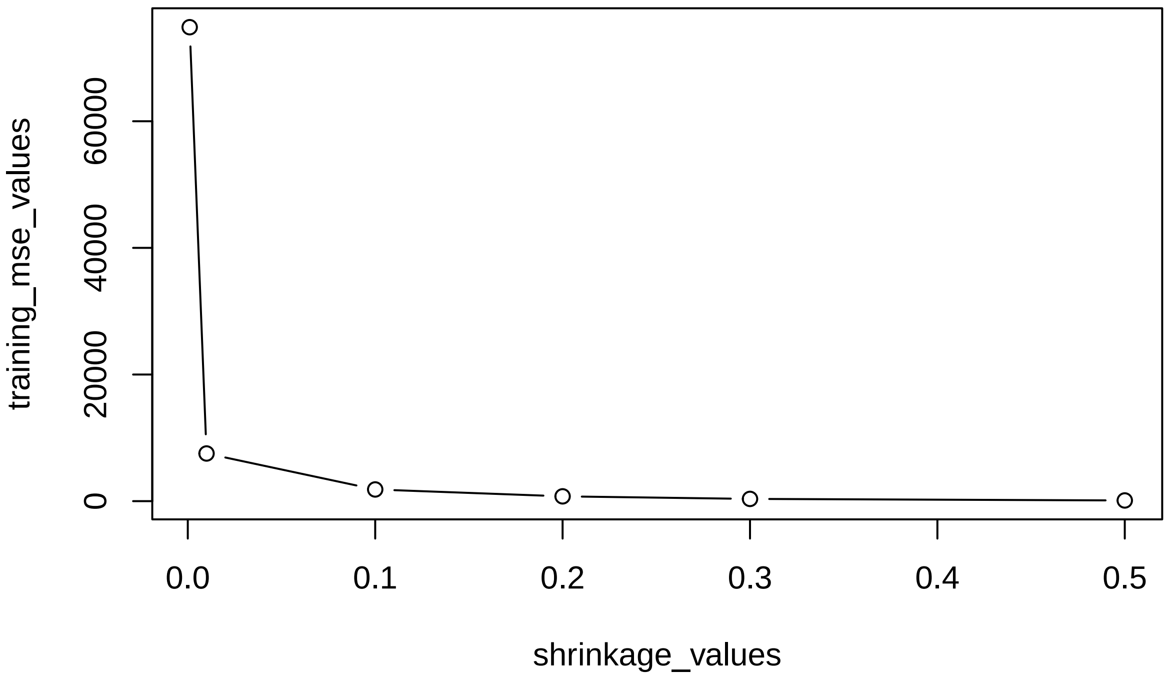
```
set.seed(1)

shrinkage_values <- c(0.001, 0.01, 0.1, 0.2, 0.3, 0.5)

training_mse_values <- numeric(length(shrinkage_values))

for (i in seq_along(shrinkage_values)) {
  hitters.boost <- gbm(Salary ~ ., data = hitters.train,
             distribution = "gaussian",
             n.trees = 1000,
             shrinkage = shrinkage_values[i],
             cv.folds = 0)
  predictions <- predict(hitters.boost, hitters.train, n.trees = 1000)
  training_mse_values[i] <- mean((hitters.train$Salary - predictions)^2)
}

plot(shrinkage_values, training_mse_values, type = "b")
```



d) Produce a plot with different shrinkage values on the x-axis and the corresponding test set MSE on the y-axis.

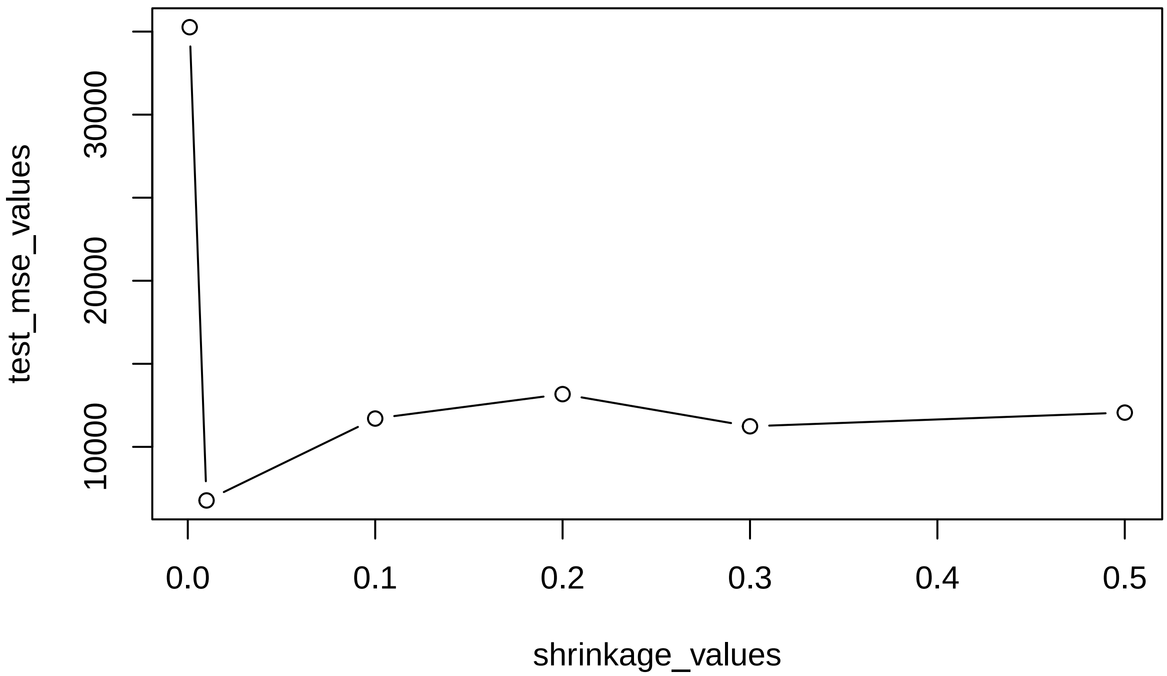
```r
set.seed(1)

shrinkage_values <- c(0.001, 0.01, 0.1, 0.2, 0.3, 0.5)

test_mse_values <- numeric(length(shrinkage_values))

for (i in seq_along(shrinkage_values)) {
  hitters.boost <- gbm(Salary ~ ., data = hitters.train,
                       distribution = "gaussian",
                       n.trees = 1000,
                       shrinkage = shrinkage_values[i],
                       cv.folds = 0)
  predictions <- predict(hitters.boost, hitters.test, n.trees = 1000)
  test_mse_values[i] <- mean((hitters.test$Salary - predictions)^2)
}

plot(shrinkage_values, test_mse_values, type = "b")
```



e) Compare the test MSE of boosting to the test MSE that results from applying two of the regression approaches seen in Chapters 3 and 6.

Linear Regression

```
hiiter.lm <- lm(Salary ~ ., data = hitters.train)

lm.pred <- predict(hiiter.lm, hitters.test)

lm.mse <- mean((hitters.test$Salary - lm.pred)^2)
lm.mse
```

```
## [1] 25671.77
```

Ridge Regression

```
x_train <- model.matrix(Salary ~ ., hitters.train)[, -1]
y_train <- hitters.train$Salary
x_test <- model.matrix(Salary ~ ., hitters.test)[, -1]
y_test <- hitters.test$Salary

hitters.ridge <- glmnet(x_train, y_train, alpha = 0)

ridge_cv <- cv.glmnet(x_train, y_train, alpha = 0)
ridge_best_lambda <- ridge_cv$lambda.min
ridge_pred<- predict(hitters.ridge, s = ridge_best_lambda, newx = x_test)

ridge_mse <- mean((y_test - ridge_pred)^2)
ridge_mse
```

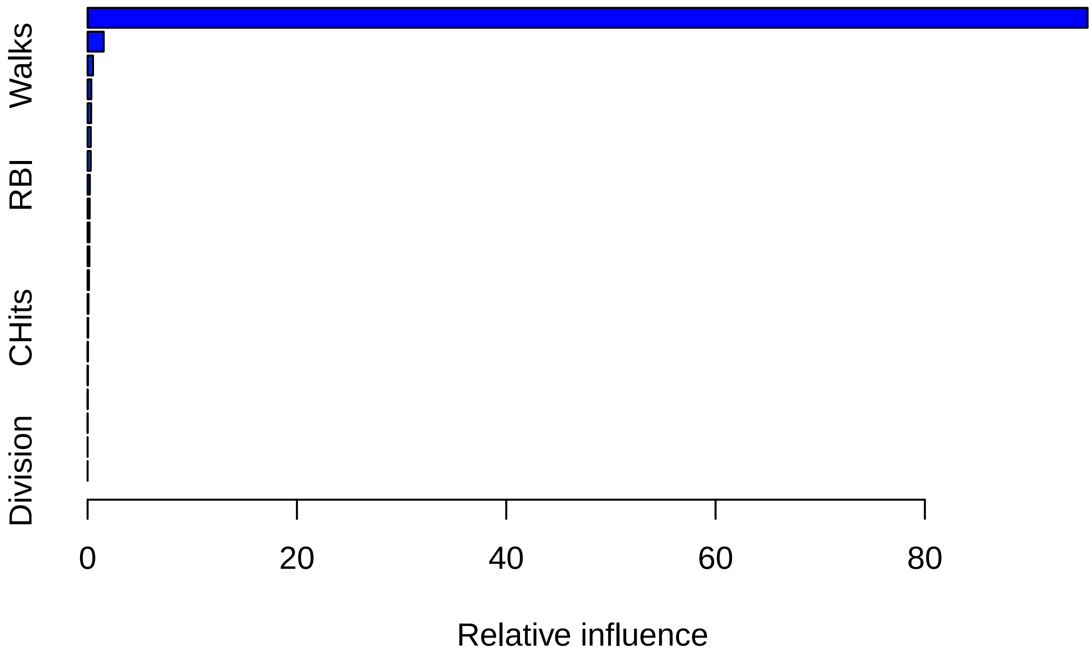
```
## [1] 22435.18
```

f) Which variables appear to be the most important predictors in the boosted model?

```
set.seed(1)
hitters_boost <- gbm(Salary ~ ., data = hitters.train,
```

```r
                     distribution = "gaussian",
                     n.trees = 1000,
                     shrinkage = 0.01)

summary(hitters_boost)
```




```
##                 var      rel.inf
## log_sal     log_sal 95.544435081
## CHmRun       CHmRun  1.539665915
## Walks         Walks  0.517006328
## PutOuts     PutOuts  0.354206461
## Hits           Hits  0.335534795
## HmRun         HmRun  0.305691009
## CRBI           CRBI  0.299709909
## RBI             RBI  0.213106408
## CWalks       CWalks  0.202228969
## AtBat         AtBat  0.181865959
## CAtBat       CAtBat  0.170949297
## CRuns         CRuns  0.130009679
## Runs           Runs  0.084030327
## CHits         CHits  0.059506972
## Assists     Assists  0.025810291
## Errors       Errors  0.020435896
## Years         Years  0.012497069
## NewLeague NewLeague  0.003309634
## League       League  0.000000000
## Division   Division  0.000000000
```

log_sal and CHmRun seem to be the most important vairables.

g) Now apply bagging to the training set. What is the test set MSE for this approach?

```r
set.seed(2)
hitters.bag <- randomForest(Salary ~ ., hitters.train,
```

```r
                              mtry = 20, importance = TRUE)
hitters.yhat<- predict(hitters.bag, newdata = hitters.test)
mean((hitters.yhat - hitters.test$Salary)^2)
```

```
## [1] 398.7028
```