HW9

Jingshi

4/7/2017

Question 8.4 #4

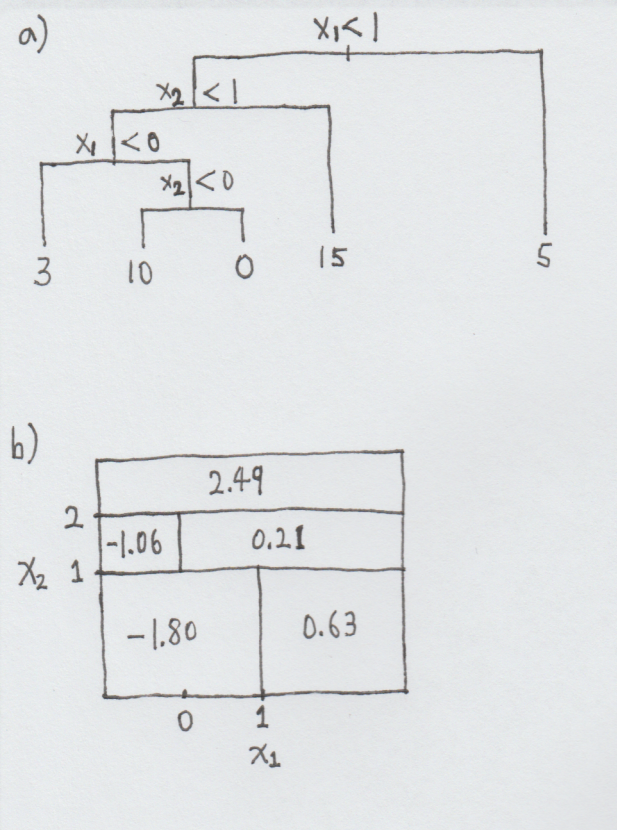


Figure 1: #4 Answers

Question 8.4 #7

```
#install.packages("randomForest")
library(MASS)
library(randomForest)
```

```
## randomForest 4.6-12
```

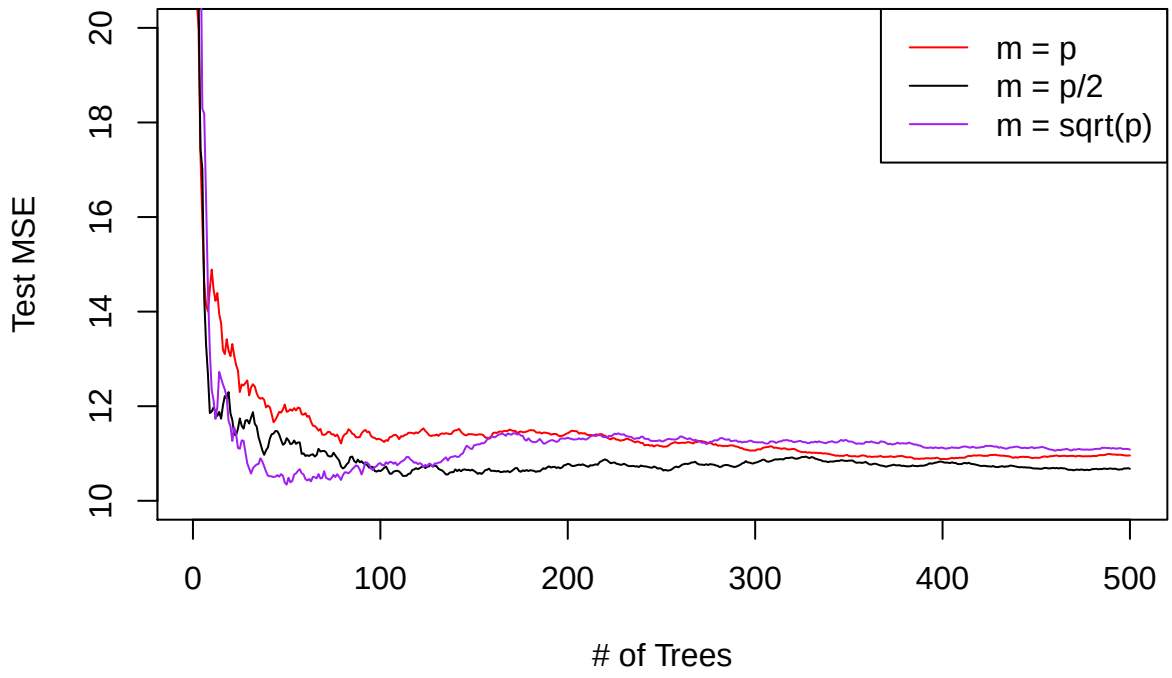
```
## Type rfNews() to see new features/changes/bug fixes.
```

```
set.seed(15)
# Split the data into 70% training and 30% testing
mytrain <- sample(1:nrow(Boston), nrow(Boston)*0.7)

rf1 <- randomForest(Boston[mytrain, -14], y = Boston$medv[mytrain], xtest = Boston[-mytrain, -14],
```

```
                              ytest = Boston$medv[-mytrain], mtry = ncol(Boston) - 1, ntree = 500)
rf2 <- randomForest(Boston[mytrain, -14], y = Boston$medv[mytrain], xtest = Boston[-mytrain, -14],
                              ytest = Boston$medv[-mytrain], mtry = (ncol(Boston) - 1) / 2, ntree = 500)
rf3 <- randomForest(Boston[mytrain, -14], y = Boston$medv[mytrain], xtest = Boston[-mytrain, -14],
                              ytest = Boston$medv[-mytrain], mtry = sqrt(ncol(Boston) - 1), ntree = 500)

plot(1:500, rf1$test$mse, col = "red", type = "l", ylim = c(10, 20),
     ylab = "Test MSE", xlab = "# of Trees")
lines(1:500, rf2$test$mse, col = "black", type = "l")
lines(1:500, rf3$test$mse, col = "purple", type = "l")
legend("topright", c("m = p", "m = p/2", "m = sqrt(p)"),
       col = c("red", "black", "purple"), cex = 1, lty = 1)
```



According to the output, test MSE decreases when number of trees increase. Test MSE is pretty high when number of trees is 1. Additionally, the test MSE of all predictors is higher than it is for both the square root of the predictors and half of the predictors at the beginning. Then, it becomes slightly lower than the test MSE for square root of the predictors as the number of trees increase.

Question 8.4 10

a)

```
library(ISLR)
Hitters=Hitters[!is.na(Hitters$Salary),,drop=F]
Hitters$Salary <- log(Hitters$Salary)
```

b)

```
mytrain=1:200
train=Hitters[mytrain,]
test=Hitters[-mytrain,]
```

c)

```
library(gbm)
```

```
## Warning: package 'gbm' was built under R version 3.3.2
```

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

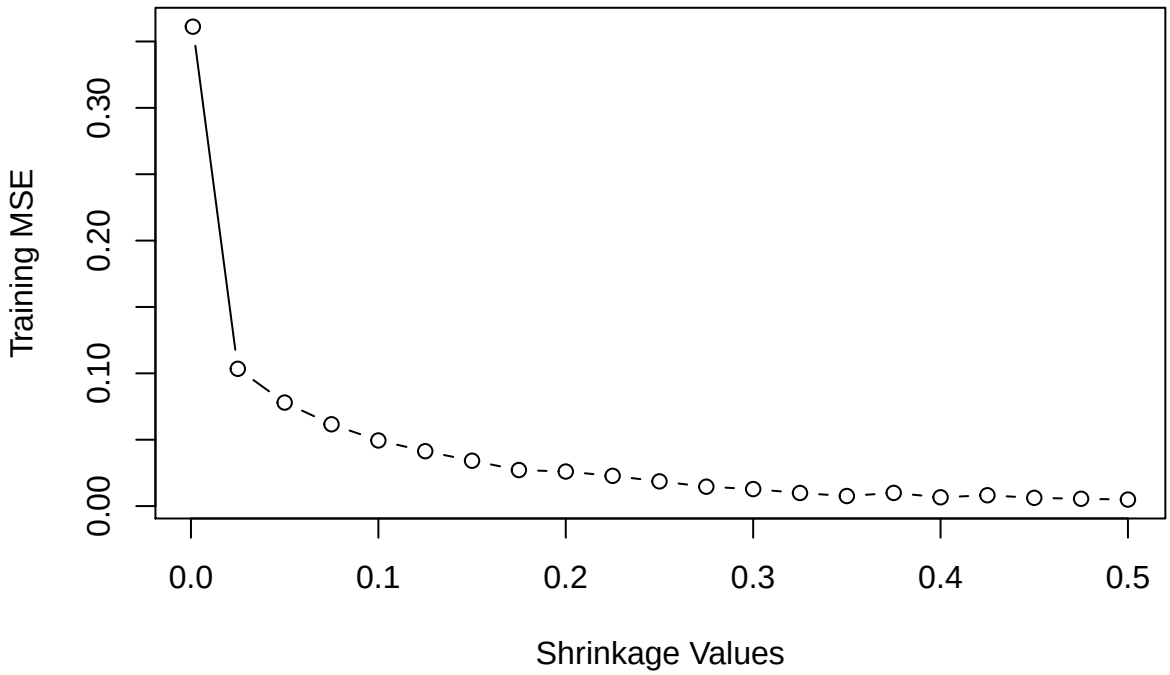
```
## Loading required package: lattice
```

```
## Loading required package: splines
```

```
## Loading required package: parallel
```

```
## Loaded gbm 2.1.3
```

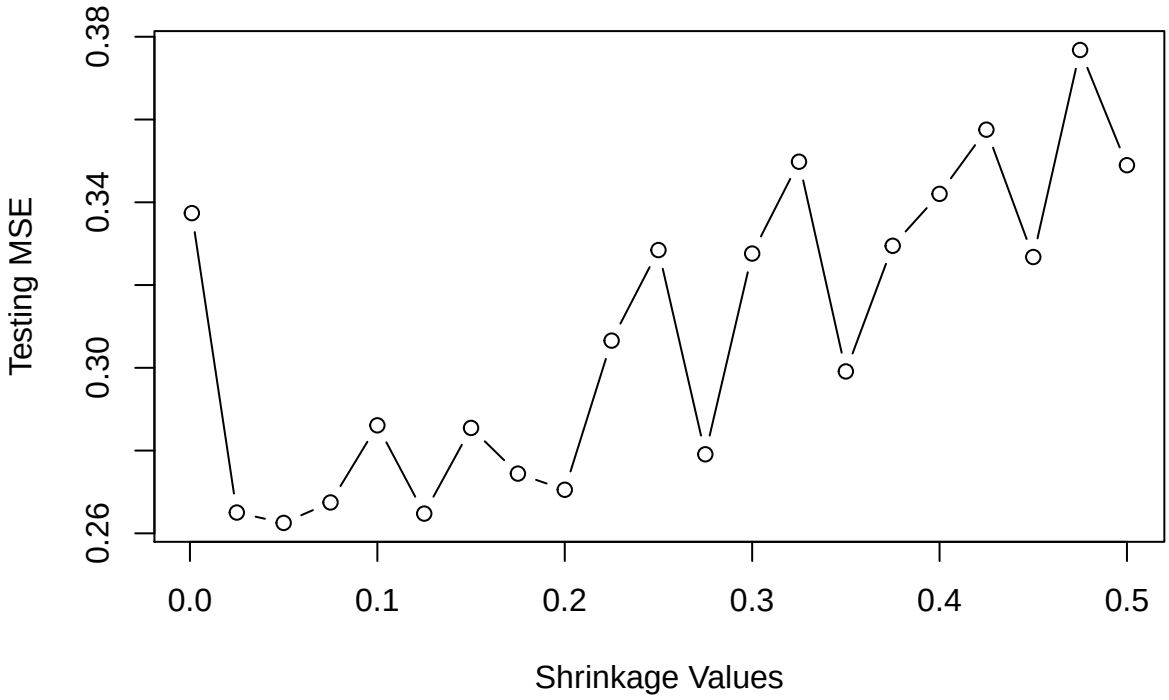
```
set.seed(15)
# use at least 10 different values of the shrinkage parameter between 0.001 and 0.5
shrinkage=c(0.001,0.025,0.05,0.075,0.1,0.125,0.15,0.175,0.2,0.225,0.25,0.275,0.3,
            0.325,0.35,0.375,0.4,0.425,0.45,0.475,0.5)
trainMSE=rep(NA,length(shrinkage))
for (i in 1:length(shrinkage)){
  h.boost=gbm(Salary~., data=train,
              shrinkage = shrinkage[i], distribution="gaussian", n.trees=1000)
  ypred=predict(h.boost,newdata=train, n.trees=1000)
  trainMSE[i]=mean((ypred-train$Salary)^2)
}
plot(shrinkage,trainMSE,type = "b", xlab = "Shrinkage Values", ylab = "Training MSE")
```



According to the output, as shinkage values increases, the training MSE decreases gradually.

d)

```
# use at least 10 different values of the shrinkage parameter between 0.001 and 0.5
shrinkage=c(0.001,0.025,0.05,0.075,0.1,0.125,0.15,0.175,0.2,0.225,0.25,0.275,0.3,
            0.325,0.35,0.375,0.4,0.425,0.45,0.475,0.5)
testMSE=rep(NA,length(shrinkage))
for (i in 1:length(shrinkage)){
  h.boost=gbm(Salary~., data=train,
              shrinkage =shrinkage[i], distribution="gaussian", n.trees=1000)
  ypred=predict(h.boost,newdata=test, n.trees=1000)
  testMSE[i]=mean((ypred-test$Salary)^2)
}
plot(shrinkage,testMSE,type = "b", xlab = "Shrinkage Values", ylab = "Testing MSE")
```




```
# find out the shrinkage value when minimum value of test MSE
# occurs
shrinkage[which.min(testMSE)]
```

```
## [1] 0.05
```

```
# find out the minimum value of test MSE
min(testMSE)
```

```
## [1] 0.2625214
```

As the outputs show, the minimum value of test MSE is 0.2625214. It occurs when shrinkage value is 0.05.

Question 8.4 12

```
# read the data
mydata=read.table("https://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data",sep=",")
# factorize the response variable
mydata$V1=as.factor(mydata$V1)

# Split into train and test. Use a 70/30 training-test split.
train <- sample(nrow(mydata), nrow(mydata) *0.7)
mytrain <- mydata[train, ]
mytest <- mydata[-train, ]

# Fit a multinomial logistic
# regression with all the parameters as predictors.
# And see the accuracy.
# This is my basic model.
library("nnet")
logit<-multinom(V1~.,data=mytrain)
```

```
## # weights:  45 (28 variable)
## initial  value 136.227924
## iter  10 value 18.559094
## iter  20 value 1.496523
## iter  30 value 0.003201
## final  value 0.000050
## converged
```

```
pred=predict(logit,newdata = mytest,type='class')
tab=table(mytest$V1,pred)
tab # See the confusion matrix
```

```
##    pred
##      1  2  3
##   1 16  1  0
##   2  0 22  1
##   3  1  1 12
```

```
print(paste('Test accuracy rate:',sum(diag(tab))/sum(tab)))
```

```
## [1] "Test accuracy rate: 0.925925925925926"
```

```
library(gbm)
# Fit a boosting with all the parameters as predictors and see the accuracy.
boosting<-gbm(V1~.,data=mytrain,distribution='multinomial',n.trees=5000)
#summary(logit)
pred=predict(boosting, mytest,n.trees = 5000,type='response')
p.pred <- apply(pred, 1, which.max)
tab=table(mytest$V1,p.pred)
tab # See the confusion matrix
```

```
##    p.pred
##      1  2  3
##   1 17  0  0
##   2  1 20  2
##   3  0  0 14
```

```
print(paste('Test accuracy rate:',sum(diag(tab))/sum(tab)))
```

```
## [1] "Test accuracy rate: 0.944444444444444"
```

```
# Fit a boosting with 'Alcohol'(V2) and 'Malic acid' (V3)
# as predictors and see the accuracy.
boosting<-gbm(V1~V2+V3,data=mytrain, distribution='multinomial',n.trees=5000)
#summary(logit)
pred=predict(boosting, mytest,n.trees = 5000,type='response')
p.pred <- apply(pred, 1, which.max)
tab=table(mytest$V1,p.pred)
tab # See the confusion matrix
```

```
##    p.pred
##      1  2  3
##   1 15  0  2
##   2  2 18  3
##   3  1  4  9
```

```
print(paste("Test accuracy rate:",sum(diag(tab))/sum(tab)))
```

```
## [1] "Test accuracy rate: 0.777777777777778"
```

```
# bagging with all
# predictors and see the accuracy.
library(randomForest)
#length(mydata)
bagging<-randomForest(V1~.,data=mydata,subset=train,
                      mytry=13, importance = TRUE)
#summary(bagging)
pred=predict(bagging, newdata =mytest)
#length(pred)
tab=table(mytest$V1,pred)
tab # See the confusion matrix
```

```
##    pred
##      1  2  3
##   1 17  0  0
##   2  1 21  1
##   3  0  0 14
```

```
print(paste('Test accuracy rate:',sum(diag(tab))/sum(tab)))
```

```
## [1] "Test accuracy rate: 0.962962962962963"
```

```
# bagging with 'Alcohol'(V2) and 'Malic acid' (V3)
# as predictors and see the accuracy.
library(randomForest)
#length(mydata)
bagging<-randomForest(V1~V2+V3,data=mydata,subset=train,
                      mytry=2, importance = TRUE)
#summary(bagging)
pred=predict(bagging, newdata =mytest)
#length(pred)
tab=table(mytest$V1,pred)
tab # See the confusion matrix
```

```
##    pred
##      1  2  3
##   1 15  1  1
##   2  3 17  3
##   3  2  4  8
```

```
print(paste(
  'Test accuracy rate:',sum(diag(tab))/sum(tab)))
```

```
## [1] "Test accuracy rate: 0.740740740740741"
```

```
# Random forest with all the parameters as predictors and see the accuracy.
library(randomForest)
#length(mydata)
rf<-randomForest(V1~.,data=mydata,subset=train,
                 mytry=ceiling(sqrt(13)), importance = TRUE)
#summary(bagging)
pred=predict(rf, newdata =mytest)
#length(pred)
tab=table(mytest$V1,pred)
tab # See the confusion matrix
```

```
##    pred
##      1  2  3
##   1 17  0  0
##   2  1 21  1
##   3  0  0 14
```

```
print(paste('Test accuracy rate:',sum(diag(tab))/sum(tab)))
```

```
## [1] "Test accuracy rate: 0.962962962962963"
```

```
# Random forest with 'Alcohol' (V2) and 'Malic acid' (V3)
# as predictors and see the accuracy.
library(randomForest)
#length(mydata)
rf<-randomForest(V1~V2+V3,data=mydata,subset=train,
                 mytry=ceiling(sqrt(2)), importance = TRUE)
#summary(bagging)
pred=predict(rf, newdata =mytest)
#length(pred)
tab=table(mytest$V1,pred)
tab # See the confusion matrix
```

```
##    pred
##      1  2  3
##   1 15  1  1
##   2  3 17  3
##   3  2  4  8
```

```
print(paste(
  'Test accuracy rate:',sum(diag(tab))/sum(tab)))
```

```
## [1] "Test accuracy rate: 0.740740740740741"
```

I have used two different parameter settings each for bagging, random forests, and boosting. The two parameter settings are all parameters and the subset of all parameters ('Alcohol' (V2) and 'Malic acid' (V3)). I have also fitted a multinomial logistic regression with all parameters. The models using all parameters from bagging, random forests, and boosting all have a higher accuracy rate than the model using all parameters from multinomial logistic regression.

As the outputs shows, both random forest with all predictors and bagging with all predictors achieve the highest accuracy rate (0.96296).