CS6140 Assignment 4

Chengbo Gu

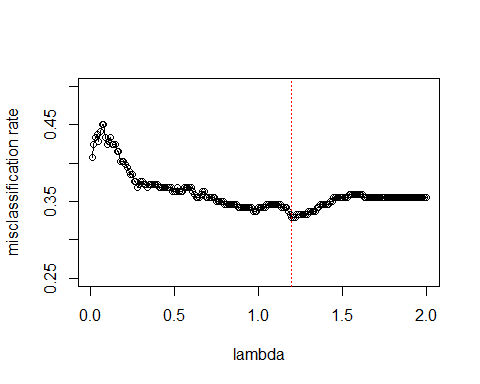
### 1. JWHT Chapter 7, Problem 1

### 2. JWHT Chapter 7, Problem 3

set.seed(123)  
Rawdata <- read.table("SouthAfricanHeartDisease.txt", sep=",",   
 stringsAsFactors = FALSE, header = TRUE)  
#Rawdata[,11] <- as.factor(Rawdata[,11])  
Mydata <- Rawdata[,-6]  
Mydata <- Mydata[,-1]  
  
# for naive bayes  
predictors <- Mydata[,1:8]  
means <- apply(predictors, 2, mean)  
sds <- apply(predictors, 2, sd)  
  
predictors <- t(apply(predictors, 1, function(x) (x-means)/sds))  
response <- Mydata[,9]  
  
train <- sample(x=1:nrow(Mydata), size=nrow(Mydata)/2)  
trainResponse <- response[train]  
testResponse <- response[-train]  
  
trainPredictors <- predictors[train,]  
testPredictors <- predictors[-train,]  
  
# for others  
Mydata <- Rawdata[,-1]  
Mydata[,5][Mydata[,5]=="Present"] <- 1  
Mydata[,5][Mydata[,5]=="Absent"] <- 0  
Mydata[,5] <- as.integer(Mydata[,5])  
trainSet <- Mydata[train,]  
testSet <- Mydata[-train,]

### problem 3. Naive Bayes classifier

# leave-one-out cross validation  
# only support response contains 0 and 1  
naive.bayes.cv <- function(lambda.seq, predictors, response){  
 miscl <- rep(NA, length(lambda.seq))  
 missd <- rep(NA, length(lambda.seq))  
 for (k in 1:length(lambda.seq)) {  
 lambda <- lambda.seq[k]  
 n <- length(response)  
 naiveBayes.raw <- rep(NA, n)  
 for (i in 1:n) {  
 data.tr.predictors <- predictors[-i,]  
 data.tr.response <- response[-i]  
 data.test.predictors <- predictors[i,]  
 data.test.response <- response[i]  
 density <- dnorm(abs(data.test.predictors - data.tr.predictors)/lambda, mean = 0, sd=1)  
 #print(density)  
 ones <- which(data.tr.response %in% c(1))  
 response.ones <- which(response %in% c(1))  
   
 len.ones <- length(response.ones)  
 len.zeros <- n - len.ones  
 prior.ones <- len.ones/n  
 prior.zeros <- len.zeros/n  
   
 density.ones <- density[ones,]  
 density.zeros <- density[-ones,]  
   
 density.one <- prod(apply(density.ones, 2, sum)/len.ones)  
 density.zero <- prod(apply(density.zeros, 2, sum)/len.zeros)  
   
 prob.one <- prior.ones \* density.one / (prior.ones \* density.one + prior.zeros \* density.zero)  
 naiveBayes.raw[i] <- prob.one   
 }  
 cl <- sapply(naiveBayes.raw, function(x) x>0.5)  
 miscl[k] <- 1-mean(cl == response)  
 missd[k] <- sd(cl == response)  
 }  
 plot(lambda.seq, miscl, ylim=c(0.25, 0.5), ylab="misclassification rate", xlab="lambda")  
 lines(lambda.seq, miscl)  
 abline(v=lambda.seq[which.min(miscl)], col="red", lty=3)  
 cat("Best lambda is :", lambda.seq[which.min(miscl)],"\n")  
}  
  
lambda.seq <- seq(0.01, 2, 0.01)  
naive.bayes.cv(lambda.seq, trainPredictors, trainResponse)



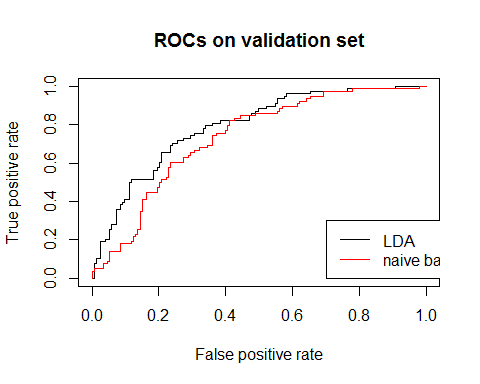
## Best lambda is : 1.2

# using train dataset  
naive.bayes.cl <- function(lambda, predictors, response, trainPredictors) {  
 n <- length(response)  
 naiveBayes.raw <- rep(NA, n)  
 for (i in 1:n) {  
  
 data.test.predictors <- predictors[i,]  
 density <- dnorm(abs(data.test.predictors - trainPredictors)/lambda, mean = 0, sd=1)  
  
 ones <- which(trainResponse %in% c(1))  
 response.ones <- which(trainResponse %in% c(1))  
   
 len.ones <- length(response.ones)  
 len.zeros <- length(trainResponse) - len.ones  
 prior.ones <- len.ones/length(trainResponse)  
 prior.zeros <- len.zeros/length(trainResponse)  
   
 density.ones <- density[ones,]  
 density.zeros <- density[-ones,]  
  
 density.one <- prod(apply(density.ones, 2, sum)/len.ones)  
 density.zero <- prod(apply(density.zeros, 2, sum)/len.zeros)  
   
 prob.one <- prior.ones \* density.one / (prior.ones \* density.one + prior.zeros \* density.zero)  
 naiveBayes.raw[i] <- prob.one   
 }  
 return(naiveBayes.raw)  
}

naiveBayes.train.scores <- naive.bayes.cl(1.2, trainPredictors, trainResponse, trainPredictors)  
naiveBayes.test.scores <- naive.bayes.cl(1.2, testPredictors, testResponse, trainPredictors)

lda.fit <- lda(x=as.matrix(trainPredictors), grouping= as.factor(trainResponse), cv=TRUE)

# lda  
# ROC on the validation set  
scores <- predict(lda.fit, newdata= testPredictors)$posterior[,2]  
pred <- prediction( scores, labels= testResponse )  
perf <- performance(pred, "tpr", "fpr")  
plot(perf, col= 1, main="ROCs on validation set")  
# print out the area under the curve  
lda.test <- unlist(attributes(performance(pred, "auc"))$y.values)  
  
# naive bayes  
pred <- prediction(naiveBayes.test.scores, labels=testResponse)  
perf <- performance(pred, "tpr", "fpr")  
  
# plot the ROC curve  
plot(perf, col= 2, add=T)  
# print out the area under the curve  
naive.test <- unlist(attributes(performance(pred, "auc"))$y.values)  
legend(0.7,0.3, c("LDA", "naive bayes"), col=c(1:2), lty=1)



### problem 4. Tree-based Methods

##### (a) JWHT Chapter 8, Problem 9, but using the "South African Heart Disease" dataset

###### (a-b)

tree.cl <- tree(as.factor(chd) ~ ., data=trainSet)  
summary(tree.cl)

##   
## Classification tree:  
## tree(formula = as.factor(chd) ~ ., data = trainSet)  
## Number of terminal nodes: 26   
## Residual mean deviance: 0.5249 = 107.6 / 205   
## Misclassification error rate: 0.1169 = 27 / 231

Training error rate is 11.69%. There are 26 terminal nodes.

###### (c)

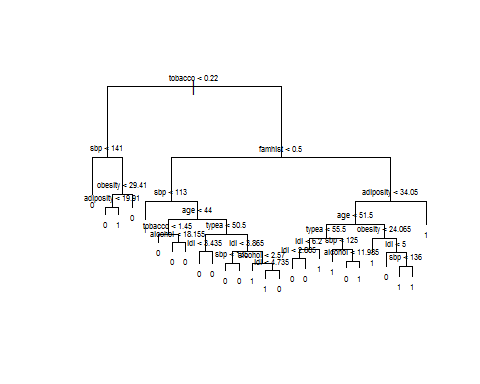
tree.cl

## node), split, n, deviance, yval, (yprob)  
## \* denotes terminal node  
##   
## 1) root 231 300.500 0 ( 0.64502 0.35498 )   
## 2) tobacco < 0.22 67 40.400 0 ( 0.91045 0.08955 )   
## 4) sbp < 141 49 0.000 0 ( 1.00000 0.00000 ) \*  
## 5) sbp > 141 18 22.910 0 ( 0.66667 0.33333 )   
## 10) obesity < 29.41 12 16.640 0 ( 0.50000 0.50000 )   
## 20) adiposity < 19.91 5 5.004 0 ( 0.80000 0.20000 ) \*  
## 21) adiposity > 19.91 7 8.376 1 ( 0.28571 0.71429 ) \*  
## 11) obesity > 29.41 6 0.000 0 ( 1.00000 0.00000 ) \*  
## 3) tobacco > 0.22 164 226.500 0 ( 0.53659 0.46341 )   
## 6) famhist < 0.5 83 99.830 0 ( 0.71084 0.28916 )   
## 12) sbp < 113 6 5.407 1 ( 0.16667 0.83333 ) \*  
## 13) sbp > 113 77 86.050 0 ( 0.75325 0.24675 )   
## 26) age < 44 31 19.710 0 ( 0.90323 0.09677 )   
## 52) tobacco < 1.45 14 0.000 0 ( 1.00000 0.00000 ) \*  
## 53) tobacco > 1.45 17 15.840 0 ( 0.82353 0.17647 )   
## 106) alcohol < 18.155 9 11.460 0 ( 0.66667 0.33333 ) \*  
## 107) alcohol > 18.155 8 0.000 0 ( 1.00000 0.00000 ) \*  
## 27) age > 44 46 59.440 0 ( 0.65217 0.34783 )   
## 54) typea < 50.5 18 12.560 0 ( 0.88889 0.11111 )   
## 108) ldl < 3.435 5 6.730 0 ( 0.60000 0.40000 ) \*  
## 109) ldl > 3.435 13 0.000 0 ( 1.00000 0.00000 ) \*  
## 55) typea > 50.5 28 38.820 1 ( 0.50000 0.50000 )   
## 110) ldl < 3.865 10 10.010 0 ( 0.80000 0.20000 )   
## 220) sbp < 135 5 0.000 0 ( 1.00000 0.00000 ) \*  
## 221) sbp > 135 5 6.730 0 ( 0.60000 0.40000 ) \*  
## 111) ldl > 3.865 18 22.910 1 ( 0.33333 0.66667 )   
## 222) alcohol < 2.57 8 6.028 1 ( 0.12500 0.87500 ) \*  
## 223) alcohol > 2.57 10 13.860 0 ( 0.50000 0.50000 )   
## 446) ldl < 4.735 5 5.004 1 ( 0.20000 0.80000 ) \*  
## 447) ldl > 4.735 5 5.004 0 ( 0.80000 0.20000 ) \*  
## 7) famhist > 0.5 81 105.700 1 ( 0.35802 0.64198 )   
## 14) adiposity < 34.05 70 94.970 1 ( 0.41429 0.58571 )   
## 28) age < 51.5 38 52.260 0 ( 0.55263 0.44737 )   
## 56) typea < 55.5 21 25.130 0 ( 0.71429 0.28571 )   
## 112) ldl < 6.2 14 11.480 0 ( 0.85714 0.14286 )   
## 224) ldl < 2.805 5 6.730 0 ( 0.60000 0.40000 ) \*  
## 225) ldl > 2.805 9 0.000 0 ( 1.00000 0.00000 ) \*  
## 113) ldl > 6.2 7 9.561 1 ( 0.42857 0.57143 ) \*  
## 57) typea > 55.5 17 22.070 1 ( 0.35294 0.64706 )   
## 114) sbp < 125 5 0.000 1 ( 0.00000 1.00000 ) \*  
## 115) sbp > 125 12 16.640 0 ( 0.50000 0.50000 )   
## 230) alcohol < 11.985 5 5.004 0 ( 0.80000 0.20000 ) \*  
## 231) alcohol > 11.985 7 8.376 1 ( 0.28571 0.71429 ) \*  
## 29) age > 51.5 32 35.990 1 ( 0.25000 0.75000 )   
## 58) obesity < 24.065 9 0.000 1 ( 0.00000 1.00000 ) \*  
## 59) obesity > 24.065 23 29.720 1 ( 0.34783 0.65217 )   
## 118) ldl < 5 9 11.460 0 ( 0.66667 0.33333 ) \*  
## 119) ldl > 5 14 11.480 1 ( 0.14286 0.85714 )   
## 238) sbp < 136 5 6.730 1 ( 0.40000 0.60000 ) \*  
## 239) sbp > 136 9 0.000 1 ( 0.00000 1.00000 ) \*  
## 15) adiposity > 34.05 11 0.000 1 ( 0.00000 1.00000 ) \*

1. adiposity > 34.05 11 0.000 1 ( 0.00000 1.00000 ) To go to the 15th terminal node, adiposity should be greater than 34.05, there is 1 observation in this branch, the deviance is 0, the prediction is chd = 1 with probability 1.

###### (d)

plot(tree.cl)  
text(tree.cl ,pretty =0, cex=0.5)



Initial tree built for training set without pruning.

###### (e)

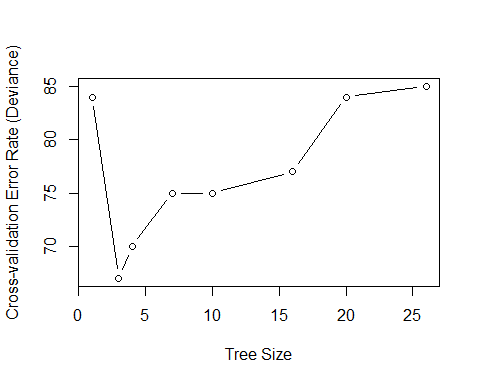
tree.pred <- predict (tree.cl , testSet, type ="class")  
table(tree.pred, testSet[,10])

##   
## tree.pred 0 1  
## 0 117 40  
## 1 36 38

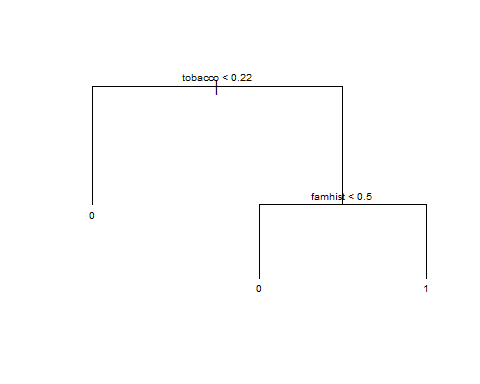
The misclassification rate for testing set is 32.90%.

###### (f-i)

set.seed(258)  
cv.tree.cl =cv.tree(tree.cl , FUN=prune.misclass)  
plot(cv.tree.cl$size ,cv.tree.cl$dev ,type="b", ylab="Cross-validation Error Rate (Deviance)", xlab="Tree Size")



prune.tree =prune.misclass (tree.cl ,best =3)  
plot(prune.tree)  
text(prune.tree ,pretty =0, cex=0.6)



###### (j)

tree.prune.train.pred <- predict(prune.tree, trainSet, type="class")  
mean(tree.prune.train.pred != trainSet[,10])

## [1] 0.2554113

prune.train.acc <- 1-mean(tree.prune.train.pred != trainSet[,10])

The misclassification rate on training set of pruned tree is 25.54% which is higher than the one (11.69%) of unpruned tree.

###### (k)

tree.prune.test.pred <- predict (prune.tree , testSet, type="class")  
mean(tree.prune.test.pred != testSet[,10])

## [1] 0.3376623

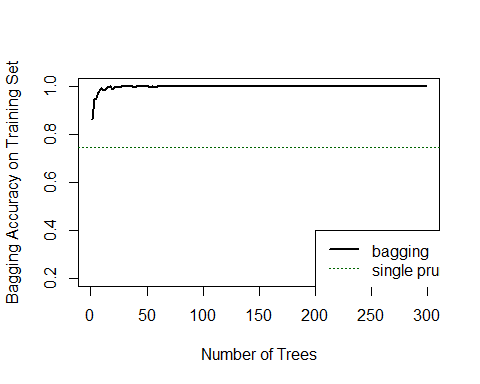
prune.test.acc <- 1-mean(tree.prune.test.pred != testSet[,10])

The misclassification rate on testing set of pruned tree is 33.77% which is lower than the one (38.53%) of unpruned tree.

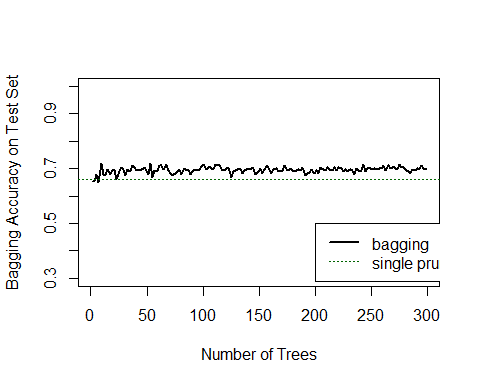
##### (b) Bagging

B.seq <- seq(1, 300, 2)  
acc.train.seq.bag <- rep(NA, length(B.seq))  
acc.test.seq.bag <- rep(NA, length(B.seq))  
set.seed (123)  
for (i in 1:length(B.seq)) {  
 bagging <- randomForest(as.factor(chd)~.,data=trainSet , mtry=9, importance =TRUE, ntree = B.seq[i])  
 #acc.train.seq.OOB[i] <- (bagging$confusion[1,1] + bagging$confusion[2,2])/dim(trainSet)[1]  
 pred.bagging <- predict(bagging, trainSet)  
 acc.train.seq.bag[i] <- mean(pred.bagging == trainSet[,10])  
   
 pred.bagging <- predict(bagging ,newdata =testSet)  
 acc.test.seq.bag[i] <- mean(pred.bagging == testSet[,10])  
}

plot(B.seq, acc.train.seq.bag, ylim=c(0.2,1), type="l", xlab="Number of Trees", ylab="Bagging Accuracy on Training Set", lwd=2)  
#lines(B.seq, acc.train.seq, ylim=c(0.2,1))  
abline(h=prune.train.acc, col="darkgreen", lty=3, lwd=1)  
legend(200,0.4, c("bagging", "single pruned tree"),  
 lty=c(1,3), lwd=c(2,1),col=c("black", "darkgreen"))



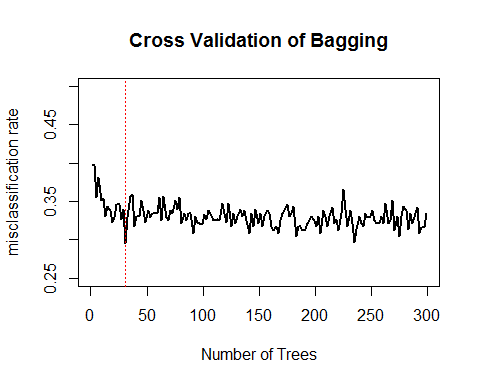
plot(B.seq, acc.test.seq.bag, ylim=c(0.3,1), type="l", xlab="Number of Trees", ylab="Bagging Accuracy on Test Set", lwd=2)  
#lines(B.seq, acc.test.seq, ylim=c(0.6,0.8))  
abline(h=prune.test.acc, col="darkgreen", lty=3, lwd=1)  
legend(200,0.5, c("bagging", "single pruned tree"),  
 lty=c(1,3), lwd=c(2,1),col=c("black", "darkgreen"))



#bagging  
#(bagging$confusion[1,1] + bagging$confusion[2,2])/dim(trainSet)[1]

# cross-validation to choose number of Trees on Bagging  
set.seed(321)  
data <- trainSet  
n <- dim(trainSet)[1]  
index <- 1:n  
K <- 10  
flds <- createFolds(index, k=K)  
miscl.cv <- rep(NA, length(B.seq))  
  
for (j in 1:length(B.seq)){  
 miscl.cv.raw <- rep(NA, K)  
 for (i in 1:K){  
 testID <- flds[[i]]  
 data.tr <- data[-testID,]  
 data.test <- data[testID,]  
 tree.cv <- randomForest(as.factor(chd)~.,data=data.tr , mtry=9, importance =TRUE, ntree = B.seq[j])  
 tree.cv.pred <- predict(tree.cv, newdata = data.test, type = "c")  
 #fnr.cv.raw[i] <- sum(tree.cv.pred == "FALSE" & data.test$IsB == "TRUE")/sum(data.test$IsB == "TRUE")  
 miscl.cv.raw[i] <- mean(tree.cv.pred != data.test[,10])  
 }  
 miscl.cv[j] <- mean(miscl.cv.raw)  
}

plot(B.seq, miscl.cv, type="l", lwd=2.5, ylim=c(0.25,0.5),   
 xlab= "Number of Trees", ylab="misclassification rate", main="Cross Validation of Bagging")  
abline(v=B.seq[which.min(miscl.cv)], lty=3, col="red")



cat("Best B for Bagging is :", B.seq[which.min(miscl.cv)], "\n")

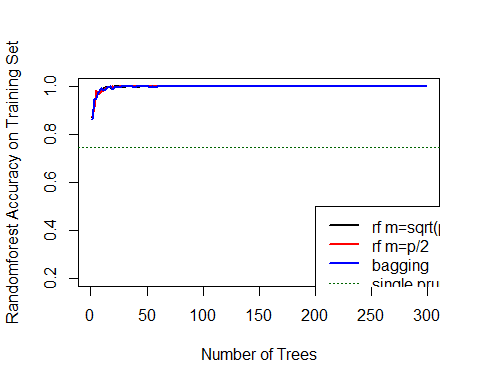
## Best B for Bagging is : 31

bagging.tree <- randomForest(as.factor(chd)~.,data=trainSet , mtry=9, importance =TRUE, ntree = 31)

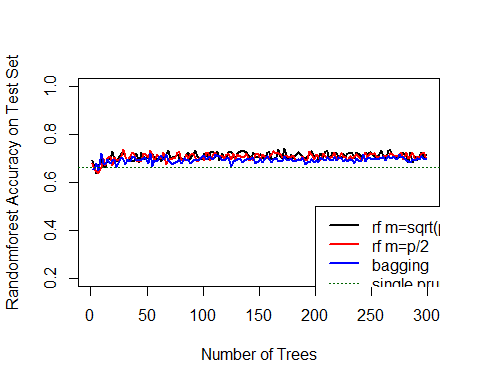
##### (c) RandomForest

acc.train.seq.sqrt <- rep(NA, length(B.seq))  
acc.test.seq.sqrt <- rep(NA, length(B.seq))  
acc.train.seq.by2 <- rep(NA, length(B.seq))  
acc.test.seq.by2 <- rep(NA, length(B.seq))  
set.seed (123)  
for (i in 1:length(B.seq)) {  
   
 rf.tree.sqrt <- randomForest(as.factor(chd)~., data=trainSet , mtry=sqrt(9), importance =TRUE, ntree = B.seq[i])  
 rf.tree.by2 <- randomForest(as.factor(chd)~., data=trainSet , mtry=9/2, importance =TRUE, ntree = B.seq[i])  
   
 #acc.train.seq.OOB.sqrt[i] <- (rf.tree.sqrt$confusion[1,1] + rf.tree.sqrt$confusion[2,2])/dim(trainSet)[1]  
 pred.rf.sqrt <- predict (rf.tree.sqrt, newdata = trainSet)  
 acc.train.seq.sqrt[i] <- mean(pred.rf.sqrt ==trainSet[,10])  
   
 pred.rf.sqrt <- predict (rf.tree.sqrt ,newdata = testSet)  
 acc.test.seq.sqrt[i] <- mean(pred.rf.sqrt == testSet[,10])  
  
 #acc.train.seq.by2[i] <- (rf.tree.by2$confusion[1,1] + rf.tree.by2$confusion[2,2])/dim(trainSet)[1]  
 pred.rf.by2 <- predict(rf.tree.by2, trainSet)  
 acc.train.seq.by2[i] <- mean(pred.rf.by2 == trainSet[,10])   
  
   
 pred.rf.by2 <- predict (rf.tree.by2 ,newdata =testSet)  
 acc.test.seq.by2[i] <- mean(pred.rf.by2 == testSet[,10])   
}

# black: rf with sqrt(p) features  
# red: rf with p/2 features  
# blue: bagging  
# purple: single pruned tree  
plot(B.seq, acc.train.seq.sqrt, ylim=c(0.2,1), type="l", xlab="Number of Trees", ylab="Randomforest Accuracy on Training Set", lwd=2)  
lines(B.seq, acc.train.seq.by2, col="red", lwd=2)  
lines(B.seq, acc.train.seq.bag, col="blue", lwd=2)  
abline(h=prune.train.acc, col="darkgreen", lty=3, lwd=1)  
legend(200,0.5, c("rf m=sqrt(p)", "rf m=p/2", "bagging", "single pruned tree"),  
 lty=c(1,1,1,3), lwd=c(2,2,2,1),col=c("black","red", "blue", "darkgreen"))

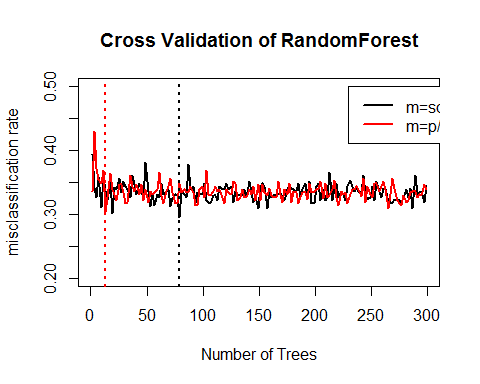


plot(B.seq, acc.test.seq.sqrt, ylim=c(0.2,1), type="l", xlab="Number of Trees", ylab="Randomforest Accuracy on Test Set", lwd=2)  
lines(B.seq, acc.test.seq.by2, col="red", lwd=2)  
lines(B.seq, acc.test.seq.bag, col="blue", lwd=2)  
abline(h=prune.test.acc, col="darkgreen", lty=3, lwd=1)  
legend(200,0.5, c("rf m=sqrt(p)", "rf m=p/2", "bagging", "single pruned tree"),  
 lty=c(1,1,1,3), lwd=c(2,2,2,1),col=c("black","red", "blue", "darkgreen"))



# cross-validation to choose number of Trees on RandomForest  
set.seed(321)  
data <- trainSet  
n <- dim(trainSet)[1]  
index <- 1:n  
K <- 10  
flds <- createFolds(index, k=K)  
miscl.sqrt.cv <- rep(NA, length(B.seq))  
miscl.by2.cv <- rep(NA, length(B.seq))  
  
for (j in 1:length(B.seq)){  
 miscl.sqrt.cv.raw <- rep(NA, K)  
 miscl.by2.cv.raw <- rep(NA, K)  
 for (i in 1:K){  
 testID <- flds[[i]]  
 data.tr <- data[-testID,]  
 data.test <- data[testID,]  
 tree.cv <- randomForest(as.factor(chd)~.,data=data.tr , mtry=sqrt(9), importance =FALSE, ntree = B.seq[j])  
 tree.cv.pred <- predict(tree.cv, newdata = data.test, type = "c")  
 miscl.sqrt.cv.raw[i] <- mean(tree.cv.pred != data.test[,10])  
   
 tree.cv <- randomForest(as.factor(chd)~.,data=data.tr , mtry=9/2, importance =FALSE, ntree = B.seq[j])  
 tree.cv.pred <- predict(tree.cv, newdata = data.test, type = "c")  
 miscl.by2.cv.raw[i] <- mean(tree.cv.pred != data.test[,10])  
 }  
 miscl.sqrt.cv[j] <- mean(miscl.sqrt.cv.raw)  
 miscl.by2.cv[j] <- mean(miscl.by2.cv.raw)  
}

plot(B.seq, miscl.sqrt.cv, type="l", lwd=2.5, ylim=c(0.2,0.5),   
 xlab= "Number of Trees", ylab="misclassification rate", main="Cross Validation of RandomForest")  
lines(B.seq, miscl.by2.cv, lwd=2.5, col="red")  
abline(v=B.seq[which.min(miscl.sqrt.cv)], lty=3, col="black", lwd=2)  
abline(v=B.seq[which.min(miscl.by2.cv)], lty=3, col="red", lwd=2)  
legend(230,0.5, c("m=sqrt(p)", "m=p/2"),  
 lty=c(1,1), lwd=c(2,2),col=c("black", "red"))



cat("Best B for RandomForest with m=sqrt(p) is :", B.seq[which.min(miscl.sqrt.cv)], "\n")

## Best B for RandomForest with m=sqrt(p) is : 79

cat("Best B for RandomForest with m=p/2 is :", B.seq[which.min(miscl.by2.cv)], "\n")

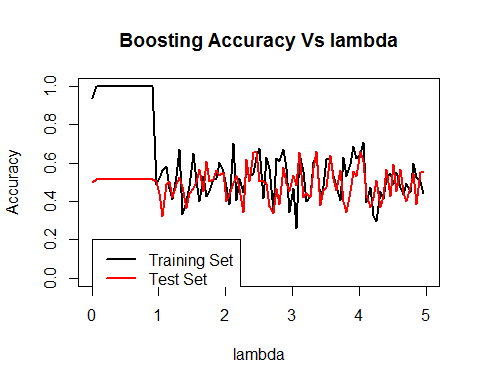
## Best B for RandomForest with m=p/2 is : 13

rf.sqrt.tree <- randomForest(as.factor(chd)~.,data=trainSet , mtry=sqrt(9), importance =TRUE, ntree = 79)  
rf.by2.tree <- randomForest(as.factor(chd)~.,data=trainSet , mtry=9/2, importance =TRUE, ntree = 13)

##### (d) Boosting

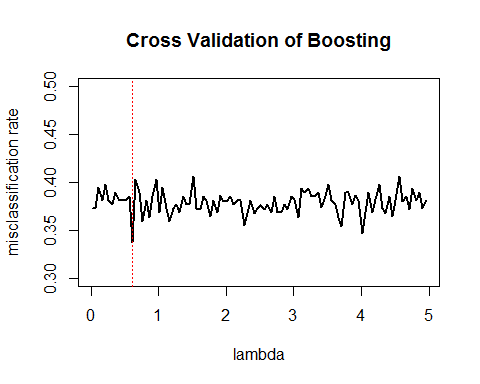
set.seed(123)  
grid <- seq(0.01, 5, 0.05)  
acc.train.seq.boost <- rep(NA, length(grid))  
acc.test.seq.boost <- rep(NA, length(grid))  
for (i in 1:length(grid)) {  
 boost.tree <- gbm(chd~., data=trainSet, distribution="bernoulli", n.trees =1000, interaction.depth=4, shrinkage=grid[i])  
 boost.tree.train.pred <- predict(boost.tree, data=trainSet, n.trees=1000, type="response") > 0.5  
 acc.train.seq.boost[i] <- mean(boost.tree.train.pred == trainSet$chd)  
  
 boost.tree.test.pred <- predict(boost.tree, data=testSet, n.trees=1000, type="response") > 0.5  
 acc.test.seq.boost[i] <- mean(boost.tree.test.pred == testSet$chd)  
}

# black: train  
# red: test  
plot(grid, acc.train.seq.boost, type="l", xlim=c(0,5), ylim=c(0,1), xlab="lambda", ylab="Accuracy", lwd=2, main="Boosting Accuracy Vs lambda")  
lines(grid, acc.test.seq.boost, type="l", col="red", lwd=2)  
  
legend(0,0.2, c("Training Set", "Test Set"),  
 lty=c(1,1), lwd=c(2,2),col=c("black", "red"))



# cross-validation to choose lambda on Boosting  
set.seed(321)  
data <- trainSet  
n <- dim(trainSet)[1]  
index <- 1:n  
K <- 10  
flds <- createFolds(index, k=K)  
miscl.cv <- rep(NA, length(grid))  
  
for (j in 1:length(grid)){  
 miscl.cv.raw <- rep(NA, K)  
 for (i in 1:K){  
 testID <- flds[[i]]  
 data.tr <- data[-testID,]  
 data.test <- data[testID,]  
 tree.cv <- gbm(chd~., data=data.tr, distribution="bernoulli", n.trees =1000, interaction.depth=4, shrinkage=grid[i])  
 tree.cv.pred <- predict(tree.cv, newdata = data.test, type = "response", n.trees=1000) > 0.5  
 miscl.cv.raw[i] <- mean(tree.cv.pred != data.test[,10])  
 }  
 miscl.cv[j] <- mean(miscl.cv.raw)  
}

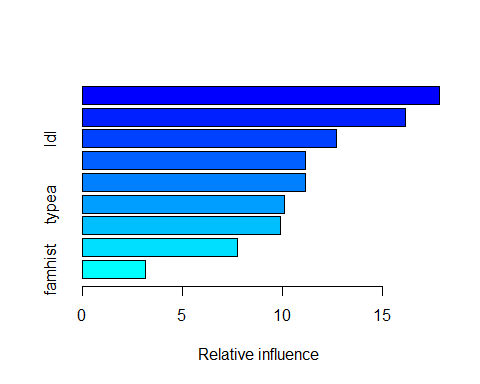
plot(grid, miscl.cv, type="l", lwd=2, ylim=c(0.3,0.5), xlab="lambda", ylab="misclassification rate",  
 main="Cross Validation of Boosting")  
abline(v=grid[which.min(miscl.cv)], col="red", lty=3)



cat("Best lambda for Boosting with B=1000 is :", grid[which.min(miscl.cv)], "\n")

## Best lambda for Boosting with B=1000 is : 0.61

boosting.tree <- gbm(chd~., data=trainSet, distribution="bernoulli", n.trees=1000, interaction.depth=4, shrinkage=0.61)  
summary(boosting.tree)



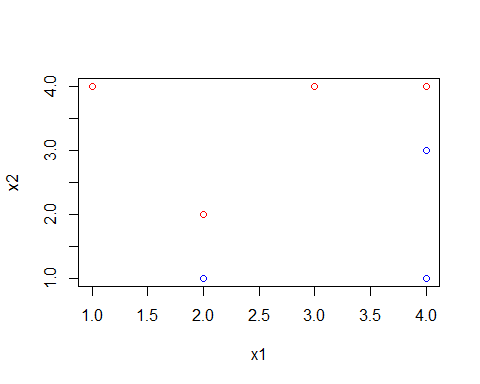
## var rel.inf  
## tobacco tobacco 17.864434  
## age age 16.165435  
## ldl ldl 12.700520  
## alcohol alcohol 11.161555  
## adiposity adiposity 11.152392  
## typea typea 10.107254  
## sbp sbp 9.931618  
## obesity obesity 7.762373  
## famhist famhist 3.154419

#boosting.pred <- predict(boosting.tree, testSet, type='response', n.trees=1000)  
#pred <- prediction(boosting.pred, labels= testSet$chd )  
#mean((boosting.pred > 0.5) == testSet$chd)

### 5. JWHT Chapter 9, Problem 3.

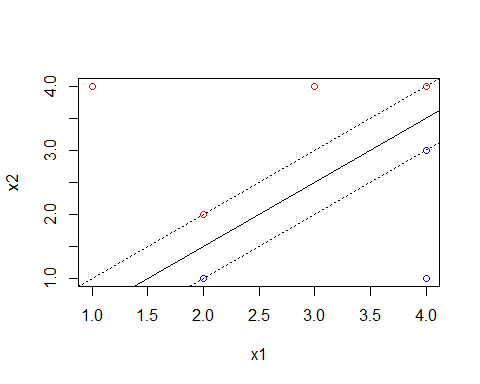
##### (a)

x1 <- c(3,2,4,1,2,4,4)  
x2 <- c(4,2,4,4,1,3,1)  
y <- c("red", "red","red","red","blue","blue","blue")  
plot(x1,x2, col=y)



##### (b)

plot(x1,x2, col=y)  
abline(a=-0.5, b=1)  
abline(a=0, b=1, lty=3)  
abline(a=-1, b=1, lty=3)



The hyperplane here is :

##### (c)

The classification rule is:

Red if

Blue if

##### (d)

The margin here is the perpendicular distance from one dotted line to the solid line.

And the value of the margin is .

##### (e)

The support vectors are:

(2,1),(2,2),(4,3),(4,4)

##### (f)

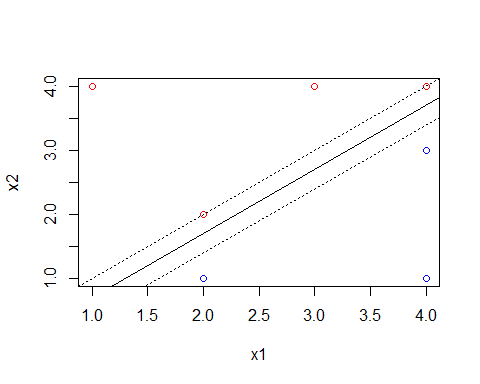
For SVM, the points that would affect the hyperplane are the support vectors.

Clearly the seventh point (4,1) is not a support vector and far from the hyperplane.

Thus, a slight movement of the seventh observation would not affect the maximal margin hyperplane.

##### (g)

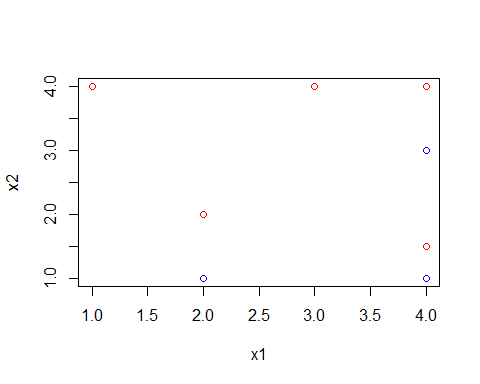
plot(x1,x2, col=y)  
abline(a=-0.3, b=1)  
abline(a=0, b=1, lty=3)  
abline(a=-0.6, b=1, lty=3)



There are infinite number of hyperplanes that are not the optimal ones. Here is one with .

##### (h)

x1 <- c(3,2,4,1,2,4,4,4)  
x2 <- c(4,2,4,4,1,3,1,1.5)  
y <- c("red", "red","red","red","blue","blue","blue", "red")  
plot(x1,x2, col=y)



The two classes won't be linearly separable with an additional red point (4, 1.5).

### 6. In this problem, we will investigate the use of support vector machines and neural networks.

##### (a) Train support vector machine the "South African Heart Disease" dataset, and evaluate its performance on the validation set.

svmdata <- Mydata  
svmdata$chd[svmdata$chd == 0] <- -1  
svm.trainSet <- svmdata[train,]  
svm.testSet <- svmdata[-train,]

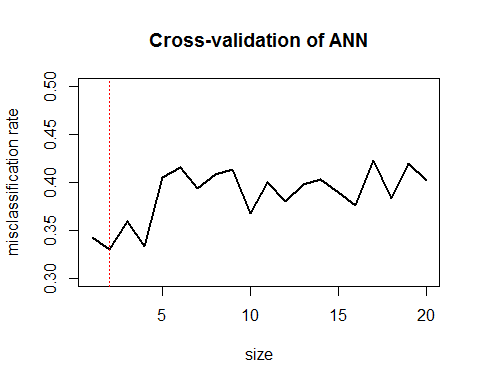
set.seed(123)  
tune.out <- tune(svm , as.factor(chd)~., data=svm.trainSet, kernel ="radial",   
 ranges =list(cost=c(0.01, 0.1, 1, 10 ,100 ,1000), gamma=c(0.005, 0.01, 0.05, 0.1, 0.5, 1,2,3,4)))  
svm.best <- tune.out$best.model  
summary(tune.out)

##   
## Parameter tuning of 'svm':  
##   
## - sampling method: 10-fold cross validation   
##   
## - best parameters:  
## cost gamma  
## 10 0.01  
##   
## - best performance: 0.2731884   
##   
## - Detailed performance results:  
## cost gamma error dispersion  
## 1 1e-02 0.005 0.3554348 0.08544121  
## 2 1e-01 0.005 0.3554348 0.08544121  
## 3 1e+00 0.005 0.3382246 0.09257879  
## 4 1e+01 0.005 0.2903986 0.09032997  
## 5 1e+02 0.005 0.2731884 0.09710325  
## 6 1e+03 0.005 0.3594203 0.07723428  
## 7 1e-02 0.010 0.3554348 0.08544121  
## 8 1e-01 0.010 0.3554348 0.08544121  
## 9 1e+00 0.010 0.3295290 0.08877613  
## 10 1e+01 0.010 0.2731884 0.09924274  
## 11 1e+02 0.010 0.3159420 0.08663386  
## 12 1e+03 0.010 0.3983696 0.08680731  
## 13 1e-02 0.050 0.3554348 0.08544121  
## 14 1e-01 0.050 0.3554348 0.08544121  
## 15 1e+00 0.050 0.2817029 0.09521647  
## 16 1e+01 0.050 0.3161232 0.08975165  
## 17 1e+02 0.050 0.3815217 0.09509748  
## 18 1e+03 0.050 0.3894928 0.07575555  
## 19 1e-02 0.100 0.3554348 0.08544121  
## 20 1e-01 0.100 0.3554348 0.08544121  
## 21 1e+00 0.100 0.2947464 0.08979552  
## 22 1e+01 0.100 0.3813406 0.08540471  
## 23 1e+02 0.100 0.3855072 0.07332818  
## 24 1e+03 0.100 0.3768116 0.05158009  
## 25 1e-02 0.500 0.3554348 0.08544121  
## 26 1e-01 0.500 0.3554348 0.08544121  
## 27 1e+00 0.500 0.3596014 0.08807839  
## 28 1e+01 0.500 0.3981884 0.06655565  
## 29 1e+02 0.500 0.3981884 0.06655565  
## 30 1e+03 0.500 0.3981884 0.06655565  
## 31 1e-02 1.000 0.3554348 0.08544121  
## 32 1e-01 1.000 0.3554348 0.08544121  
## 33 1e+00 1.000 0.3597826 0.09314484  
## 34 1e+01 1.000 0.4202899 0.08855219  
## 35 1e+02 1.000 0.4202899 0.08855219  
## 36 1e+03 1.000 0.4202899 0.08855219  
## 37 1e-02 2.000 0.3554348 0.08544121  
## 38 1e-01 2.000 0.3554348 0.08544121  
## 39 1e+00 2.000 0.3554348 0.08544121  
## 40 1e+01 2.000 0.3641304 0.09353863  
## 41 1e+02 2.000 0.3641304 0.09353863  
## 42 1e+03 2.000 0.3641304 0.09353863  
## 43 1e-02 3.000 0.3554348 0.08544121  
## 44 1e-01 3.000 0.3554348 0.08544121  
## 45 1e+00 3.000 0.3554348 0.08544121  
## 46 1e+01 3.000 0.3554348 0.08544121  
## 47 1e+02 3.000 0.3554348 0.08544121  
## 48 1e+03 3.000 0.3554348 0.08544121  
## 49 1e-02 4.000 0.3554348 0.08544121  
## 50 1e-01 4.000 0.3554348 0.08544121  
## 51 1e+00 4.000 0.3554348 0.08544121  
## 52 1e+01 4.000 0.3554348 0.08544121  
## 53 1e+02 4.000 0.3554348 0.08544121  
## 54 1e+03 4.000 0.3554348 0.08544121

##### (b) Train neural networks on the training set of the "South African Heart Disease" dataset, and evaluate its performance on the validation set.

set.seed(666)  
data <- trainSet  
n <- dim(trainSet)[1]  
index <- 1:n  
K <- 10  
flds <- createFolds(index, k=K)  
sizes <- seq(1, 20, 1)  
miscl.cv <- rep(NA, length(sizes))  
  
  
for (j in 1:length(sizes)){  
 miscl.cv.raw <- rep(NA, K)  
 for (i in 1:K){  
 testID <- flds[[i]]  
 data.tr <- data[-testID,]  
 #rownames(data.tr) <- c(1:dim(data.tr)[1])  
 ideal <- class.ind(data.tr$chd)  
 data.test <- data[testID,]  
 ann.cv <- nnet(data.tr[,-10], ideal, size=sizes[j], softmax=TRUE, maxit = 300, decay = 5e-4)  
 ann.cv.pred <- predict(ann.cv, data.test[,-10], type = "class")  
 miscl.cv.raw[i] <- mean(ann.cv.pred != data.test[,10])  
 }  
 miscl.cv[j] <- mean(miscl.cv.raw)  
}

plot(sizes, miscl.cv, type="l", xlab="size", ylab="misclassification rate", ylim=c(0.3, 0.5)  
 , main="Cross-validation of ANN", lwd=2)  
abline(v=sizes[which.min(miscl.cv)], col="red", lty=3)



cat("The best size for ANN is :", sizes[which.min(miscl.cv)],"\n")

## The best size for ANN is : 2

set.seed(345)  
ideal <- class.ind(trainSet$chd)  
ANN <- nnet(trainSet[,-10], ideal, size=2, softmax=TRUE, maxit = 300, decay = 5e-4)

## # weights: 26  
## initial value 157.951648   
## iter 10 value 150.259617  
## iter 20 value 144.766737  
## iter 30 value 137.735324  
## iter 40 value 135.837205  
## iter 50 value 135.568013  
## iter 60 value 135.553163  
## iter 70 value 134.067545  
## iter 80 value 129.454830  
## iter 90 value 124.422583  
## iter 100 value 122.581245  
## iter 110 value 122.517402  
## iter 120 value 122.451933  
## iter 130 value 122.437841  
## iter 140 value 122.393578  
## iter 150 value 122.307510  
## iter 160 value 122.302696  
## iter 170 value 122.301466  
## final value 122.301390   
## converged

mean(predict(ANN, trainSet[, -10], type="class") == trainSet$chd)

## [1] 0.6926407

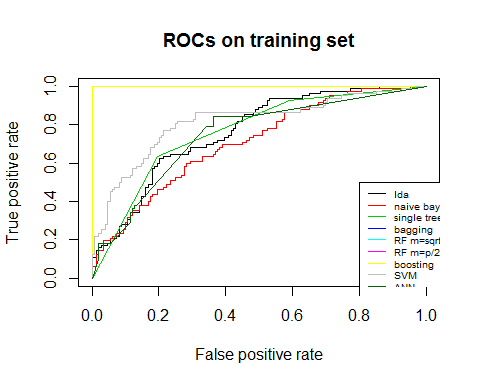
mean(predict(ANN, testSet[, -10], type="class") == testSet$chd)

## [1] 0.6363636

### 7. Summarize the classification results obtained for the "South African Heart Disease" in homework 3 and 4. Which method performed better, which performed worse? Discuss the possible reasons.

# ROCs on the training set  
# lda  
scores <- predict(lda.fit, newdata= trainPredictors)$posterior[,2]  
pred <- prediction( scores, labels= trainResponse )  
perf <- performance(pred, "tpr", "fpr")  
plot(perf, col= 1, main="ROCs on training set")  
# print out the area under the curve  
lda.train <- unlist(attributes(performance(pred, "auc"))$y.values)  
  
# naive bayes  
pred <- prediction(naiveBayes.train.scores, labels=trainResponse)  
perf <- performance(pred, "tpr", "fpr")  
  
# plot the ROC curve  
plot(perf, col= 2, add=T)  
# print out the area under the curve  
naive.train <- unlist(attributes(performance(pred, "auc"))$y.values)  
  
# single pruned tree  
scores <- predict(prune.tree, newdata= trainSet[,-10], type="vector")[,2]  
pred <- prediction(scores, labels= trainSet$chd )  
perf <- performance(pred, "tpr", "fpr")  
plot(perf, col= 3, add=T)  
# AUC  
single.tree.train <- unlist(attributes(performance(pred, "auc"))$y.values)  
  
# bagging  
pred.bagging <- predict(bagging.tree ,newdata =trainSet, type="prob")[,2]  
pred <- prediction(pred.bagging, labels= trainSet$chd )  
perf <- performance(pred, "tpr", "fpr")  
plot(perf, col= 4, add=T)  
  
# AUC  
bagging.train <- unlist(attributes(performance(pred, "auc"))$y.values)  
  
  
# random forest  
pred.rf.sqrt <- predict(rf.sqrt.tree ,newdata =trainSet, type="prob")[,2]  
pred <- prediction(pred.rf.sqrt, labels= trainSet$chd )  
perf <- performance(pred, "tpr", "fpr")  
plot(perf, col= 5, add=T)  
# AUC for rf.sqrt  
rf.sqrt.train <- unlist(attributes(performance(pred, "auc"))$y.values)  
  
pred.rf.by2 <- predict(rf.by2.tree ,newdata =trainSet, type="prob")[,2]  
pred <- prediction(pred.rf.by2, labels= trainSet$chd )  
perf <- performance(pred, "tpr", "fpr")  
plot(perf, col= 6, add=T)  
# AUC for rf.by2  
rf.by2.train <- unlist(attributes(performance(pred, "auc"))$y.values)  
  
# boosting  
boosting.pred <- predict(boosting.tree, trainSet, type='response', n.trees=1000)  
pred <- prediction(boosting.pred, labels= trainSet$chd )  
perf <- performance(pred, "tpr", "fpr")  
plot(perf, col= 7, add=T)  
  
# AUC for boosting  
boosting.train <- unlist(attributes(performance(pred, "auc"))$y.values)  
  
# SVM  
scores <- attributes(predict(svm.best, svm.trainSet, decision.values =TRUE))$decision.values  
pred <- prediction(scores, svm.trainSet$chd)  
perf <- performance(pred, "tpr", "fpr")  
plot(perf, col = 8, add=T)  
# AUC for svm  
svm.train <- unlist(attributes(performance(pred, "auc"))$y.values)  
  
# ANN  
scores <- predict(ANN, trainSet[,-10], type="raw")[,2]  
pred <- prediction(scores, svm.trainSet$chd)  
perf <- performance(pred, "tpr", "fpr")  
plot(perf, col = "darkgreen", add=T)  
# AUC for ANN

ANN.train <- unlist(attributes(performance(pred, "auc"))$y.values)  
legend(0.8,0.5, c("lda", "naive bayes", "single tree", "bagging", "RF m=sqrt(p)",  
 "RF m=p/2", "boosting", "SVM", "ANN") ,col=c(1:8, "darkgreen"), lty=1, cex=0.6)



# ROCs on the validation set  
# lda  
#scores <- predict(lda.fit, newdata= trainPredictors)$posterior[,2]  
scores <- predict(lda.fit, newdata = testPredictors)$posterior[,2]  
pred <- prediction( scores, labels= testResponse )  
perf <- performance(pred, "tpr", "fpr")  
plot(perf, col= 1, main="ROCs on validation set")  
  
  
# naive bayes  
pred <- prediction(naiveBayes.test.scores, labels=testResponse)  
perf <- performance(pred, "tpr", "fpr")  
  
# plot the ROC curve  
plot(perf, col= 2, add=T)  
  
  
# single pruned tree  
# ROC on the validation set  
scores <- predict(prune.tree, newdata= testSet[,-10], type="vector")[,2]  
pred <- prediction(scores, labels= testSet$chd )  
perf <- performance(pred, "tpr", "fpr")  
plot(perf, col= 3, add=T)  
# AUC  
single.tree.test <- unlist(attributes(performance(pred, "auc"))$y.values)  
  
# bagging  
pred.bagging <- predict(bagging.tree ,newdata =testSet, type="prob")[,2]  
pred <- prediction(pred.bagging, labels= testSet$chd )  
perf <- performance(pred, "tpr", "fpr")  
plot(perf, col= 4, add=T)  
# AUC  
bagging.test <- unlist(attributes(performance(pred, "auc"))$y.values)  
  
  
# random forest  
pred.rf.sqrt <- predict(rf.sqrt.tree ,newdata =testSet, type="prob")[,2]  
pred <- prediction(pred.rf.sqrt, labels= testSet$chd )  
perf <- performance(pred, "tpr", "fpr")  
plot(perf, col= 5, add=T)  
# AUC for rf.sqrt  
rf.sqrt.test <- unlist(attributes(performance(pred, "auc"))$y.values)  
  
#rf.by2.tree <- randomForest(as.factor(chd)~.,data=trainSet , mtry=9/2, importance =TRUE, ntree = 13)  
pred.rf.by2 <- predict(rf.by2.tree ,newdata =testSet, type="prob")[,2]  
pred <- prediction(pred.rf.by2, labels= testSet$chd )  
perf <- performance(pred, "tpr", "fpr")  
plot(perf, col= 6, add=T)  
# AUC for rf.by2  
rf.by2.test <- unlist(attributes(performance(pred, "auc"))$y.values)  
  
# boosting  
boosting.pred <- predict(boosting.tree, testSet, type='response', n.trees=1000)  
pred <- prediction(boosting.pred, labels= testSet$chd )  
perf <- performance(pred, "tpr", "fpr")  
plot(perf, col= 7, add=T)  
  
# AUC for boosting  
boosting.test <- unlist(attributes(performance(pred, "auc"))$y.values)  
  
# SVM  
scores <- attributes(predict(svm.best, svm.testSet, decision.values =TRUE))$decision.values  
pred <- prediction(scores, svm.testSet$chd)  
perf <- performance(pred, "tpr", "fpr")  
plot(perf, col = 8, add=T)  
# AUC for svm  
svm.test <- unlist(attributes(performance(pred, "auc"))$y.values)  
  
# ANN  
scores <- predict(ANN, testSet[,-10], type="raw")[,2]  
pred <- prediction(scores, svm.testSet$chd)  
perf <- performance(pred, "tpr", "fpr")  
plot(perf, col = "darkgreen", add=T)  
# AUC for ANN  
  
ANN.test <- unlist(attributes(performance(pred, "auc"))$y.values)  
legend(0.8,0.5, c("lda", "naive bayes", "single tree", "bagging", "RF m=sqrt(p)",  
 "RF m=p/2", "boosting", "SVM", "ANN") ,col=c(1:8, "darkgreen"), lty=1, cex=0.6)

