Epi\_ML\_7\_Assignment

Arielle Coq / AC4140

3/3/2020

### Question 1

First cleaned the data the same way that was done in class and partition the data into testing and training data sets to 30/70 partition. Removed the missing values and also created the outcome varaible so that it is Heart disease present or not present. The seed is set at 100 for this analysis.

heart.data <- read.csv("processed.cleveland.data", header=FALSE)  
  
var.names<-c("age", "sex", "pain\_type", "resting\_sysbp", "chol", "fast\_blsugar\_gt120", "rest\_ecg", "max\_hr", "exerc\_angina", "ST\_depression", "ST\_slope", "vessels\_colorflu", "defect", "heart\_disease\_present")  
  
colnames(heart.data)<-var.names  
str(heart.data)

## 'data.frame': 303 obs. of 14 variables:  
## $ age : num 63 67 67 37 41 56 62 57 63 53 ...  
## $ sex : num 1 1 1 1 0 1 0 0 1 1 ...  
## $ pain\_type : num 1 4 4 3 2 2 4 4 4 4 ...  
## $ resting\_sysbp : num 145 160 120 130 130 120 140 120 130 140 ...  
## $ chol : num 233 286 229 250 204 236 268 354 254 203 ...  
## $ fast\_blsugar\_gt120 : num 1 0 0 0 0 0 0 0 0 1 ...  
## $ rest\_ecg : num 2 2 2 0 2 0 2 0 2 2 ...  
## $ max\_hr : num 150 108 129 187 172 178 160 163 147 155 ...  
## $ exerc\_angina : num 0 1 1 0 0 0 0 1 0 1 ...  
## $ ST\_depression : num 2.3 1.5 2.6 3.5 1.4 0.8 3.6 0.6 1.4 3.1 ...  
## $ ST\_slope : num 3 2 2 3 1 1 3 1 2 3 ...  
## $ vessels\_colorflu : Factor w/ 5 levels "?","0.0","1.0",..: 2 5 4 2 2 2 4 2 3 2 ...  
## $ defect : Factor w/ 4 levels "?","3.0","6.0",..: 3 2 4 2 2 2 2 2 4 4 ...  
## $ heart\_disease\_present: int 0 2 1 0 0 0 3 0 2 1 ...

heart.data[heart.data=="?"]<-NA  
  
heart.data$defect<-as.numeric(factor(heart.data$defect))  
heart.data$vessels\_colorflu<-as.numeric(factor(heart.data$vessels\_colorflu))  
  
heart.data$outcome<-ifelse(heart.data$heart\_disease\_present==0, 0,1)  
heart.data$heart\_disease\_present<-NULL  
heart.data$outcome<-factor(heart.data$outcome)  
levels(heart.data$outcome)<-c("Not Present", "Present")  
str(heart.data)

## 'data.frame': 303 obs. of 14 variables:  
## $ age : num 63 67 67 37 41 56 62 57 63 53 ...  
## $ sex : num 1 1 1 1 0 1 0 0 1 1 ...  
## $ pain\_type : num 1 4 4 3 2 2 4 4 4 4 ...  
## $ resting\_sysbp : num 145 160 120 130 130 120 140 120 130 140 ...  
## $ chol : num 233 286 229 250 204 236 268 354 254 203 ...  
## $ fast\_blsugar\_gt120: num 1 0 0 0 0 0 0 0 0 1 ...  
## $ rest\_ecg : num 2 2 2 0 2 0 2 0 2 2 ...  
## $ max\_hr : num 150 108 129 187 172 178 160 163 147 155 ...  
## $ exerc\_angina : num 0 1 1 0 0 0 0 1 0 1 ...  
## $ ST\_depression : num 2.3 1.5 2.6 3.5 1.4 0.8 3.6 0.6 1.4 3.1 ...  
## $ ST\_slope : num 3 2 2 3 1 1 3 1 2 3 ...  
## $ vessels\_colorflu : num 1 4 3 1 1 1 3 1 2 1 ...  
## $ defect : num 2 1 3 1 1 1 1 1 3 3 ...  
## $ outcome : Factor w/ 2 levels "Not Present",..: 1 2 2 1 1 1 2 1 2 2 ...

summary(heart.data)

## age sex pain\_type resting\_sysbp   
## Min. :29.00 Min. :0.0000 Min. :1.000 Min. : 94.0   
## 1st Qu.:48.00 1st Qu.:0.0000 1st Qu.:3.000 1st Qu.:120.0   
## Median :56.00 Median :1.0000 Median :3.000 Median :130.0   
## Mean :54.44 Mean :0.6799 Mean :3.158 Mean :131.7   
## 3rd Qu.:61.00 3rd Qu.:1.0000 3rd Qu.:4.000 3rd Qu.:140.0   
## Max. :77.00 Max. :1.0000 Max. :4.000 Max. :200.0   
##   
## chol fast\_blsugar\_gt120 rest\_ecg max\_hr   
## Min. :126.0 Min. :0.0000 Min. :0.0000 Min. : 71.0   
## 1st Qu.:211.0 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:133.5   
## Median :241.0 Median :0.0000 Median :1.0000 Median :153.0   
## Mean :246.7 Mean :0.1485 Mean :0.9901 Mean :149.6   
## 3rd Qu.:275.0 3rd Qu.:0.0000 3rd Qu.:2.0000 3rd Qu.:166.0   
## Max. :564.0 Max. :1.0000 Max. :2.0000 Max. :202.0   
##   
## exerc\_angina ST\_depression ST\_slope vessels\_colorflu  
## Min. :0.0000 Min. :0.00 Min. :1.000 Min. :1.000   
## 1st Qu.:0.0000 1st Qu.:0.00 1st Qu.:1.000 1st Qu.:1.000   
## Median :0.0000 Median :0.80 Median :2.000 Median :1.000   
## Mean :0.3267 Mean :1.04 Mean :1.601 Mean :1.672   
## 3rd Qu.:1.0000 3rd Qu.:1.60 3rd Qu.:2.000 3rd Qu.:2.000   
## Max. :1.0000 Max. :6.20 Max. :3.000 Max. :4.000   
## NA's :4   
## defect outcome   
## Min. :1.000 Not Present:164   
## 1st Qu.:1.000 Present :139   
## Median :1.000   
## Mean :1.837   
## 3rd Qu.:3.000   
## Max. :3.000   
## NA's :2

#Remove the missings  
heart.data.nomiss<-na.omit(heart.data)  
  
#Set No Heart Disease as Reference Level  
heart.data.nomiss$outcome<-relevel(heart.data.nomiss$outcome, ref="Not Present")

set.seed(100)  
train.indices<-createDataPartition(y=heart.data.nomiss$outcome,p=0.7,list=FALSE)  
  
training<-heart.data.nomiss[train.indices,]  
testing<-heart.data.nomiss[-train.indices,]

### Question 2

The next step in this analysis would be to run a single classification tree (categorical outcome) using all the features in the dataset. Will also calculate an evaluation metrics and outupt the important variables metrics.

train.control<-trainControl(method="cv", number=10)  
grid.1<-expand.grid(cp=seq(0.001, 0.3, by=0.01))  
tree.HD<-train(outcome~., data=training, method="rpart",trControl=train.control, tuneGrid=grid.1)  
tree.HD$bestTune

## cp  
## 3 0.021

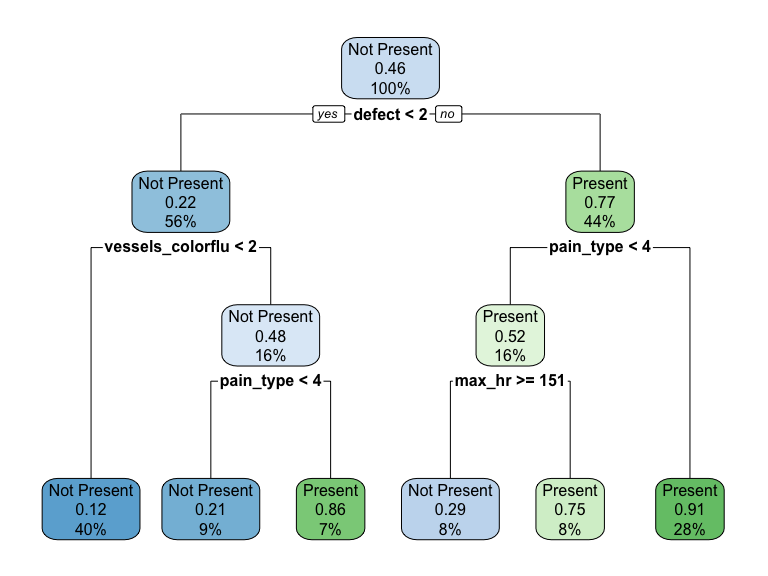
tree.HD

## CART   
##   
## 208 samples  
## 13 predictor  
## 2 classes: 'Not Present', 'Present'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 188, 188, 188, 187, 187, 187, ...   
## Resampling results across tuning parameters:  
##   
## cp Accuracy Kappa   
## 0.001 0.7932251 0.5840945  
## 0.011 0.8027489 0.6024348  
## 0.021 0.8027489 0.6024348  
## 0.031 0.7746104 0.5471660  
## 0.041 0.7843723 0.5703408  
## 0.051 0.7703247 0.5404576  
## 0.061 0.7700866 0.5378785  
## 0.071 0.7750866 0.5474825  
## 0.081 0.7750866 0.5474825  
## 0.091 0.7750866 0.5474825  
## 0.101 0.7750866 0.5474825  
## 0.111 0.7750866 0.5474825  
## 0.121 0.7750866 0.5474825  
## 0.131 0.7750866 0.5474825  
## 0.141 0.7750866 0.5474825  
## 0.151 0.7750866 0.5474825  
## 0.161 0.7750866 0.5474825  
## 0.171 0.7750866 0.5474825  
## 0.181 0.7750866 0.5474825  
## 0.191 0.7750866 0.5474825  
## 0.201 0.7750866 0.5474825  
## 0.211 0.7750866 0.5474825  
## 0.221 0.7750866 0.5474825  
## 0.231 0.7750866 0.5474825  
## 0.241 0.7750866 0.5474825  
## 0.251 0.7750866 0.5474825  
## 0.261 0.7750866 0.5474825  
## 0.271 0.7750866 0.5474825  
## 0.281 0.7750866 0.5474825  
## 0.291 0.7750866 0.5474825  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was cp = 0.021.

varImp(tree.HD)

## rpart variable importance  
##   
## Overall  
## pain\_type 100.00  
## vessels\_colorflu 91.63  
## defect 72.76  
## exerc\_angina 71.78  
## max\_hr 70.39  
## ST\_depression 25.32  
## age 17.18  
## ST\_slope 14.84  
## sex 13.51  
## chol 6.77  
## rest\_ecg 0.00  
## fast\_blsugar\_gt120 0.00  
## resting\_sysbp 0.00

rpart.plot(tree.HD$finalModel)



pred.HD<-predict(tree.HD, testing)  
pred.HD.prob<-predict(tree.HD, testing, type="prob")  
  
eval.results<-confusionMatrix(pred.HD, testing$outcome, positive = "Present")  
print(eval.results)

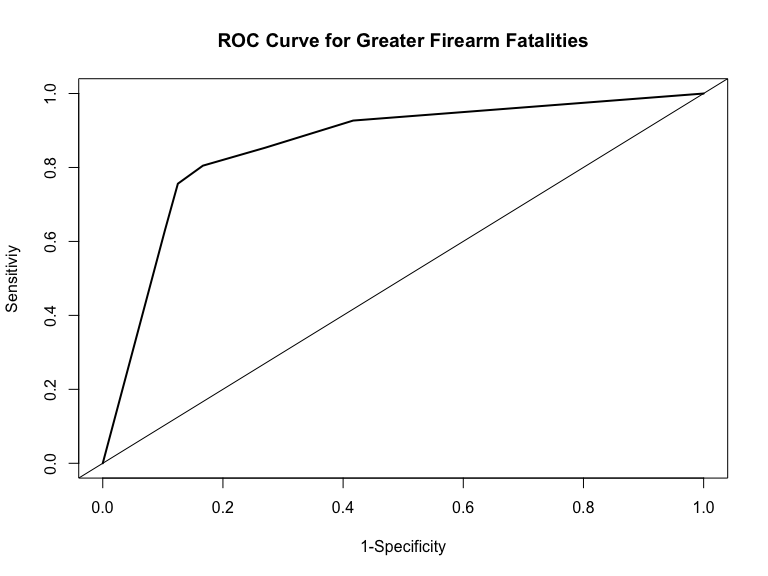
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Not Present Present  
## Not Present 40 8  
## Present 8 33  
##   
## Accuracy : 0.8202   
## 95% CI : (0.7245, 0.8936)  
## No Information Rate : 0.5393   
## P-Value [Acc > NIR] : 2.567e-08   
##   
## Kappa : 0.6382   
##   
## Mcnemar's Test P-Value : 1   
##   
## Sensitivity : 0.8049   
## Specificity : 0.8333   
## Pos Pred Value : 0.8049   
## Neg Pred Value : 0.8333   
## Prevalence : 0.4607   
## Detection Rate : 0.3708   
## Detection Prevalence : 0.4607   
## Balanced Accuracy : 0.8191   
##   
## 'Positive' Class : Present   
##

analysis <- roc(response=testing$outcome, predictor=pred.HD.prob[,2])

## Setting levels: control = Not Present, case = Present

## Setting direction: controls < cases

plot(1-analysis$specificities,analysis$sensitivities,type="l",  
ylab="Sensitiviy",xlab="1-Specificity",col="black",lwd=2,  
main = "ROC Curve for Greater Firearm Fatalities")  
abline(a=0,b=1)



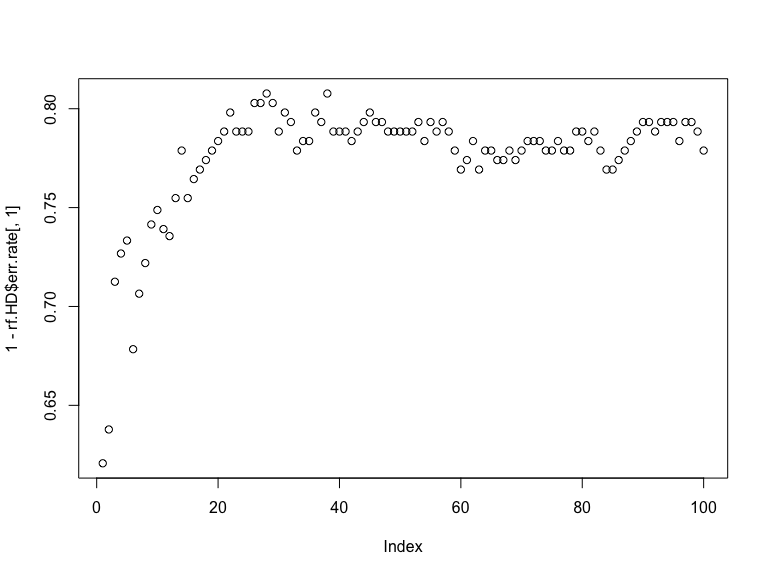
### Question 3

The next thing I will do is run a random forest to classify heart disease. I will set up a pipeline to try different values of mtry and different numbers of trees to obtain your optimal model. After this I will then run the evaluation metrics and outputting the important variables metrics.

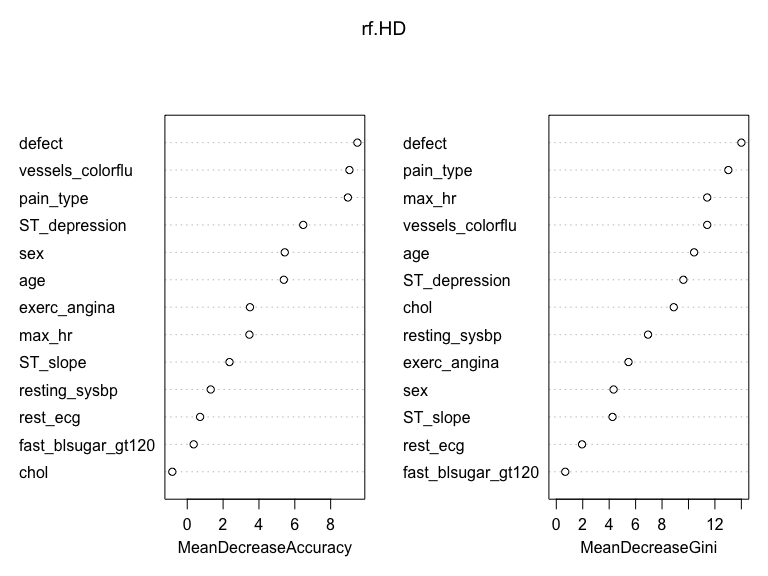
rf.HD<-randomForest(outcome ~., data=training, mtry=sqrt(ncol(training)-1), importance=TRUE, ntree=100)  
  
rf.HD

##   
## Call:  
## randomForest(formula = outcome ~ ., data = training, mtry = sqrt(ncol(training) - 1), importance = TRUE, ntree = 100)   
## Type of random forest: classification  
## Number of trees: 100  
## No. of variables tried at each split: 4  
##   
## OOB estimate of error rate: 22.12%  
## Confusion matrix:  
## Not Present Present class.error  
## Not Present 93 19 0.1696429  
## Present 27 69 0.2812500

#Create plot of Accuracy by tree index   
plot(1-rf.HD$err.rate[,1])



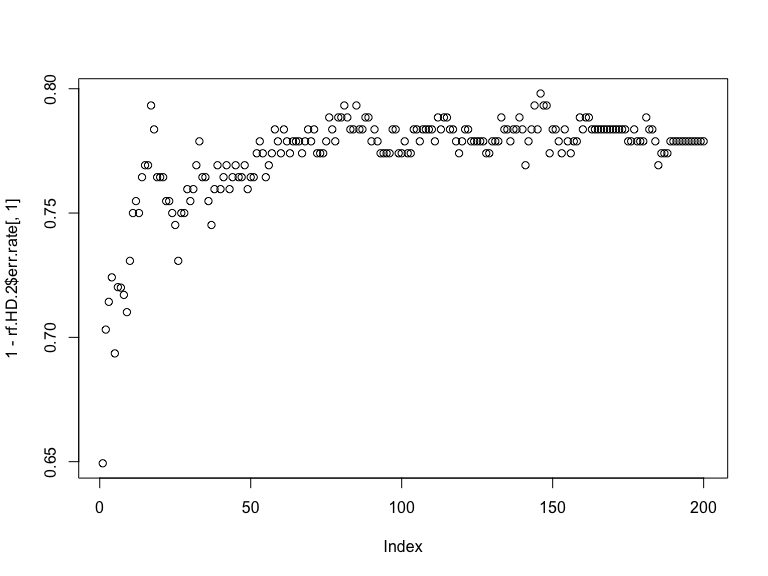
#Examine variable importance plots  
varImpPlot(rf.HD)



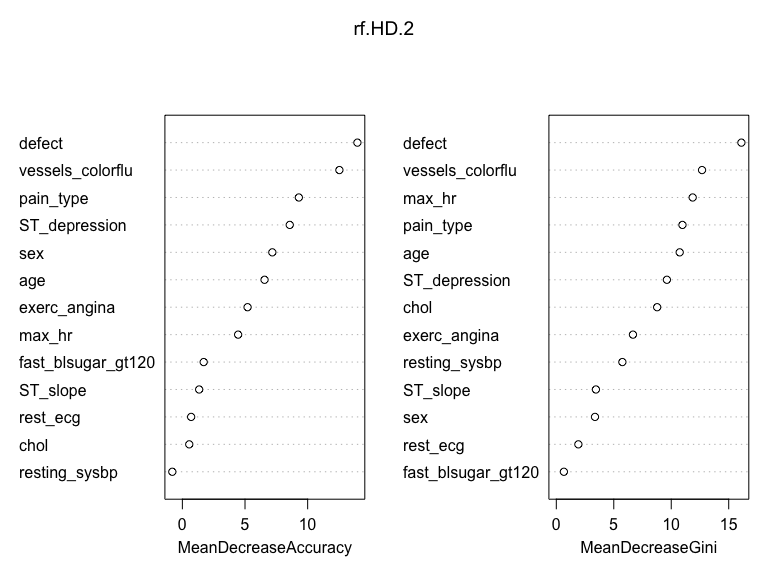
rf.HD.2<-randomForest(outcome ~., data=training, mtry=sqrt(ncol(training)-1), importance=TRUE, ntree=200)  
  
rf.HD.2

##   
## Call:  
## randomForest(formula = outcome ~ ., data = training, mtry = sqrt(ncol(training) - 1), importance = TRUE, ntree = 200)   
## Type of random forest: classification  
## Number of trees: 200  
## No. of variables tried at each split: 4  
##   
## OOB estimate of error rate: 22.12%  
## Confusion matrix:  
## Not Present Present class.error  
## Not Present 91 21 0.1875000  
## Present 25 71 0.2604167

#Create plot of Accuracy by tree index   
plot(1-rf.HD.2$err.rate[,1])



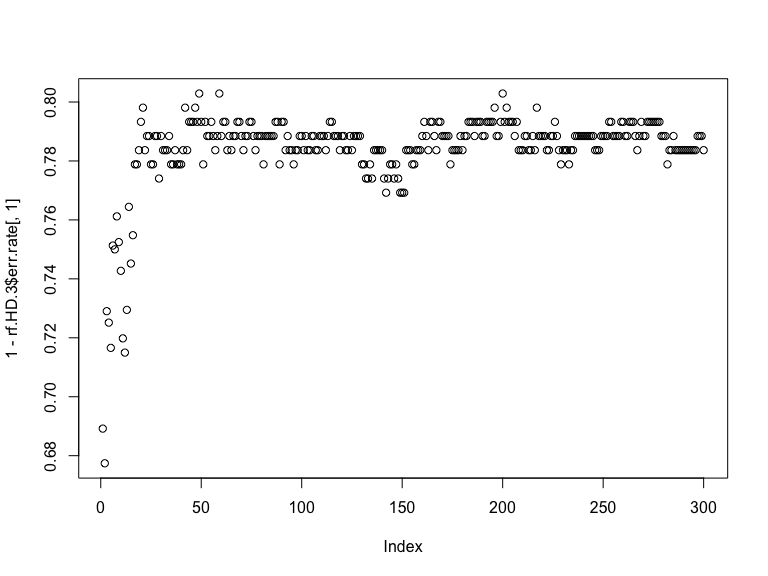
#Examine variable importance plots  
varImpPlot(rf.HD.2)



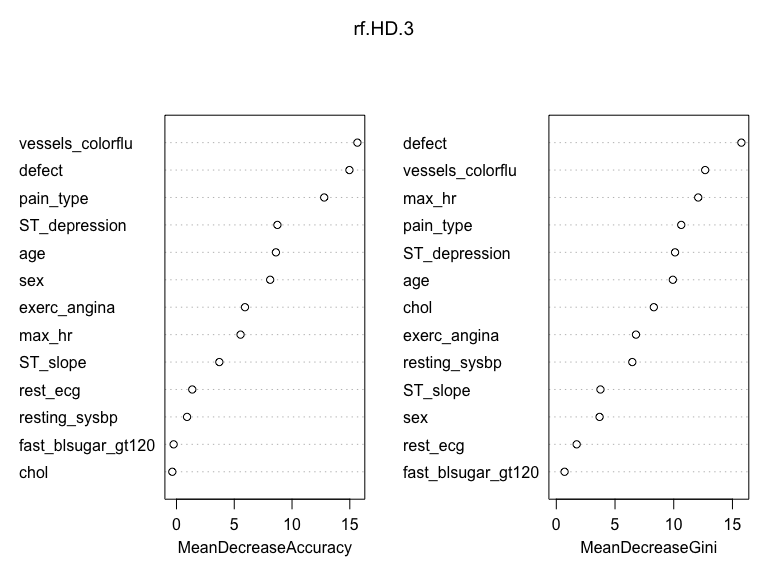
rf.HD.3<-randomForest(outcome ~., data=training, mtry=sqrt(ncol(training)-1), importance=TRUE, ntree=300)  
  
rf.HD.3

##   
## Call:  
## randomForest(formula = outcome ~ ., data = training, mtry = sqrt(ncol(training) - 1), importance = TRUE, ntree = 300)   
## Type of random forest: classification  
## Number of trees: 300  
## No. of variables tried at each split: 4  
##   
## OOB estimate of error rate: 21.63%  
## Confusion matrix:  
## Not Present Present class.error  
## Not Present 92 20 0.1785714  
## Present 25 71 0.2604167

#Create plot of Accuracy by tree index   
plot(1-rf.HD.3$err.rate[,1])



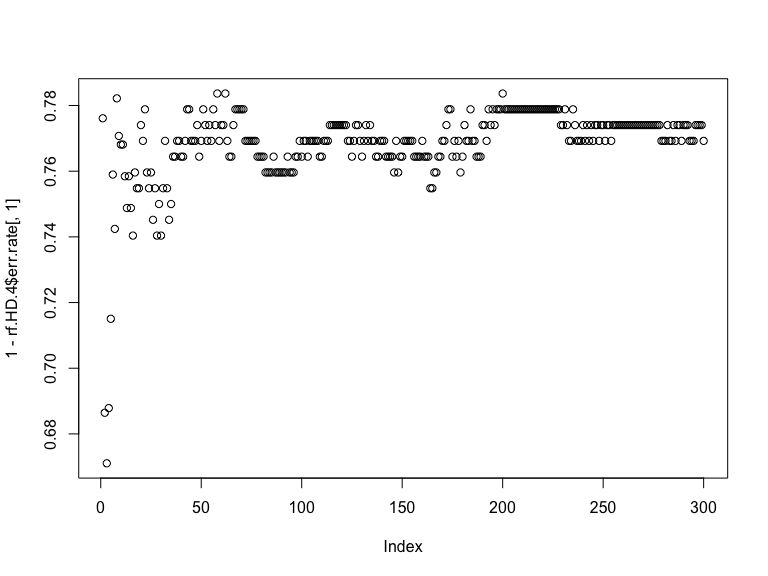
#Examine variable importance plots  
varImpPlot(rf.HD.3)



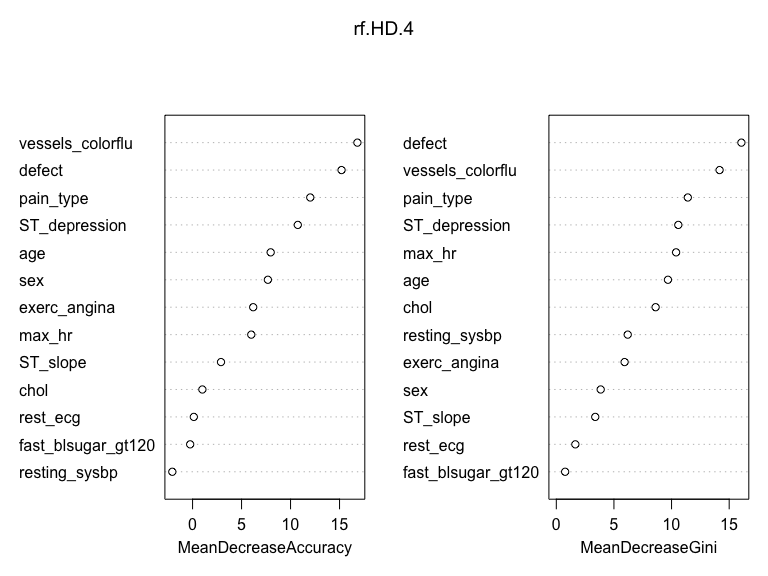
#Vary value of mtry-First decrease from 8 to 5  
  
rf.HD.4<-randomForest(outcome ~., data=training, mtry=5, importance=TRUE, ntree=300)  
  
rf.HD.4

##   
## Call:  
## randomForest(formula = outcome ~ ., data = training, mtry = 5, importance = TRUE, ntree = 300)   
## Type of random forest: classification  
## Number of trees: 300  
## No. of variables tried at each split: 5  
##   
## OOB estimate of error rate: 23.08%  
## Confusion matrix:  
## Not Present Present class.error  
## Not Present 90 22 0.1964286  
## Present 26 70 0.2708333

#Create plot of Accuracy by tree index   
plot(1-rf.HD.4$err.rate[,1])



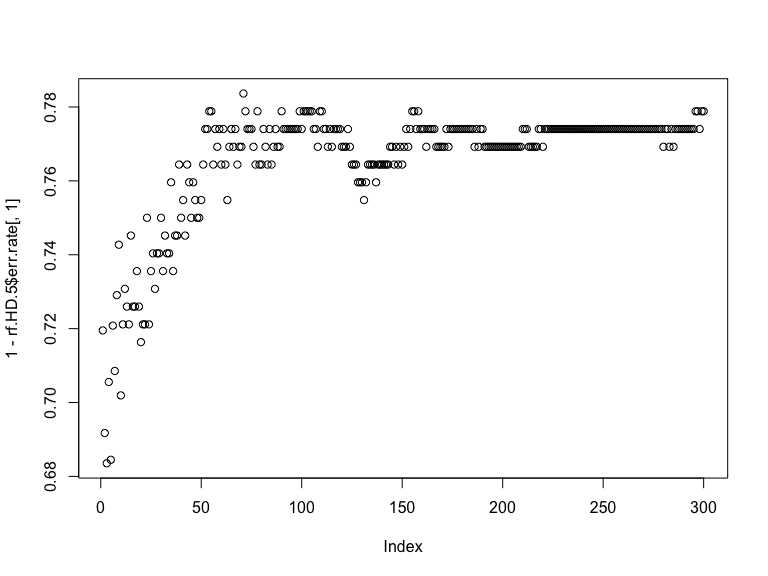
#Examine variable importance plots  
varImpPlot(rf.HD.4)



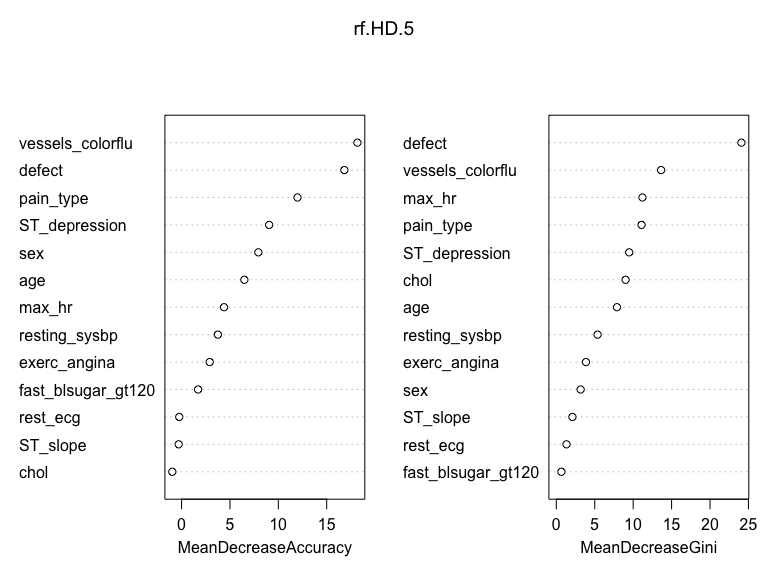
#Vary value of mtry-Try increase to 20  
  
rf.HD.5<-randomForest(outcome ~., data=training, mtry=20, importance=TRUE, ntree=300)  
  
rf.HD.5

##   
## Call:  
## randomForest(formula = outcome ~ ., data = training, mtry = 20, importance = TRUE, ntree = 300)   
## Type of random forest: classification  
## Number of trees: 300  
## No. of variables tried at each split: 13  
##   
## OOB estimate of error rate: 22.12%  
## Confusion matrix:  
## Not Present Present class.error  
## Not Present 90 22 0.1964286  
## Present 24 72 0.2500000

#Create plot of Accuracy by tree index   
plot(1-rf.HD.5$err.rate[,1])



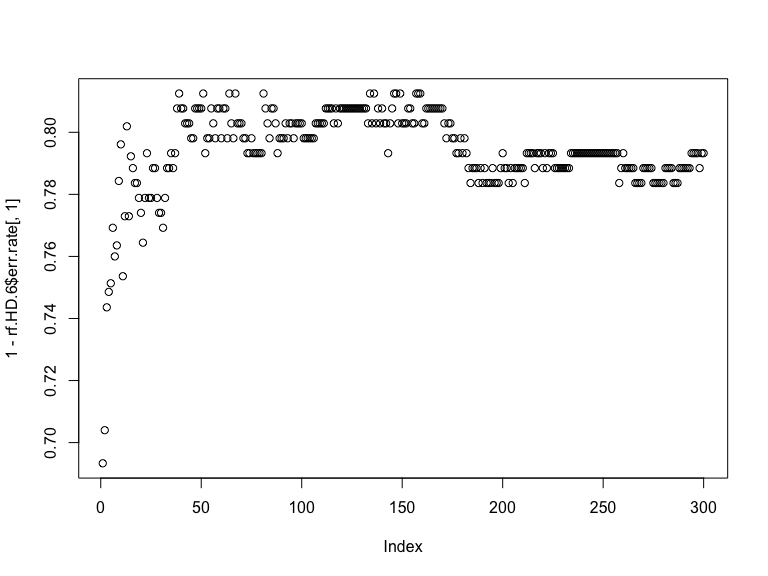
#Examine variable importance plots  
varImpPlot(rf.HD.5)



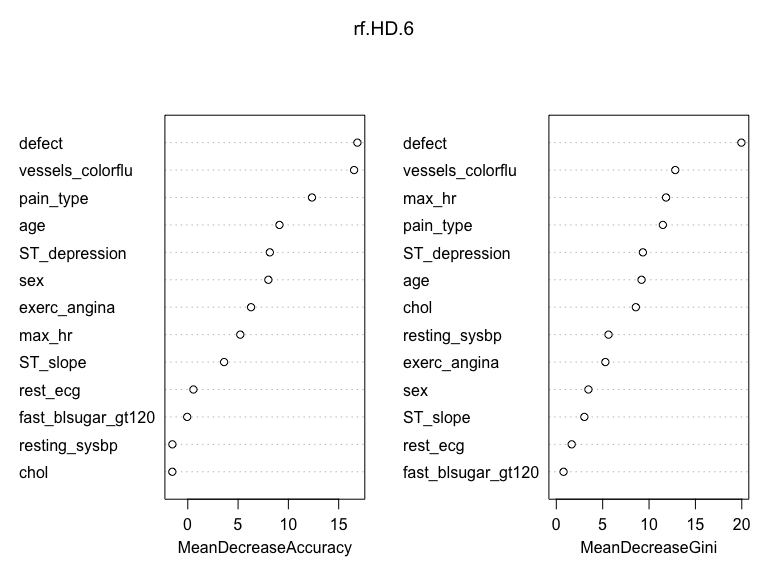
#Vary value of mtry-Try increase to half of the features, 33  
  
rf.HD.6<-randomForest(outcome ~., data=training, mtry=0.5\*(ncol(training)-1), importance=TRUE, ntree=300)  
  
rf.HD.6

##   
## Call:  
## randomForest(formula = outcome ~ ., data = training, mtry = 0.5 \* (ncol(training) - 1), importance = TRUE, ntree = 300)   
## Type of random forest: classification  
## Number of trees: 300  
## No. of variables tried at each split: 6  
##   
## OOB estimate of error rate: 20.67%  
## Confusion matrix:  
## Not Present Present class.error  
## Not Present 93 19 0.1696429  
## Present 24 72 0.2500000

#Create plot of Accuracy by tree index   
plot(1-rf.HD.6$err.rate[,1])



#Examine variable importance plots  
varImpPlot(rf.HD.6)



### Question 4

Are there differences in variable importance that you see between a single tree and an ensemble metric?

Yes, there are some difference in variable importance that I see between a single tree and an ensemble metric. The variable importance for the single classification tree is pain\_type while in the ensemble the most importnat variable in this model is defect.

Are there differences observed across the different variable importance metrics output from the ensemble?

Yes, there are some differences across the different variable importance metrics output from the ensemble. Some metrics say that defect is the most imoportant variable while others say that vessels\_colorflu is the most important model.

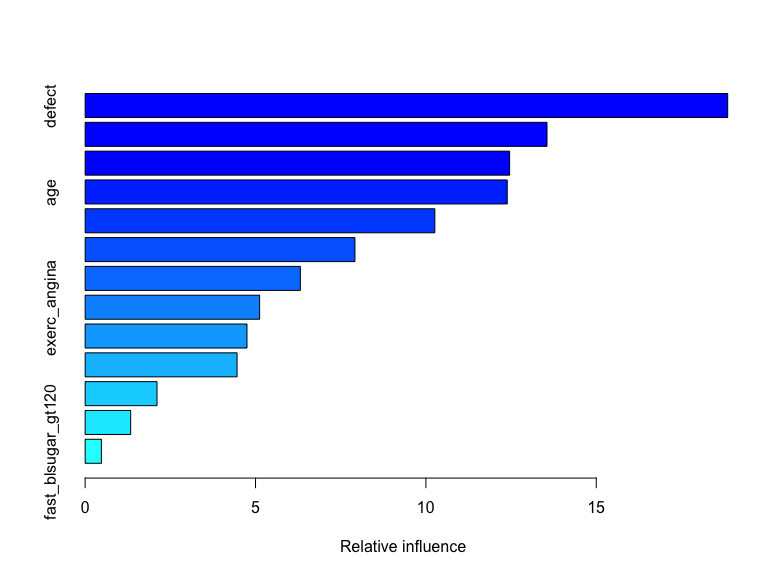
How do you interpret those differences?

Those differences are interpretted as which variables are used in the model and which varible explains the decrease in accuracies that is observed.

### Question 5

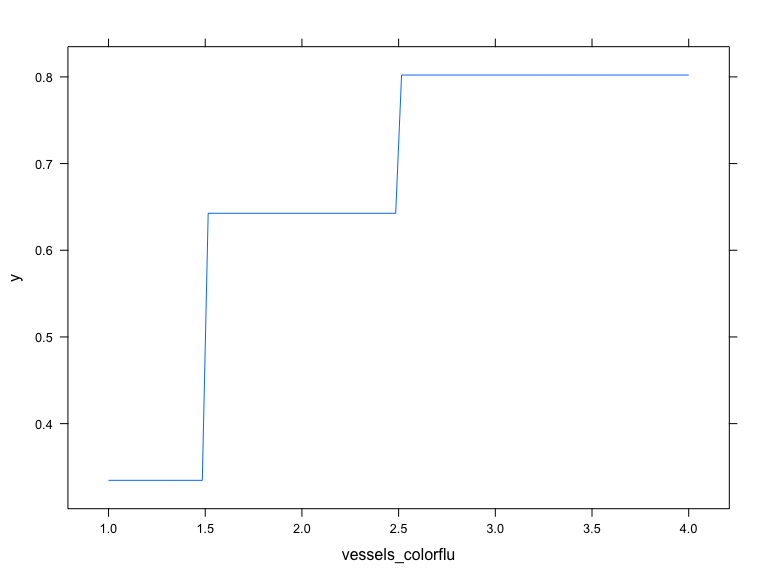
The next thing that I will do is use a boosting algorithm and tune to obtain your optimal model. Compare to the results from the single classification tree and the random forest.

train.data=  
 training %>%  
 mutate(outcome.num = recode(outcome, "Present"= 1, "Not Present" = 0))  
  
train.data$outcome<-NULL  
  
#leaving cvar importance as the default  
gbm.outcome<-gbm(outcome.num ~., data=train.data, distribution='bernoulli', n.trees=2000, shrinkage=0.01)  
summary(gbm.outcome)



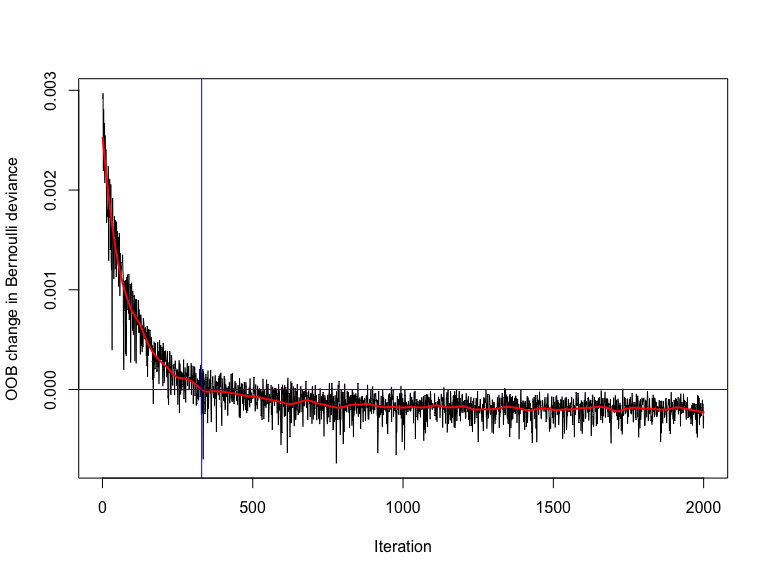
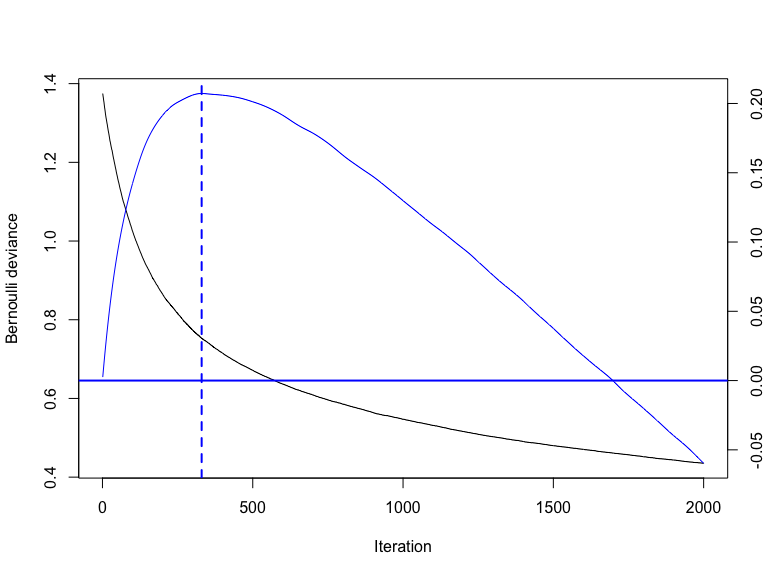
## var rel.inf  
## defect defect 18.8566188  
## vessels\_colorflu vessels\_colorflu 13.5547144  
## pain\_type pain\_type 12.4568288  
## age age 12.3868610  
## ST\_depression ST\_depression 10.2624112  
## max\_hr max\_hr 7.9180951  
## chol chol 6.3148933  
## exerc\_angina exerc\_angina 5.1203699  
## resting\_sysbp resting\_sysbp 4.7495449  
## sex sex 4.4584132  
## ST\_slope ST\_slope 2.1089542  
## rest\_ecg rest\_ecg 1.3358641  
## fast\_blsugar\_gt120 fast\_blsugar\_gt120 0.4764309

#plot of the marginal effect of the selected variable  
plot.gbm(gbm.outcome, 'vessels\_colorflu', type="response")



#Plot loss function as a result of n trees added to the ensemble  
gbm.perf(gbm.outcome, plot.it=TRUE, oobag.curve=TRUE, overlay=TRUE, method='OOB')

## OOB generally underestimates the optimal number of iterations although predictive performance is reasonably competitive. Using cv\_folds>1 when calling gbm usually results in improved predictive performance.



## [1] 330  
## attr(,"smoother")  
## Call:  
## loess(formula = object$oobag.improve ~ x, enp.target = min(max(4,   
## length(x)/10), 50))  
##   
## Number of Observations: 2000   
## Equivalent Number of Parameters: 39.99   
## Residual Standard Error: 0.0001289

pred.gbm.outcome<-predict(gbm.outcome, train.data, n.trees=2000, type="response")  
  
pred.gbm.outcome.class<-round(pred.gbm.outcome)  
  
misClasificError <- mean(pred.gbm.outcome.class != train.data$outcome.num)  
print(paste('Accuracy Model',1-misClasificError))

## [1] "Accuracy Model 0.923076923076923"

### Question 6

Which model performs the best? Provide justification for your answer.

The model that performed best in this analysis would be the single classification with an accuracy of 82%. The accurancy in the boosting random forest is 92% which suggest some overfitting happening in the model, therefore the single classification tree would be better because it would be more generalizable to other populations.

### Question 7

How do these results compare to the SVC analysis we did back in Class 6?

Code from class 6- Suppoert Vector Machine

set.seed(100)  
svm.heart<-svm(outcome ~ ., data=training, kernel="linear", cost=1, scale=TRUE)  
print(svm.heart)

##   
## Call:  
## svm(formula = outcome ~ ., data = training, kernel = "linear",   
## cost = 1, scale = TRUE)  
##   
##   
## Parameters:  
## SVM-Type: C-classification   
## SVM-Kernel: linear   
## cost: 1   
##   
## Number of Support Vectors: 74

### Cost- hyper parameter   
svm.pred<-predict(svm.heart, newdata=training[,1:13])  
table(svm.pred, training$outcome)

##   
## svm.pred Not Present Present  
## Not Present 101 18  
## Present 11 78

misClasificError <- mean(svm.pred != training$outcome, na.rm=T)  
print(paste('Accuracy Model 1',1-misClasificError))

## [1] "Accuracy Model 1 0.860576923076923"

features<-training[,1:13]  
outcome<-training$outcome  
  
svm\_tune <- tune(svm, train.x=features, train.y=outcome, kernel="linear", range=list(cost=10^(-1:2)))  
  
summary(svm\_tune)

##   
## Parameter tuning of 'svm':  
##   
## - sampling method: 10-fold cross validation   
##   
## - best parameters:  
## cost  
## 0.1  
##   
## - best performance: 0.1490476   
##   
## - Detailed performance results:  
## cost error dispersion  
## 1 0.1 0.1490476 0.1041843  
## 2 1.0 0.1492857 0.1044250  
## 3 10.0 0.1590476 0.1154799  
## 4 100.0 0.1590476 0.1154799

svm.heart.new<-svm(outcome ~ ., data=training, kernel="linear", cost=0.1, scale=TRUE)  
  
print(svm.heart.new)

##   
## Call:  
## svm(formula = outcome ~ ., data = training, kernel = "linear",   
## cost = 0.1, scale = TRUE)  
##   
##   
## Parameters:  
## SVM-Type: C-classification   
## SVM-Kernel: linear   
## cost: 0.1   
##   
## Number of Support Vectors: 86

svm.pred.new<-predict(svm.heart.new, newdata=training[,1:13])  
table(svm.pred.new, training$outcome)

##   
## svm.pred.new Not Present Present  
## Not Present 102 18  
## Present 10 78

misClasificError.new <- mean(svm.pred.new != training$outcome, na.rm=T)  
print(paste('Accuracy Model 1',1-misClasificError.new))

## [1] "Accuracy Model 1 0.865384615384615"

The Accuracy for the support vector machine with linear kernel is 86% which is in between both the random forest (92%) and the single classification tree (82%) and therefore might be the best model to use in this case.