# Project 2 - Regression and CART

Group 3

3/13/2021

Group Leader: Brianna Johnson

Member Names: Marco Sousa, Ben Pfeffer, Nikita Seleznev, Brianna Johnson

## Introduction to the Heart Dataset

Observing the data:

```
##   Age Sex    ChestPain RestBP Chol Fbs RestECG MaxHR ExAng Oldpeak Slope Ca
## 1  63   1      typical    145  233   1       2   150     0     2.3     3  0
## 2  67   1 asymptomatic    160  286   0       2   108     1     1.5     2  3
## 3  67   1 asymptomatic    120  229   0       2   129     1     2.6     2  2
## 4  37   1   nonanginal    130  250   0       0   187     0     3.5     3  0
## 5  41   0   nontypical    130  204   0       2   172     0     1.4     1  0
## 6  56   1   nontypical    120  236   0       0   178     0     0.8     1  0
##         Thal AHD AHDBinary
## 1      fixed  No         0
## 2     normal Yes         1
## 3 reversable Yes         1
## 4     normal  No         0
## 5     normal  No         0
## 6     normal  No         0
```

The ‘Heart’ dataset considers AHD, the binary relationship of having heart disease or not, and other perhaps associated features. The dataset contains 303 observations and 14 attributes. Some attributes like MaxHR and RestBP are numerical, yet others like Thal or ChestPain are categorical. The direct link to the data from statlearning can be found here.

In particular, we are interested in constructing a model that correctly predicts whether AHD will be “Yes”, or “No”, as a manner of classification, based on other presented features of the data. We investigate three techniques of carrying out such a classification task: logistic regression, CART, and random forests.

The data is displayed as follows.

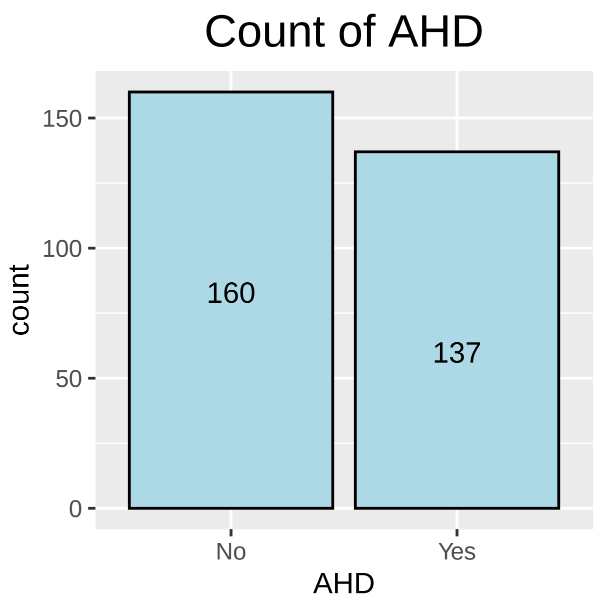
```
head(data)
```

```
##   Age Sex    ChestPain RestBP Chol Fbs RestECG MaxHR ExAng Oldpeak Slope Ca
## 1  63   1      typical    145  233   1       2   150     0     2.3     3  0
## 2  67   1 asymptomatic    160  286   0       2   108     1     1.5     2  3
## 3  67   1 asymptomatic    120  229   0       2   129     1     2.6     2  2
## 4  37   1   nonanginal    130  250   0       0   187     0     3.5     3  0
## 5  41   0   nontypical    130  204   0       2   172     0     1.4     1  0
## 6  56   1   nontypical    120  236   0       0   178     0     0.8     1  0
##         Thal AHD AHDBinary
## 1      fixed  No         0
## 2     normal Yes         1
## 3 reversable Yes         1
## 4     normal  No         0
## 5     normal  No         0
## 6     normal  No         0
```

## Exploratory Data Analysis

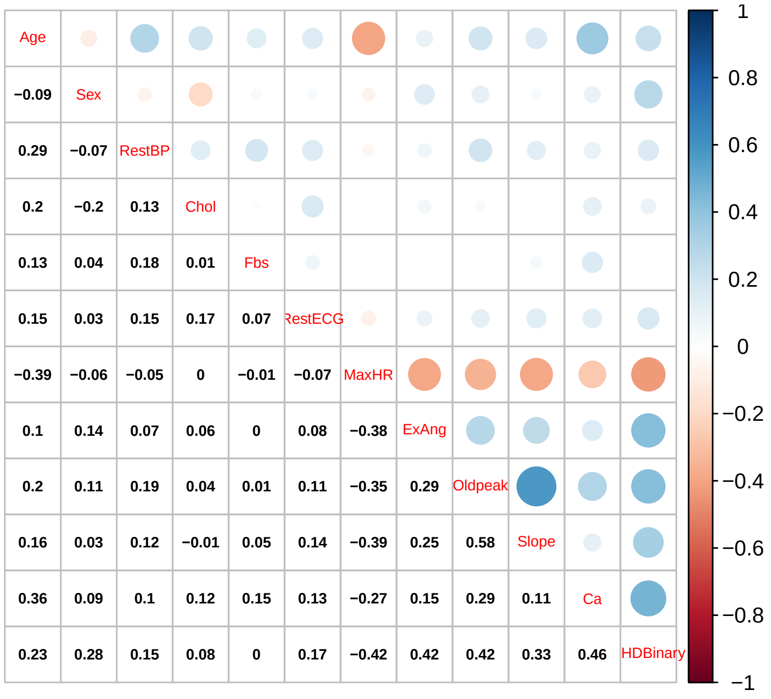
### Count of Binary Outcome

The following is a simple barplot of the count for the binary AHD outcome. We can see there are some more "No", than "Yes". More precisely, there are 160 "No", and 137 "Yes".



## Correlation matrix

No correlation among AHD Binary exceeds 0.5, naturally.



# Logistic Model

A logistic model is a standard machine learning model used to predict the probability of a binary event, like win lose, yes no, and so on. This does so, briefly speaking, by using the logit transofrmation of the odds to activate our logistic regression formula, resembling p(event)=$\frac{e^{X\beta}}{1+e^{X\beta}}$. In this application, we are interested in calculating a probability for "Yes" versus "No" (or 1 vs 0) for AHD using the other data features.

## Logistic Model

Constructing a logistic model for AHD based on other features. Training and testing sets were separated randomly in a 70/30 split. To construct a final logistic model, features with the highest p values were removed until the remaining attributes were all significant. Some summaries were skipped over for brevity.

Splitting the data and setting seed.

```r
set.seed(1)

indices <-sample(1:nrow(data), 0.7 * nrow(data), replace = TRUE)
training <-data[indices,]
test  <-data[-indices,]
```

Carrying out the feature selection and yielding a final regression model.

```r
# Entire glm fit for numeric data
glm.fit <- glm(AHDBinary  ~ Age+Sex+RestBP+Chol+Fbs+RestECG+MaxHR+ExAng+Oldpeak+Slope+Ca, data = traini
summary(glm.fit)
```

```
##
## Call:
## glm(formula = AHDBinary ~ Age + Sex + RestBP + Chol + Fbs + RestECG +
##     MaxHR + ExAng + Oldpeak + Slope + Ca, family = binomial,
##     data = training)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8883  -0.5778  -0.1832   0.4322   2.6547
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -4.427167   3.212744  -1.378 0.168203
## Age         -0.046653   0.029954  -1.558 0.119350
## Sex          1.770038   0.524263   3.376 0.000735 ***
## RestBP       0.024170   0.014339   1.686 0.091854 .
## Chol         0.012568   0.004439   2.831 0.004637 **
## Fbs         -1.254899   0.670752  -1.871 0.061361 .
## RestECG     -0.077942   0.218381  -0.357 0.721158
## MaxHR       -0.027156   0.012998  -2.089 0.036686 *
## ExAng        1.792944   0.503486   3.561 0.000369 ***
## Oldpeak      0.037720   0.245289   0.154 0.877785
## Slope        1.234978   0.411564   3.001 0.002694 **
## Ca           1.695576   0.362786   4.674 2.96e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 282.89  on 206  degrees of freedom
## Residual deviance: 154.98  on 195  degrees of freedom
## AIC: 178.98
##
## Number of Fisher Scoring iterations: 6
```

```
# Considering non-numeric categorical data
glm.fit.cat <- glm(AHDBinary  ~ChestPain+Thal, data = training, family = binomial)
summary(glm.fit.cat)
```

```
##
## Call:
## glm(formula = AHDBinary ~ ChestPain + Thal, family = binomial,
##     data = training)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1222  -0.5724  -0.1844   0.4715   2.8576
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)           0.8879     0.7777   1.142   0.2536
## ChestPainnonanginal  -1.8308     0.4452  -4.113 3.91e-05 ***
## ChestPainnontypical  -3.8667     0.8128  -4.757 1.96e-06 ***
## ChestPaintypical     -1.0123     0.6070  -1.668   0.0954 .
## Thalnormal           -1.0871     0.8114  -1.340   0.1803
## Thalreversable        1.2529     0.8419   1.488   0.1367
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 282.89  on 206  degrees of freedom
## Residual deviance: 176.17  on 201  degrees of freedom
## AIC: 188.17
##
## Number of Fisher Scoring iterations: 5
```

```
# Altogether
glm.fit <- glm(AHDBinary  ~ Age+Sex+RestBP+Chol+Fbs+RestECG+MaxHR+ExAng+Oldpeak+Slope+Ca+ChestPain+Thal
summary(glm.fit)
```

```
##
## Call:
## glm(formula = AHDBinary ~ Age + Sex + RestBP + Chol + Fbs + RestECG +
##     MaxHR + ExAng + Oldpeak + Slope + Ca + ChestPain + Thal,
##     family = binomial, data = training)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.0063  -0.4291  -0.1109   0.2791   2.5940
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -8.522904   3.965361  -2.149 0.031608 *
## Age                  -0.012372   0.034110  -0.363 0.716829
## Sex                   1.837955   0.666409   2.758 0.005816 **
## RestBP                0.026716   0.015309   1.745 0.080956 .
## Chol                  0.010956   0.004845   2.261 0.023756 *
## Fbs                  -1.015934   0.808116  -1.257 0.208695
## RestECG              -0.119485   0.254429  -0.470 0.638628
## MaxHR                -0.012725   0.015260  -0.834 0.404342
## ExAng                 1.294977   0.600853   2.155 0.031144 *
## Oldpeak              -0.045660   0.303666  -0.150 0.880479
## Slope                 1.370356   0.518486   2.643 0.008218 **
## Ca                    1.700924   0.407042   4.179 2.93e-05 ***
## ChestPainnonanginal -2.045326   0.619620  -3.301 0.000964 ***
## ChestPainnontypical -3.158668   0.935319  -3.377 0.000733 ***
## ChestPaintypical    -1.983980   0.763728  -2.598 0.009383 **
## Thalnormal           1.010428   1.224139   0.825 0.409134
## Thalreversable       2.097811   1.203044   1.744 0.081202 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 282.89  on 206  degrees of freedom
## Residual deviance: 123.51  on 190  degrees of freedom
## AIC: 157.51
```

```
##
## Number of Fisher Scoring iterations: 6
```

```
# Removing highest param (Oldpeak)
glm.fit <- glm(AHDBinary  ~ Age+Sex+RestBP+Chol+Fbs+RestECG+MaxHR+ExAng+Slope+Ca+ChestPain+Thal, data =
summary(glm.fit)
```

```
##
## Call:
## glm(formula = AHDBinary ~ Age + Sex + RestBP + Chol + Fbs + RestECG +
##     MaxHR + ExAng + Slope + Ca + ChestPain + Thal, family = binomial,
##     data = training)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.0063  -0.4277  -0.1120   0.2755   2.5968
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -8.579505   3.950732  -2.172 0.029884 *
## Age                  -0.011101   0.033006  -0.336 0.736619
## Sex                   1.842278   0.666528   2.764 0.005710 **
## RestBP                0.026199   0.014899   1.758 0.078675 .
## Chol                  0.010847   0.004786   2.266 0.023432 *
## Fbs                  -1.005109   0.807074  -1.245 0.212994
## RestECG              -0.112775   0.250381  -0.450 0.652411
## MaxHR                -0.012364   0.015084  -0.820 0.412395
## ExAng                 1.285547   0.597140   2.153 0.031331 *
## Slope                 1.347793   0.496717   2.713 0.006660 **
## Ca                    1.688378   0.398209   4.240 2.24e-05 ***
## ChestPainnonanginal -2.052046   0.619517  -3.312 0.000925 ***
## ChestPainnontypical -3.136389   0.921067  -3.405 0.000661 ***
## ChestPaintypical    -2.012248   0.741042  -2.715 0.006619 **
## Thalnormal            1.040552   1.205186   0.863 0.387920
## Thalreversable        2.118402   1.193274   1.775 0.075851 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 282.89  on 206  degrees of freedom
## Residual deviance: 123.53  on 191  degrees of freedom
## AIC: 155.53
##
## Number of Fisher Scoring iterations: 6
```

```
# Removing highest param (Age?!)
glm.fit <- glm(AHDBinary  ~ Sex+RestBP+Chol+Fbs+RestECG+MaxHR+ExAng+Slope+Ca+ChestPain+Thal, data = tra

# Removing highest param (RestECG)
glm.fit <- glm(AHDBinary  ~ Sex+RestBP+Chol+Fbs+MaxHR+ExAng+Slope+Ca+ChestPain+Thal, data = training, fa

# Removing highest params (MaxHR)
```

```r
glm.fit <- glm(AHDBinary  ~ Sex+RestBP+Chol+Fbs+ExAng+Slope+Ca+ChestPain+Thal, data = training, family =

# Removing rest of nonsignificant params in order (skipping ahead)
glm.fit5 <- glm(AHDBinary  ~ Sex+Chol+ExAng+Slope+Ca+ChestPain+Thal, data = training, family = binomial)
summary(glm.fit5)
```

```
##
## Call:
## glm(formula = AHDBinary ~ Sex + Chol + ExAng + Slope + Ca + ChestPain +
##     Thal, family = binomial, data = training)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8391  -0.4416  -0.1502   0.2739   2.5677
##
## Coefficients:
##                        Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -7.075262   2.146256  -3.297 0.000979 ***
## Sex                   1.560495   0.581312   2.684 0.007265 **
## Chol                  0.009321   0.004352   2.142 0.032189 *
## ExAng                 1.448660   0.550374   2.632 0.008485 **
## Slope                 1.210650   0.447351   2.706 0.006804 **
## Ca                    1.633395   0.369119   4.425 9.64e-06 ***
## ChestPainnonanginal -2.184537   0.594776  -3.673 0.000240 ***
## ChestPainnontypical -3.058687   0.840206  -3.640 0.000272 ***
## ChestPaintypical    -1.884966   0.702410  -2.684 0.007284 **
## Thalnormal            0.964779   1.132919   0.852 0.394443
## Thalreversable        2.236792   1.108621   2.018 0.043629 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 282.89  on 206  degrees of freedom
## Residual deviance: 128.12  on 196  degrees of freedom
## AIC: 150.12
##
## Number of Fisher Scoring iterations: 6
```

Note that the removals were not biased by intuition as to the viability of the feature. That is, they were simply removed depending on their controbution to the model, bot on their intuitive viability to the domain.

## Confusion Table

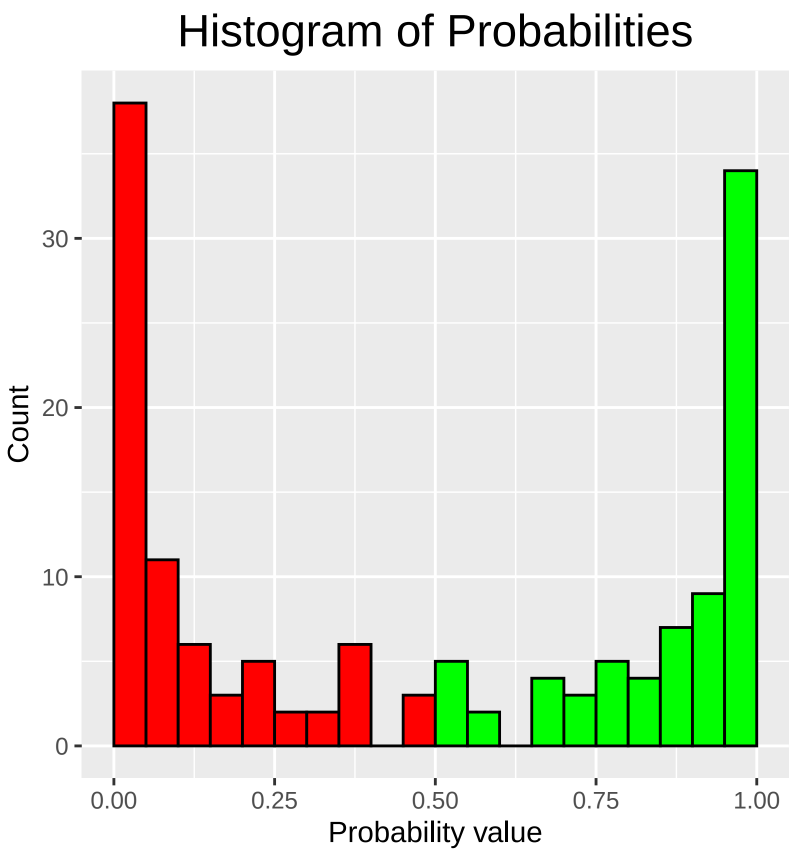
Constructing probability distribution for predictions on each test observation. The following is simply an example of the first ten prediction probabilities. We can observe, for these, that many are very high (~99%).

```r
glm.probs = predict(glm.fit5, test, type = "response")
#The first 10 predicted probabilities
glm.probs[1:10]
```

```
##            2            3            4            5            6            7
## 0.998979144 0.997511014 0.316001177 0.002336079 0.014800821 0.963959548
##           8           9          10          11
## 0.462227736 0.958686279 0.975752401 0.213535076
```

The following is a distribution of these probabilities. Green represents the probabilities that are greater than 50%, and thus will be classified for "Yes". A promising component of our model is that the distribution is skewed towards 1 and 0, rather than being more uniform. This implies there is a decent distinction between yes or no in producing our prediction, in addition to accuracy analysis.

```
#Visualizing our probability distribution
colors <- c(rep("red",10), rep("green",10))
probHist = ggplot(mapping = aes(glm.probs)) + geom_histogram(binwidth=0.05,boundary = 0,color="black", f
probHist = probHist + ggtitle("Histogram of Probabilities") + theme(plot.title = element_text(hjust = 0
probHist
```



Generate confusion table based off of 0.5 prediction cutoff.

```
#Choosing 0.5 as the cutoff for prediction
glm.pred <- ifelse(glm.probs > 0.5,1,0)
#Constructing the table
glm.table = table(glm.pred,test$AHDBinary)
glm.table
```

```
##
## glm.pred  0  1
##        0 65 11
##        1 14 59
```

## Mode-Test Statistics

Calculate classification accuracy and error, sensitivity, specificity, PPV and NPV. We indeed saw more "no" than "Yes" (represented as 0 and 1 respectively). We also seemed to have about equal PPV and NPV.

```
#Accuracy
table.trace = sum(diag(glm.table))
table.sum = sum(glm.table)
acc = table.trace / table.sum
acc
```

```
## [1] 0.8322148
```

```
#0.8754209

#error
err = 1 - acc
err
```

```
## [1] 0.1677852
```

```
#sensitivity
sens = glm.table[1]/(glm.table[1] + glm.table[2])
sens
```

```
## [1] 0.8227848
```

```
#Specificity
spec = glm.table[4]/(glm.table[4] + glm.table[3])
spec
```

```
## [1] 0.8428571
```

```
#PPV - Positive Predictive Value
PPV = glm.table[1]/(glm.table[1] + glm.table[3])
PPV
```

```
## [1] 0.8552632
```

```
#NPV - Negative Predictive Value
PPV = glm.table[4]/(glm.table[4] + glm.table[2])
PPV
```

```
## [1] 0.8082192
```

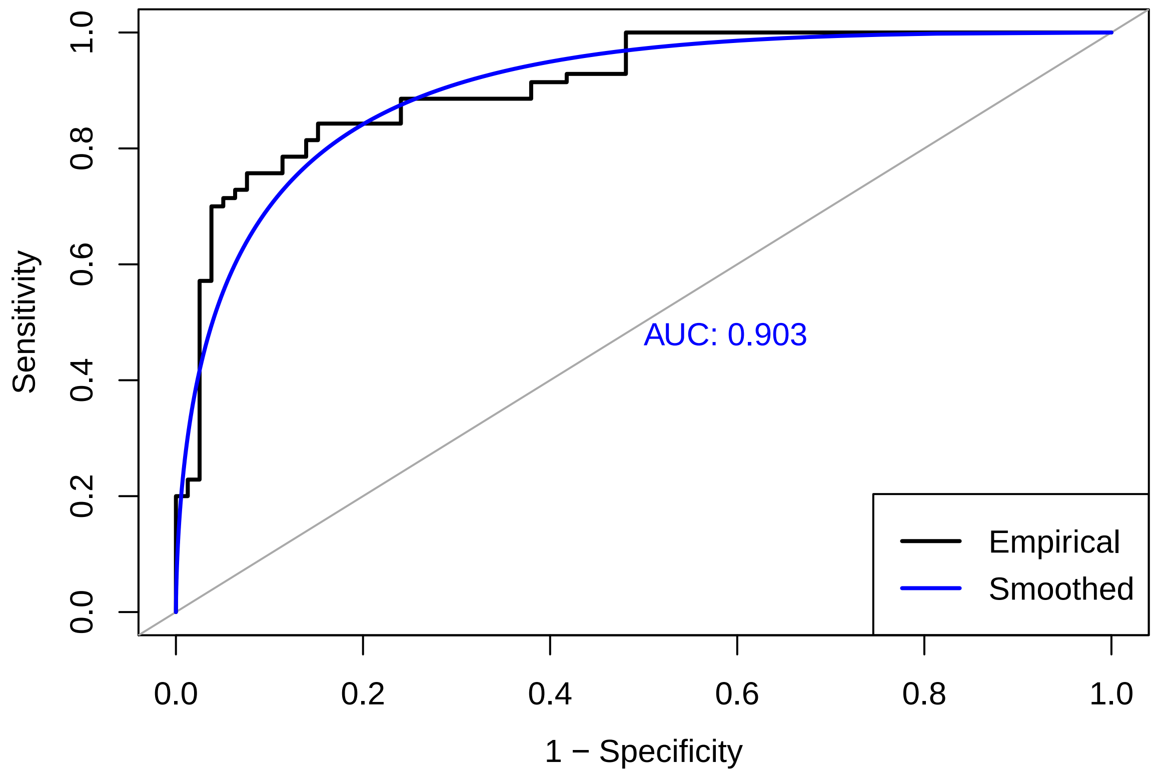
## ROC and AUC

Generate ROC and compute AUC. The AUC is displayed on the figure below, that being 0.903, generated by integrating and similarly calculating the area under the curve. The ROC and AUC gives us more insight as to the performance of our model. An AUC of 0.903 is fairly acceptable performance, and indicates that there is a class separation between yes and no.

```r
test_prob = predict(glm.fit5, newdata = test, type = "response")

test_roc = roc(test$AHDBinary, test_prob)
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```

```r
plot.roc(test_roc, col=par("fg"),print.auc=FALSE,legacy.axes=TRUE,asp =NA)

plot.roc(smooth(test_roc),col="blue",add=TRUE,print.auc=TRUE,legacy.axes = TRUE, asp =NA)
legend("bottomright",legend=c("Empirical","Smoothed"),col=c(par("fg"),"blue"), lwd=2)

abline(v = -coef(glm.fit5)[1] / coef(glm.fit5)[2], lwd = 3)
```



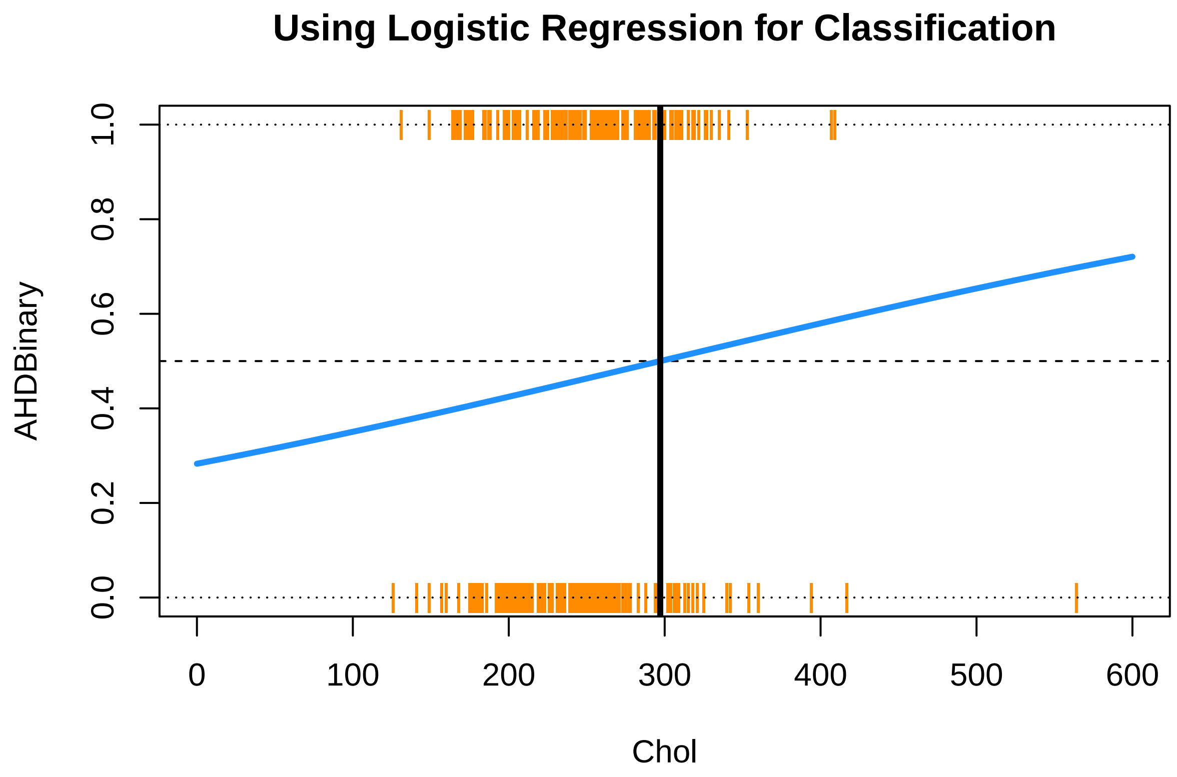
## S sigmoid curve

The following is a s-curve for Chol regarding our logistic model. The orange lines represent the points for Chol with AHDBinary (Chol,AHDBinary). The blue "s-curve" represent the predicted probabilities given the fitted logistic regression. The black line represents the 0.5 cutoff, and decision boundary.

In our case, Chol did not seem to fit our s-curve very well. Firstly, our so called s curve doesn't immediately seem to resemble a s-curve. As such, our points are not well fitted to the curve. Furthermore, 0 and 1 points lie on both sides of the dicision boundary. Chol was one of the paramaters that we included which happened to be significant within our model and random train/test split. Looking at the orange locations more closely, we can see some high cholesterol values, yet belong in the “no” category, which are interesting points that pull down on that end of the s curve.

This runs counter to what we may expect. It could be expected that the “yes” category would be full of higher cholesterol, yet the “No” category to have lower cholesteral values in contrast.

```
single.glm <- glm(AHDBinary ~ Chol, data = data, family = "binomial")


plot(AHDBinary ~ Chol, data = data,col = "darkorange", pch = "|", xlim = c(0, 600), ylim = c(0, 1),main
abline(h = 0, lty = 3)
abline(h = 1, lty = 3)
abline(h = 0.5, lty = 2)

curve(predict(single.glm, data.frame(Chol = x), type = "response"),add = TRUE, lwd = 3, col = "dodgerblu
abline(v = -coef(single.glm)[1] / coef(single.glm)[2], lwd = 3)
```



## CART Model

A Classification and Regression Tree (CART) can be used as a predictive model that explains how a combination of variables can work together, and output a singular result. Specifically, in our Heart Disease Dataset, we are able to determine how variables interact with one another, to determine the binary outcome of having Heart Disease or not. In order to do this, we first obtain our Decision Tree, a series of if/else statements that will lead to a "yes" or "no" prediction. This CART Model serves as a more probabilistic scenario that occurs when dealing with a patient's medical history. By analyzing factors such as Heart Rate, Blood Pressure, and age, doctor's can better assess the health of a patient, and make a determination into the likelihood of them having Heart Disease.

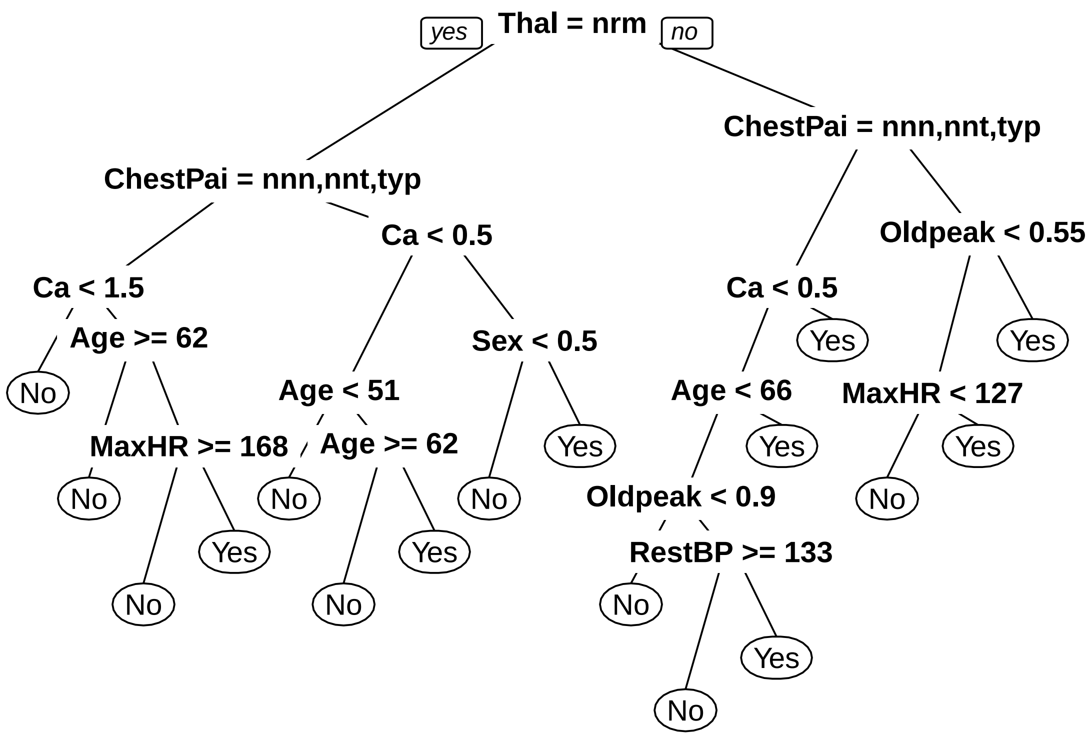
Due to the nature of our dataset, we will specifically model Classification Trees, as we are dealing with a binary outcome of "Yes, the patient has AHD", or "No, the patient does not have AHD". Although initially similar to Regression Trees, we will begin with a recursive binary splitting to grow our tree, but do not need to rely on repeating to minimize the SSR/SSE/RSS. Rather, we are more focused on our Classification Error Rate and minimizing our Gini index. Although the CART model may not have the same level of predictive accuracy as some of the other tested approaches in this report, it is the most similar to mirror human decision-making. Thus, it is a strong predictive model for our specific dataset on Heart Disease.

## CART MODEL

```
Heart <-read.csv('https://www.statlearning.com/s/Heart.csv')
set.seed(1)
heart.split <- sample(1:nrow(Heart), size=nrow(Heart) * 0.7)
heart.train <- Heart[heart.split,]
heart.test <- Heart[-heart.split,]
```

Constructing our splits.

```
class.cart <- rpart(formula = AHD ~ Age+Sex+RestBP+Chol+Fbs+RestECG+MaxHR+ExAng+Oldpeak+Slope+Ca+ChestPa
prp(class.cart, roundint = FALSE)
```

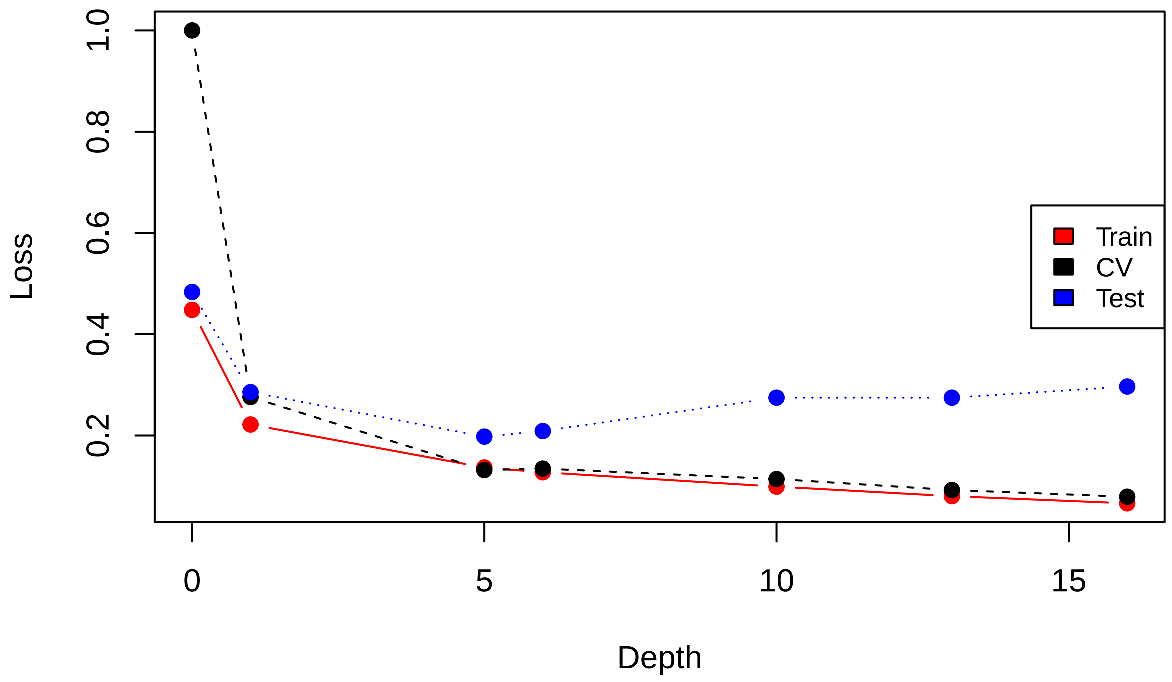


Based on the results of the above Decision Tree, Thalassemia and Chest Pain was determined as the leading predictor of Heart Disease, which was also seen, through the above logistic model. Other interesting patterns to note, would be the classification of age between 51-62, for individuals with Normal Thalassemia and no Chest Pain. 62 became a cutoff point for patients as an indication that their Chest Pain and Thalassemia would not be a factor of Heart Disease, while Chest Pain, without Thalassemia was mostly present in those over the age of 66.

```
cp.class.param <- class.cart$cptable
train.acc <- double(6)
cv.acc <- double(6)
test.acc <- double(6)
for (i in 1:nrow(cp.class.param)) {
  alpha <- cp.class.param[i, 'CP']
  train.cm <- table(heart.train$AHD, predict(prune(class.cart, cp=alpha), newdata = heart.train, type='c
  train.acc[i] <- 1-sum(diag(train.cm))/sum(train.cm)
  cv.acc[i] <- cp.class.param[i, 'xerror'] * cp.class.param[i, 'rel error']
  test.cm <- table(heart.test$AHD, predict(prune(class.cart, cp=alpha), newdata = heart.test, type='clas
  test.acc[i] <- 1-sum(diag(test.cm))/sum(test.cm)
}
```

Train cv and test accuracy.

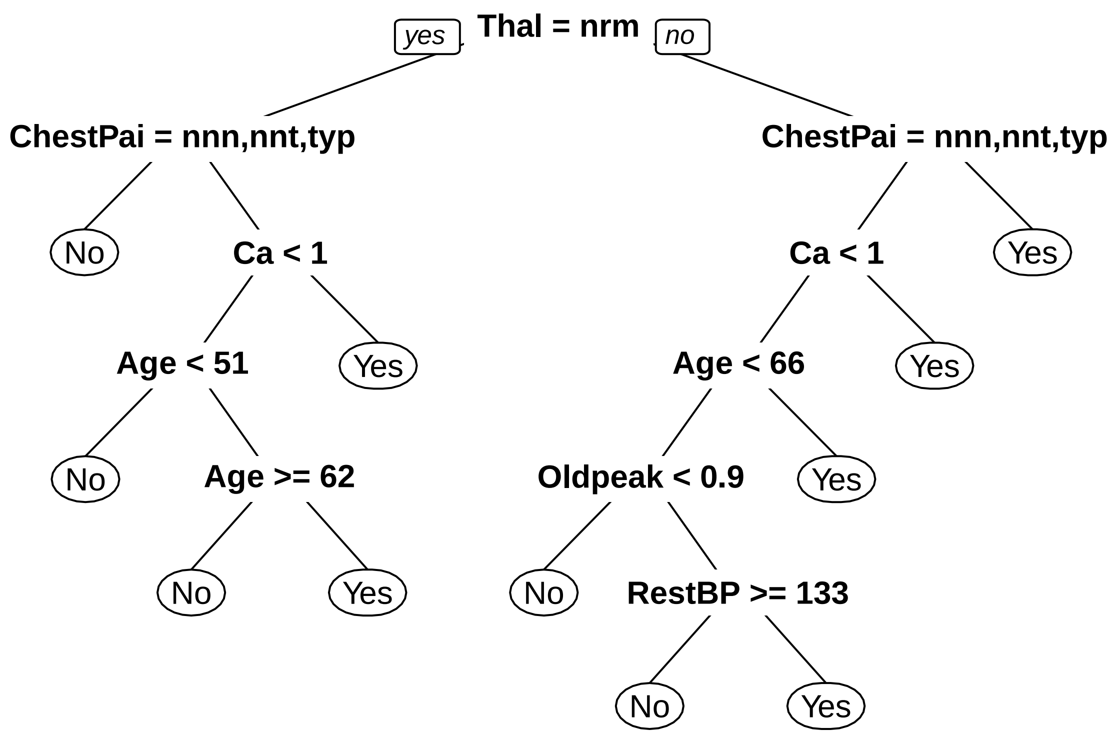
```
matplot(cp.class.param[,'nsplit'], cbind(train.acc, cv.acc, test.acc), pch=19, col=c("red", "black", "bl
legend("right", c('Train', 'CV', 'Test') ,col=seq_len(3),cex=0.8,fill=c("red", "black", "blue"))
```



Looking at the plot of Train, Test, and CV Values, a tree of size 5 seems to be most accurate.

Pruning.

```
prune.class.trees <- prune(class.cart, cp=cp.class.param[5,'CP'])
prp(prune.class.trees)
```



## Confusion Matrix

Confusion Matrix

```
conf.mat.tree <- table(heart.test$AHD, predict(prune.class.trees, type = 'class', newdata = heart.test)
conf.mat.tree
```

```
##
##       No Yes
##   No  30  17
##   Yes  8  36
```

## Mode-Test Statistics

Statistics regarding our confusion table.

```
acc <- sum(diag(conf.mat.tree))/sum(conf.mat.tree)
err <- 1 - acc
sens <- conf.mat.tree[1,1]/(conf.mat.tree[1,1] + conf.mat.tree[2,1])
spec <- conf.mat.tree[2,2]/(conf.mat.tree[2,2] + conf.mat.tree[1,2])
ppv <- conf.mat.tree[1,1]/(conf.mat.tree[1,1] + conf.mat.tree[1,2])
npv <- conf.mat.tree[2,2]/(conf.mat.tree[2,2] + conf.mat.tree[2,1])
c(Accuracy = acc, Error = err, Sensitivity=sens, Specificity = spec, PPV = ppv, NPV = npv)
```

```
##    Accuracy       Error Sensitivity Specificity         PPV         NPV
##   0.7252747   0.2747253   0.7894737   0.6792453   0.6382979   0.8181818
```

## ROC

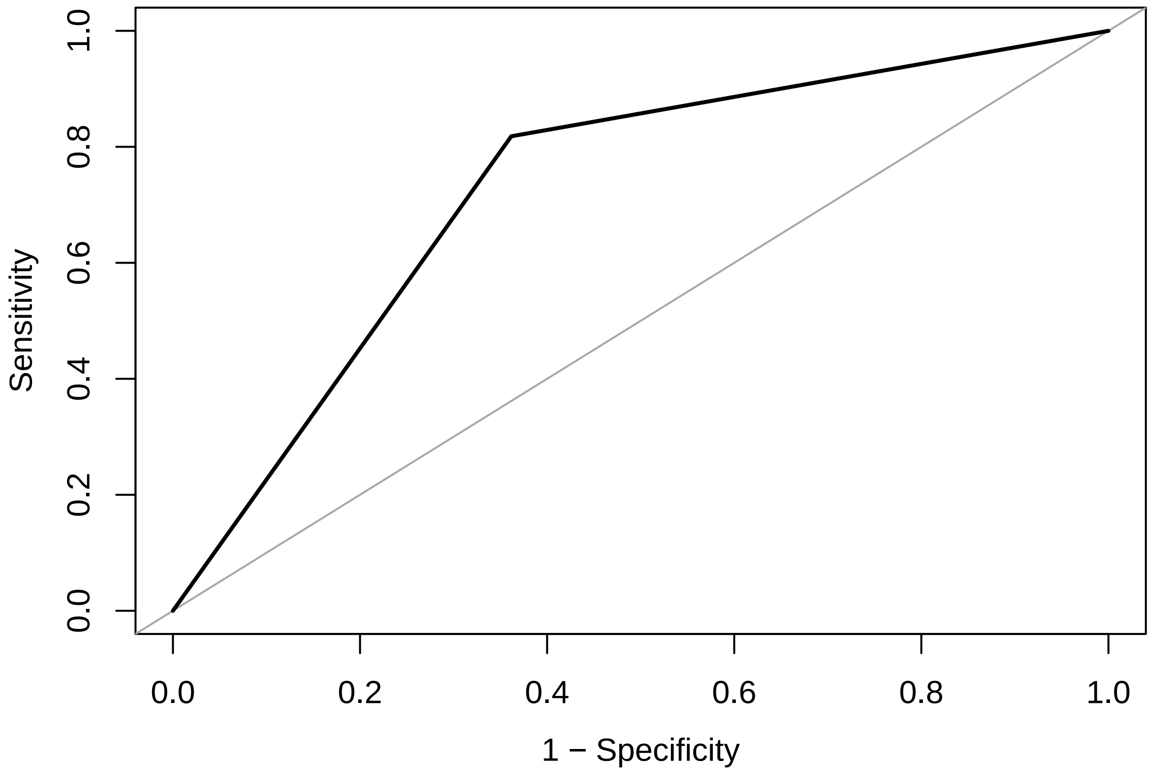
```
library(rpart)
library(ROCR)

test_prob = predict(prune.class.trees, newdata = heart.test, type='class')
test_roc = roc(heart.test$AHD, factor(test_prob, ordered = TRUE))
```

```
## Setting levels: control = No, case = Yes
```

```
## Warning in value[[3L]](cond): Ordered predictor converted to numeric vector.
## Threshold values will not correspond to values in predictor.
```

```
## Setting direction: controls < cases
```

```
plot.roc(test_roc, col=par("fg"),plot=TRUE,print.auc = FALSE, legacy.axes = TRUE, asp =NA)
```



```
#pred <- prediction(predict(class.cart, type="prob")[, 2], Heart$AHD)
#plot(performance(pred, "tpr", "fpr"), col="blue", main="ROC AHD")
#abline(0, 1, lty=2)
```

## Random Forests

The decision trees approach suffers from high variance, meaning the results of the tree fitting to the raining set can be quite different depending on the training/test set split. To combat this issue with the decision trees bootstrap aggregation is employed, which is also referred to as “bagging”. In this approach many training sets are derived from the population using bootstrap, a separate prediction model using each training set is developed, and the resulting predictions are averaged. For the classification task that we consider in this project instead of averaging the majority vote is taken across the predicted classes. This allows to reduce the variance of the statistical method.

On average, each bagged tree makes use of around two-thirds of the observations. The remaining one-third of the observations not used to fit a given bagged tree are referred to as the out-of-bag (OOB) observations. Testing the predictions on the OOB observations is the foundation for the OOB error estimate. Based on the results shown below approximately 230 trees are sufficient for both errors to stabilize.

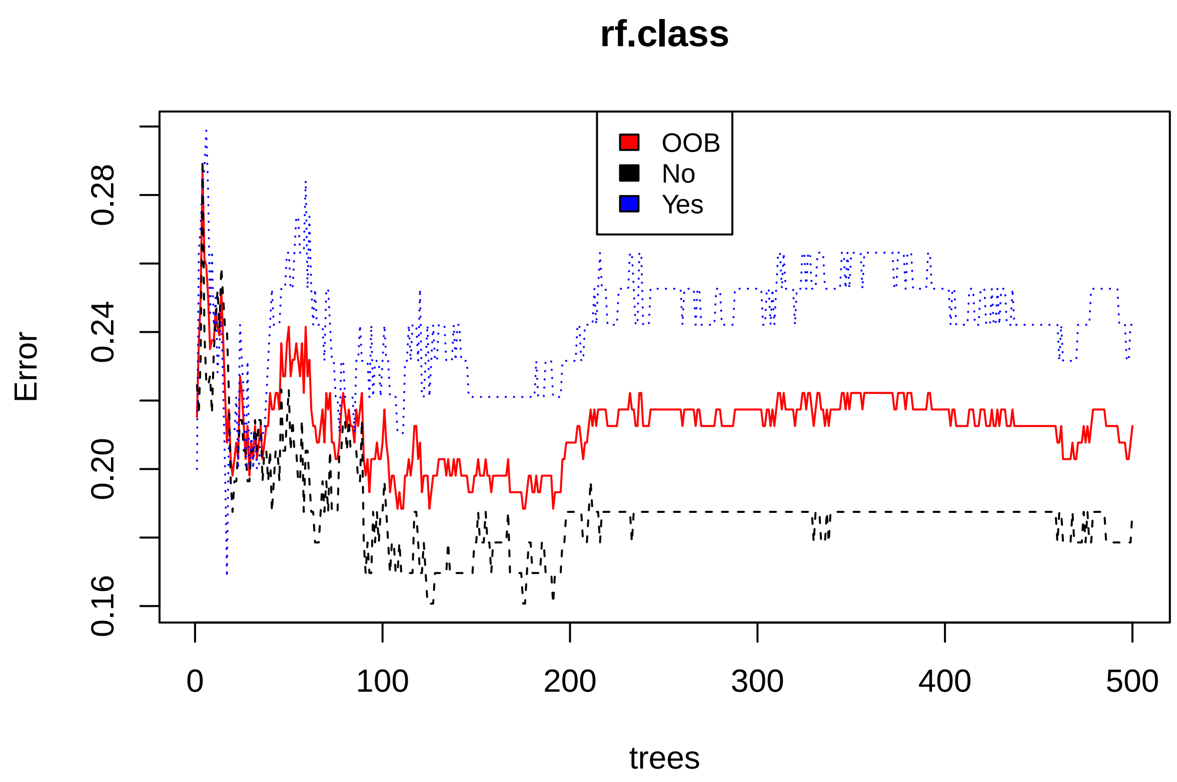
Random forests provide an improvement over bagged trees by way of a random tweak that decorrelates the trees. The split is allowed to use only one a subset of predictors. A fresh sample of predictors is taken at each split, and the number of predictors in the subset typically equals the square root of the total number of predictors. The random forest results and a variable importance plot for the Heart data are given below.

```
Heart <-read.csv('Heart.csv')
Heart <- na.omit(Heart) #Remove NA for demo
data1 <- Heart[,-1]
set.seed(490)
split <- sample(1:nrow(data1), size=nrow(data1) * 0.7)
train <- data1[split,]
test <-  data1[-split,]
train$AHD = factor(train$AHD)
```

```
library(randomForest)
```

```
## randomForest 4.6-14

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

##
## Attaching package: 'randomForest'

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

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
set.seed(1)
rf.class <- randomForest(AHD~., data=train, mtry=round(sqrt(ncol(train)-1)), importance=TRUE, xtest=tes
plot(rf.class, col=c("red", "black", "blue"))
legend("top", colnames(rf.class$err.rate) ,col=seq_len(3),cex=0.8,
       fill=c("red", "black", "blue"))
```



## Confusion Table

Carry out prediction with the Random Forest model:

```
rf_classifier <- randomForest(AHD ~ ., data=train, ntree = 500, mtry=round(sqrt(ncol(train)-1)), importa
prediction_for_table <- predict(rf_classifier, test[,-14])
c.table <- table(observed=test[,14],predicted=prediction_for_table)
c.table
```

```
##         predicted
## observed No Yes
##      No  43   5
##      Yes  9  33
```

## Mode-Test Statistics

Calculate classification accuracy and error, sensitivity, specificity, PPV and NPV.

```
#Accuracy
table.trace = sum(diag(c.table))
table.sum = sum(c.table)
acc = table.trace / table.sum
acc
```

```
## [1] 0.8444444
```

```
#0.8754209

#error
err = 1 - acc
err
```

```
## [1] 0.1555556
```

```
#sensitivity
sens = c.table[1]/(c.table[1] + c.table[2])
sens
```

```
## [1] 0.8269231
```

```
#Specificity
spec = c.table[4]/(c.table[4] + c.table[3])
spec
```

```
## [1] 0.8684211
```

```
#PPV - Positive Predictive Value
PPV = c.table[1]/(c.table[1] + c.table[3])
PPV
```

```
## [1] 0.8958333
```

```
#NPV - Negative Predictive Value
PPV = c.table[4]/(c.table[4] + c.table[2])
PPV
```

```
## [1] 0.7857143
```

Thus, the Random Forest classifier achieved the accuracy of ~ 84 % and corresponding error of ~ 16 %.

## ROC and AUC

Generate ROC and compute AUC for Random Forest

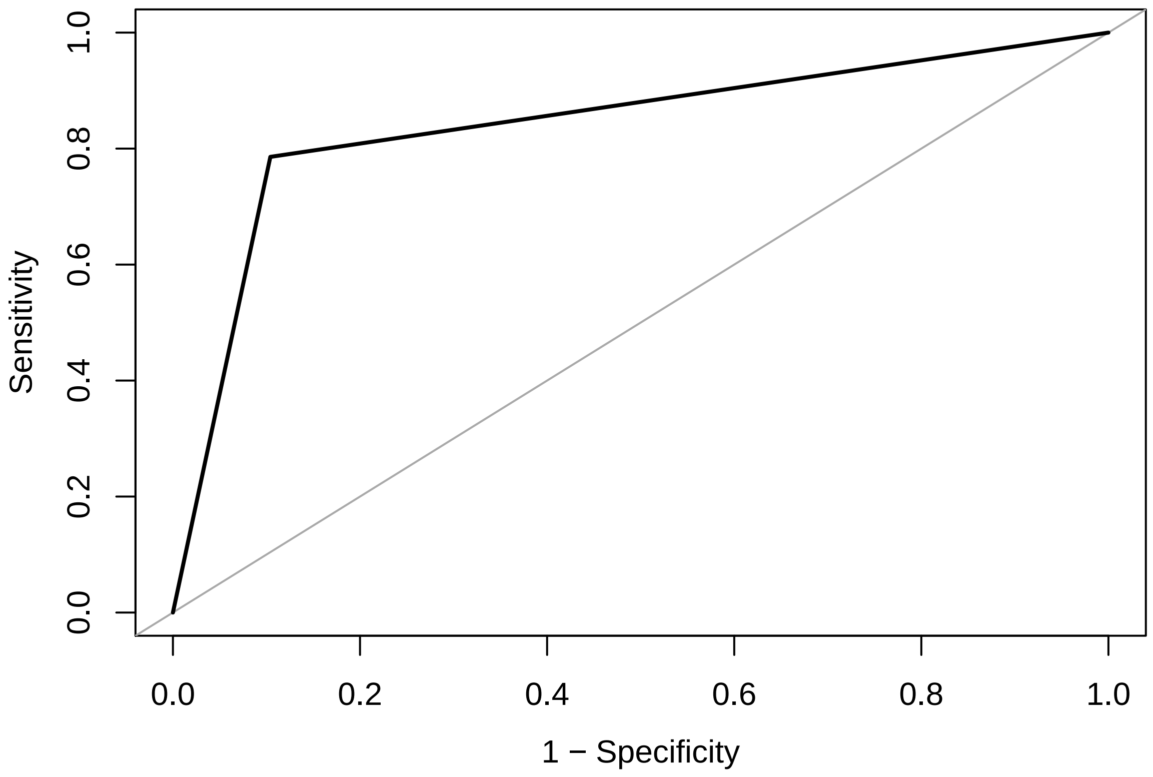
```
test_prob = predict(rf_classifier, newdata = test[,-14], type = "class")
#test_prob = predict(prune.class.trees, newdata = heart.test, type='class')
test_roc = roc(test$AHD, factor(test_prob, ordered = TRUE))
```

```
## Setting levels: control = No, case = Yes
```

```
## Warning in value[[3L]](cond): Ordered predictor converted to numeric vector.
## Threshold values will not correspond to values in predictor.
```

```
## Setting direction: controls < cases
```

```
plot.roc(test_roc, col=par("fg"),plot=TRUE,print.auc = FALSE, legacy.axes = TRUE, asp =NA)
```



## Comparing Models

| | Accuracy | sens | spec | PPV | NPV |
|---|---|---|---|---|---|
| Logistic | 0.832 | 0.822 | 0.842 | 0.855 | 0.808 |
| CART | 0.725 | 0.789 | 0.679 | 0.638 | 0.818 |
| Rand Forest | 0.844 | 0.826 | 0.868 | 0.895 | 0.785 |

The three diferent models produces different cccuracy. The Logistic model had a classification accuracy on its test set of approximately 83 percent. This was higher than the alternative CART model of roughly 72 percent

## Citations

[1] Index of An Introduction to Statistical Learning/Heart.csv, https://www.statlearning.com/s/Heart.csv.

[2] Fang, Julia. "CIS490_LS9_21S_Classification_Logistic&ROC&AUC." MyCourses, 2021.

[3] Fang, Julia. "CIS490_LS10_21S_CART." MyCourses, 2021.

[4] Fang, Julia. “R_logistic&ROC_21S.” MyCourses, 2021.

[5] Fang, Julia. “R_Trees_S21.” MyCourses, 2021.

[6] Fang, Julia. “Supplement_Reading_BaggingRandomForestBoosting.” MyCourses, 2021

[7] James, G., Witten, D., Hastie, T. and Tibshirani, R., 2013. An introduction to statistical learning (Vol. 112, p. 18). New York: springer.