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
```

Describing the data, and (1) identify Y and X.

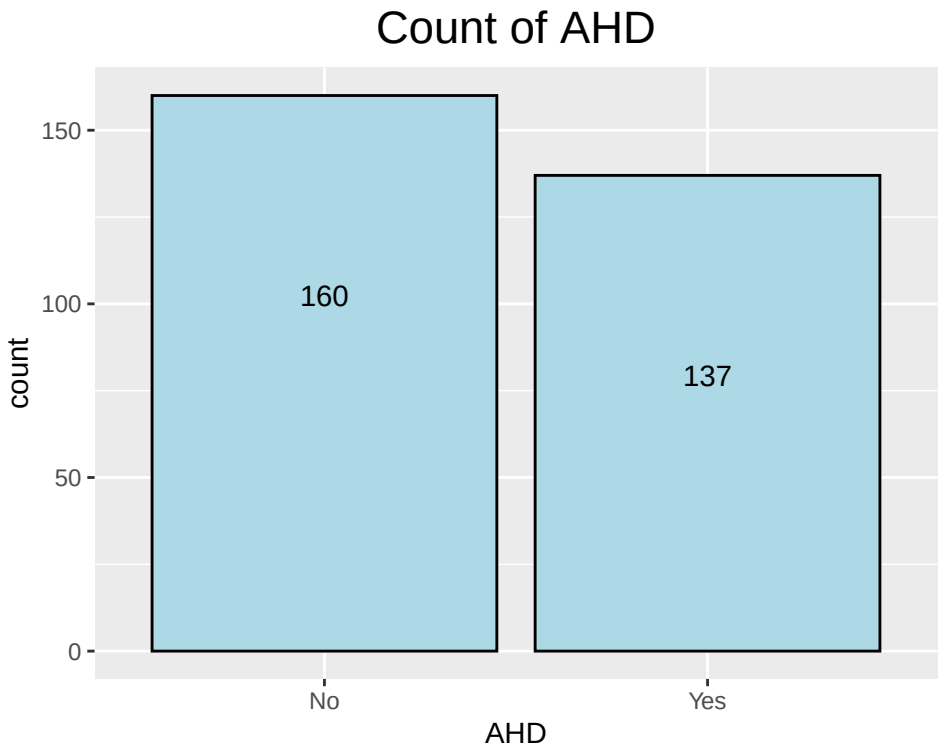
The Heart dataset considers AHD, their binary relationship of having heart disease or not, and their other associated properties. The dataset contains 303 observations and 14 attributes.

More to be added/edited

Exploratory Data Analysis

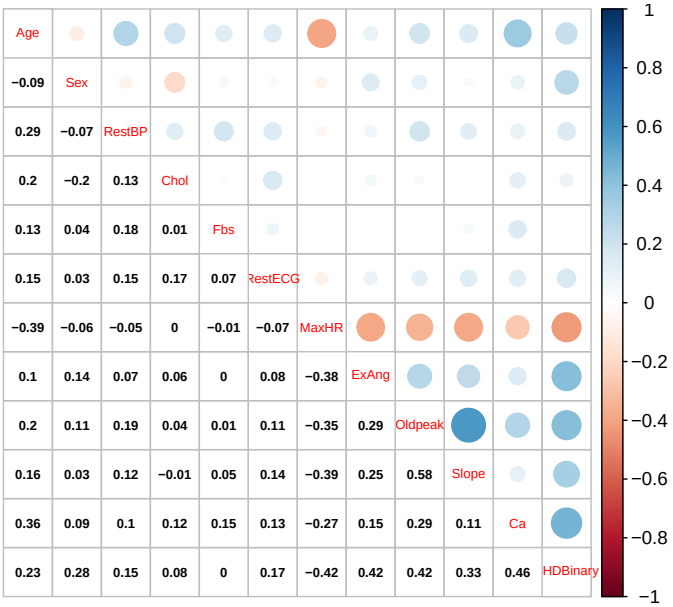
Count of Binary Outcome

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



Correlation matrix

No correlation among AHD Binary exceeds 0.5, naturally.



Logistic Model

Logistic Model

Constructing a logistic model for AHD based on other features. Tested on a random roughly 40 percent of the data.

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

```
##
## Call:
## glm(formula = AHDBinary ~ Age + Sex + RestBP + Chol + Fbs + RestECG +
##     MaxHR + ExAng + Oldpeak + Slope + Ca, family = binomial,
##     data = data)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.645  -0.571  -0.219   0.515   2.528
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.450356   2.485081  -1.388  0.16501
## Age         -0.017900   0.022323  -0.802  0.42264
## Sex          1.770589   0.416973   4.246 2.17e-05 ***
## RestBP       0.022437   0.010113   2.219  0.02651 *
## Chol         0.006737   0.003642   1.850  0.06436 .
## Fbs         -1.114571   0.504888  -2.208  0.02727 *
## RestECG      0.152906   0.171295   0.893  0.37205
## MaxHR       -0.027921   0.009457  -2.952  0.00315 **
## ExAng        1.457947   0.370803   3.932 8.43e-05 ***
## Oldpeak      0.247482   0.193969   1.276  0.20200
## Slope        0.661351   0.340964   1.940  0.05242 .
## Ca           1.364116   0.248518   5.489 4.04e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 409.95  on 296  degrees of freedom
## Residual deviance: 230.75  on 285  degrees of freedom
## AIC: 254.75
##
## Number of Fisher Scoring iterations: 5
```

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

```
##
## Call:
## glm(formula = AHDBinary ~ ChestPain + Thal, family = binomial,
##     data = data)
##
```

```
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0869  -0.4757  -0.4697   0.4904   2.1256
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)           1.4535     0.5716   2.543  0.01099 *
## ChestPainnonanginal  -2.0354     0.3648  -5.579 2.42e-08 ***
## ChestPainnontypical  -2.0084     0.4533  -4.430 9.42e-06 ***
## ChestPaintypical     -1.8031     0.5513  -3.271  0.00107 **
## Thalnormal           -1.5670     0.5934  -2.641  0.00828 **
## Thalreversable        0.6038     0.6079   0.993  0.32056
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 409.95  on 296  degrees of freedom
## Residual deviance: 275.86  on 291  degrees of freedom
## AIC: 287.86
##
## Number of Fisher Scoring iterations: 4
```

```
# Combining categorical with numeric; Removing Age
glm.fit2 <- glm(AHDBinary  ~ Sex+RestBP+Chol+Fbs+RestECG+MaxHR+ExAng+Oldpeak+Slope+Ca+ChestPain+Thal, da
# Removing RestECG
glm.fit3 <- glm(AHDBinary  ~ Sex+RestBP+Chol+Fbs+MaxHR+ExAng+Oldpeak+Slope+Ca+ChestPain+Thal, data = da

# We may consider removing more, but let's keep these for now and observe
glm.fit5 <- glm(AHDBinary  ~Sex+RestBP+MaxHR+ExAng+Oldpeak+Slope+Ca+ChestPain+Thal, data = data, family
summary(glm.fit5)
```

```
##
## Call:
## glm(formula = AHDBinary ~ Sex + RestBP + MaxHR + ExAng + Oldpeak +
##     Slope + Ca + ChestPain + Thal, family = binomial, data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7254  -0.5170  -0.1672   0.3531   2.7048
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -3.436593   2.231691  -1.540  0.12358
## Sex                  1.302571   0.477071   2.730  0.00633 **
## RestBP               0.022548   0.010254   2.199  0.02789 *
## MaxHR               -0.018967   0.009745  -1.946  0.05161 .
## ExAng                0.718634   0.426274   1.686  0.09182 .
## Oldpeak              0.378583   0.221461   1.709  0.08736 .
## Slope                0.660478   0.357868   1.846  0.06495 .
## Ca                   1.195328   0.248794   4.804 1.55e-06 ***
## ChestPainnonanginal -1.923076   0.476883  -4.033 5.52e-05 ***
## ChestPainnontypical -0.983244   0.547575  -1.796  0.07255 .
## ChestPaintypical    -2.090844   0.640439  -3.265  0.00110 **
```

```
## Thalnormal           0.212638   0.758579   0.280  0.77924
## Thalreversable       1.618077   0.746377   2.168  0.03017 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 409.95  on 296  degrees of freedom
## Residual deviance: 199.87  on 284  degrees of freedom
## AIC: 225.87
##
## Number of Fisher Scoring iterations: 6
```

Confusion Table

Constructing a test subset.

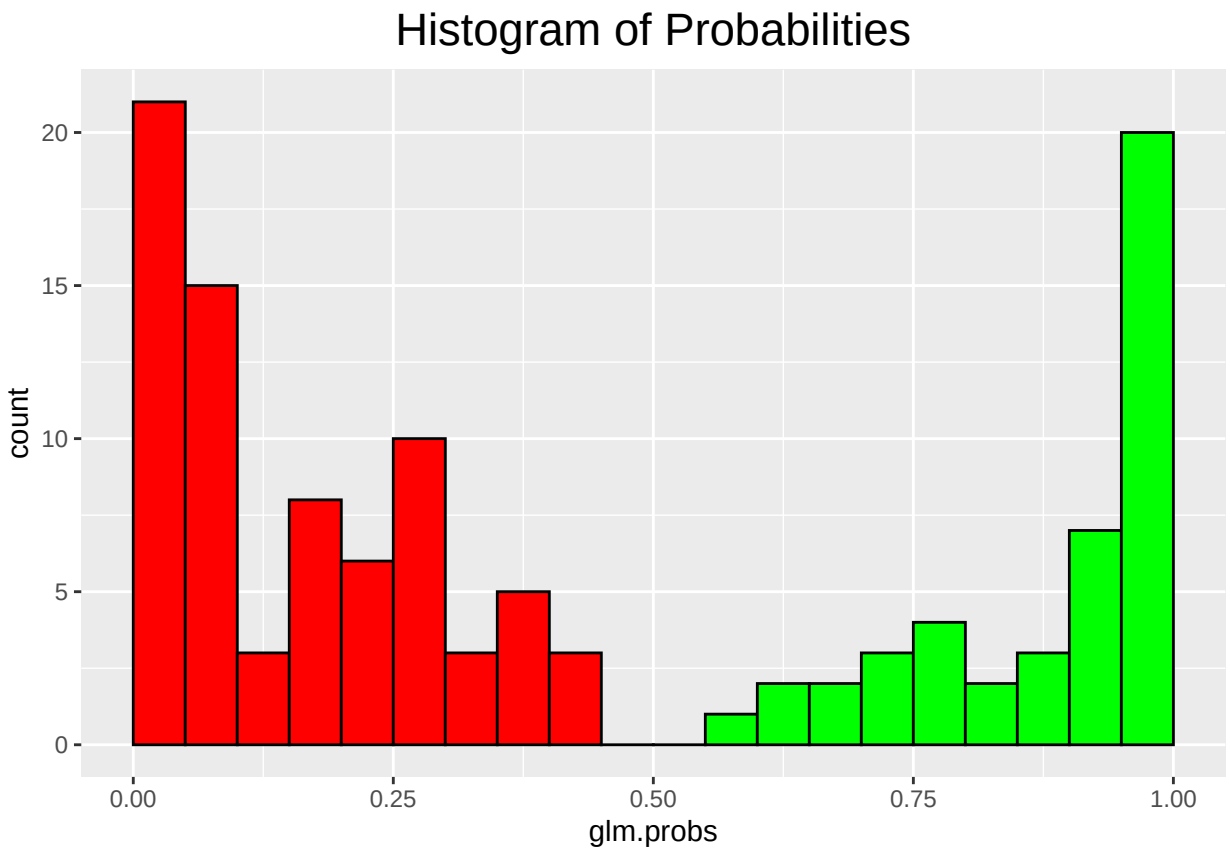
```
set.seed(1)

testing.indices = sort(sample(nrow(data), nrow(data)*.4, replace = TRUE))
test <- data[testing.indices,]
```

Constructing probability distribution for predictions on each test observation.

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

```
##          1         13         14         19         20       20.1         22
## 0.27921775 0.41164694 0.19555709 0.01604686 0.08873524 0.08873524 0.01859199
##         24         25         29
## 0.82032665 0.99478215 0.36487778
```

```
#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",
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 62 12
##        1  3 41
```

Mode-Test Statistics

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

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

```
## [1] 0.8728814
```

```
#0.8754209

#error
err = 1 - acc
err
```

```
## [1] 0.1271186
```

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

```
## [1] 0.9538462
```

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

```
## [1] 0.7735849
```

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

```
## [1] 0.8378378
```

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

```
## [1] 0.9318182
```

ROC and AUC

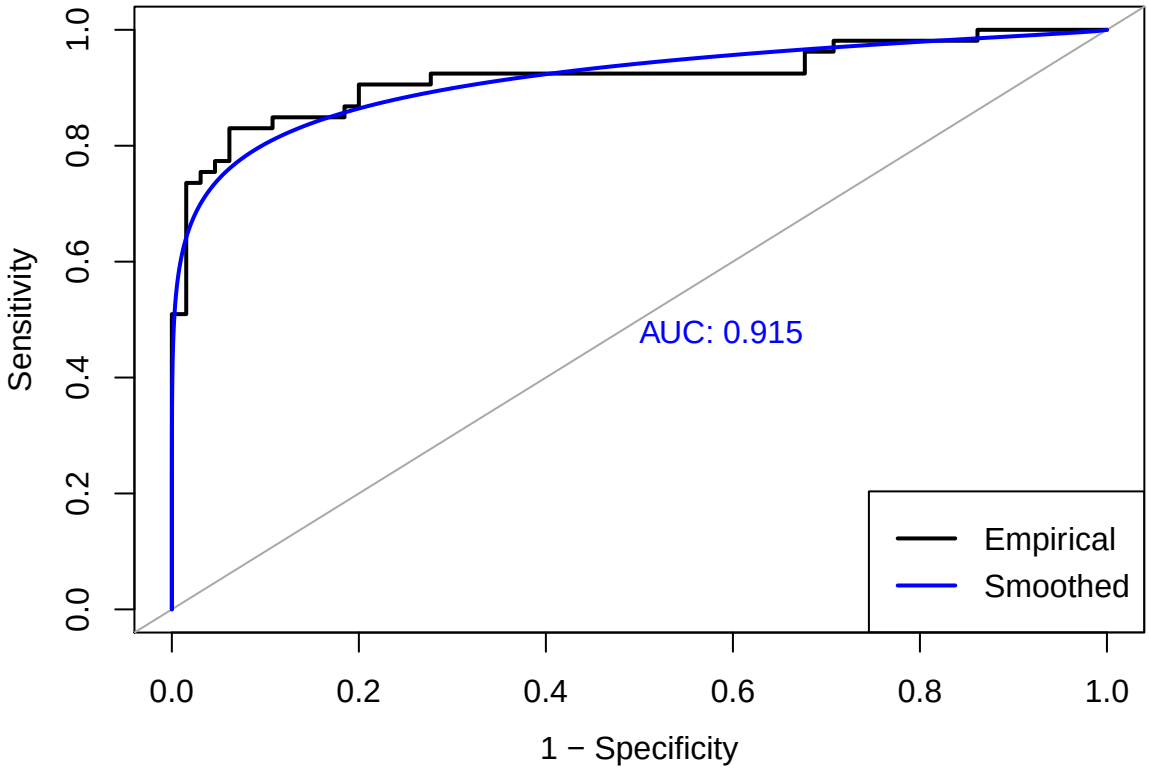
Generate ROC and compute AUC for each model

```
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
```

```
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)
```

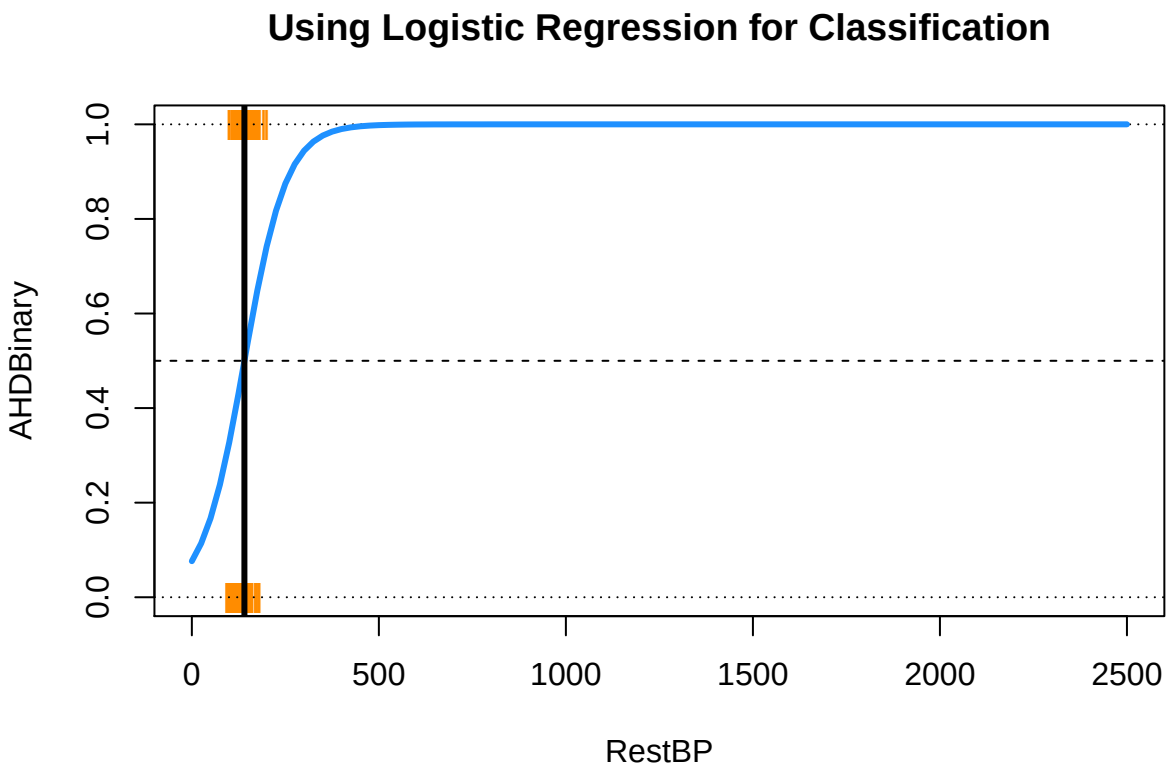


S sigmoid curve

Generate s-curve for Y against one attribute (you can pick any one attribute), and interpret your findings

I used RestBP TEMPORARILY. THis will be replaced with something more appropriate when we try our CART model.

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

plot(AHDBinary ~ RestBP, data = data,col = "darkorange", pch = "|", xlim = c(0, 2500), ylim = c(0, 1),ma
abline(h = 0, lty = 3)
abline(h = 1, lty = 3)
abline(h = 0.5, lty = 2)

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



CART Model

CART MODEL

Other subsections

Other Models (Bagging, Random Forests, Boosting)

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

```
library(randomForest)
```

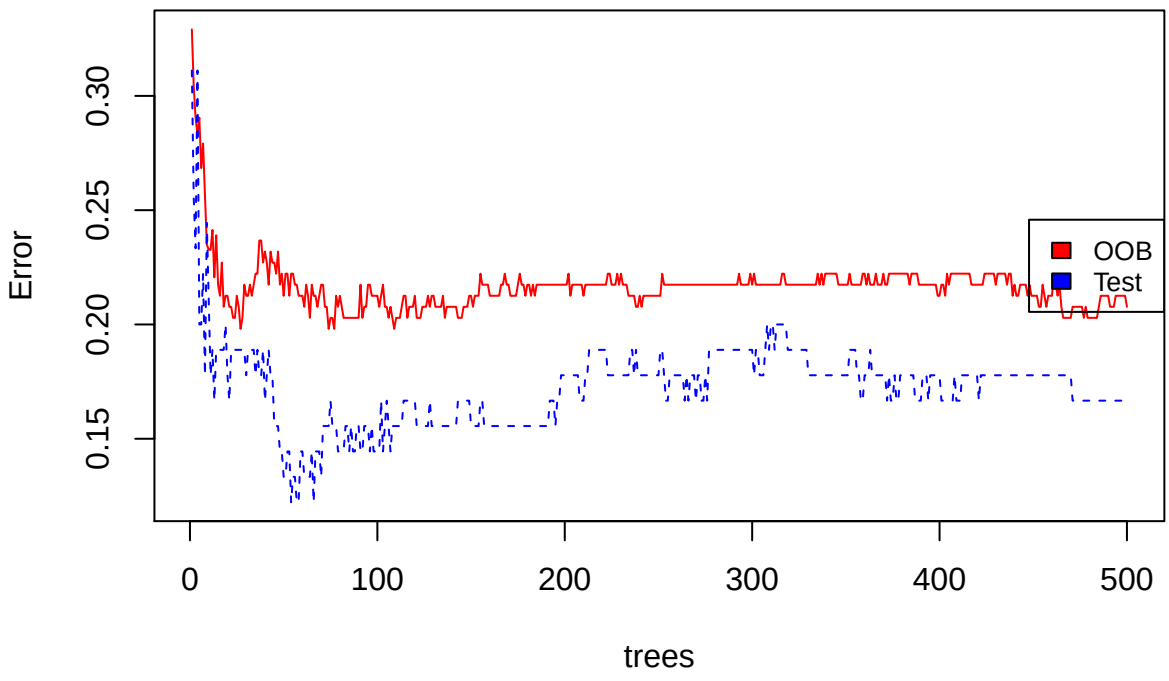
```
## Warning: package 'randomForest' was built under R version 4.0.4
```

```
## 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
```

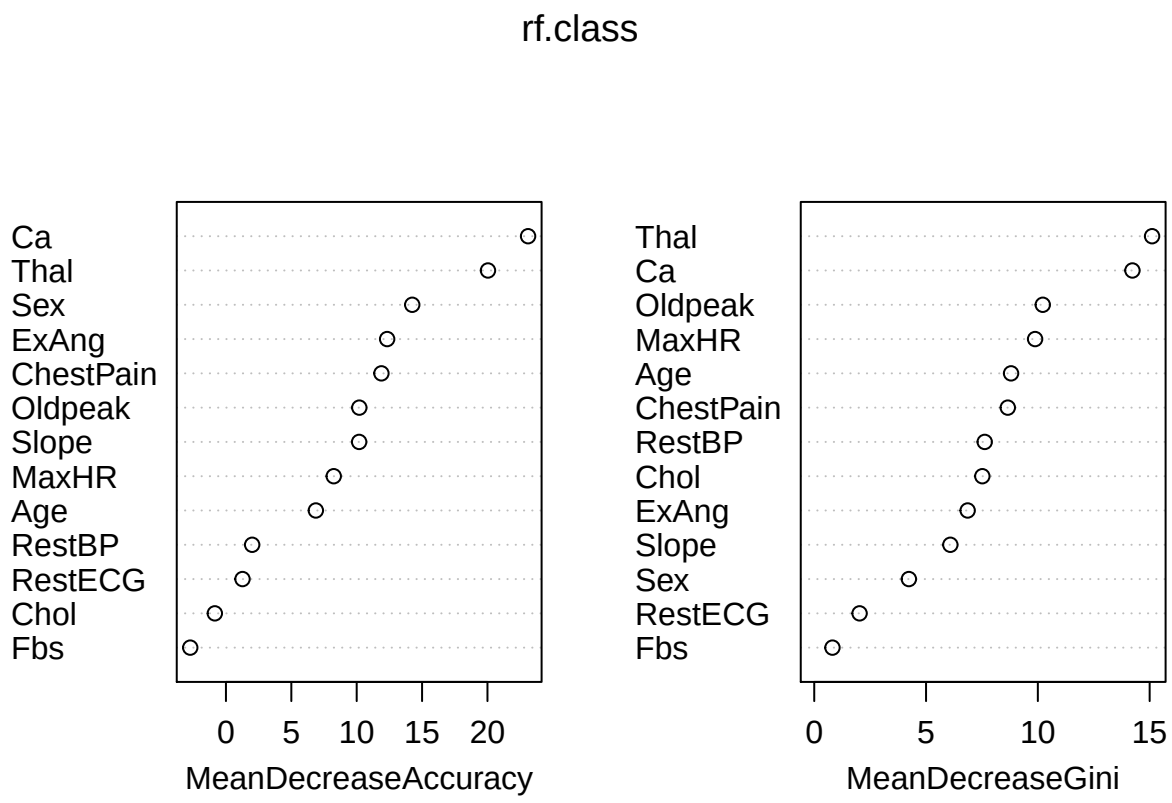
Bagged Trees

```
train$AHD = factor(train$AHD)

bag.class <- randomForest(AHD~., data = train, mtry=ncol(train) - 1, importance=TRUE,xtest=test[,-14],y

err <- bag.class$err.rate[,1]
bag.err <- cbind(err, bag.class$test$err.rate[,1])
colnames(bag.err) <- c("OOB", "Test")
matplot(1:bag.class$ntree, bag.err, type = "l", xlab="trees", ylab="Error", col = c("red", "blue"))
legend("right", c('OOB', 'Test') ,col=seq_len(2),cex=0.8,fill=c("red", "blue"))
```

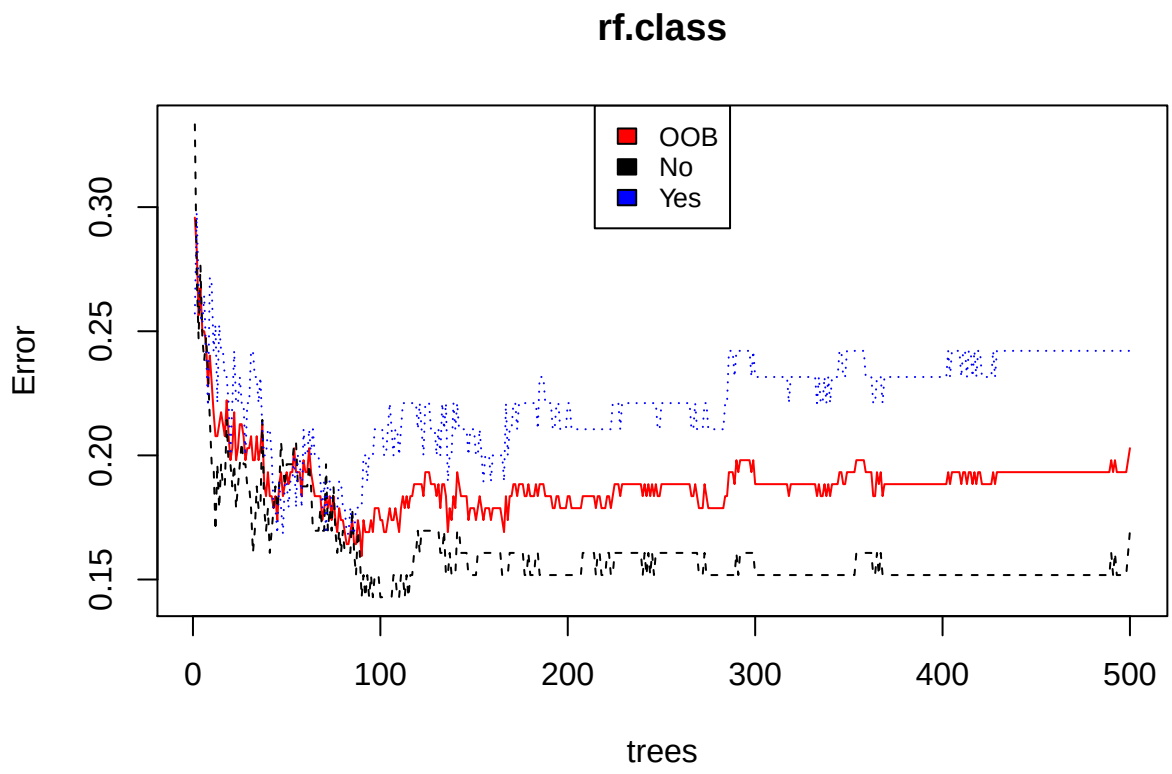


Random Forests

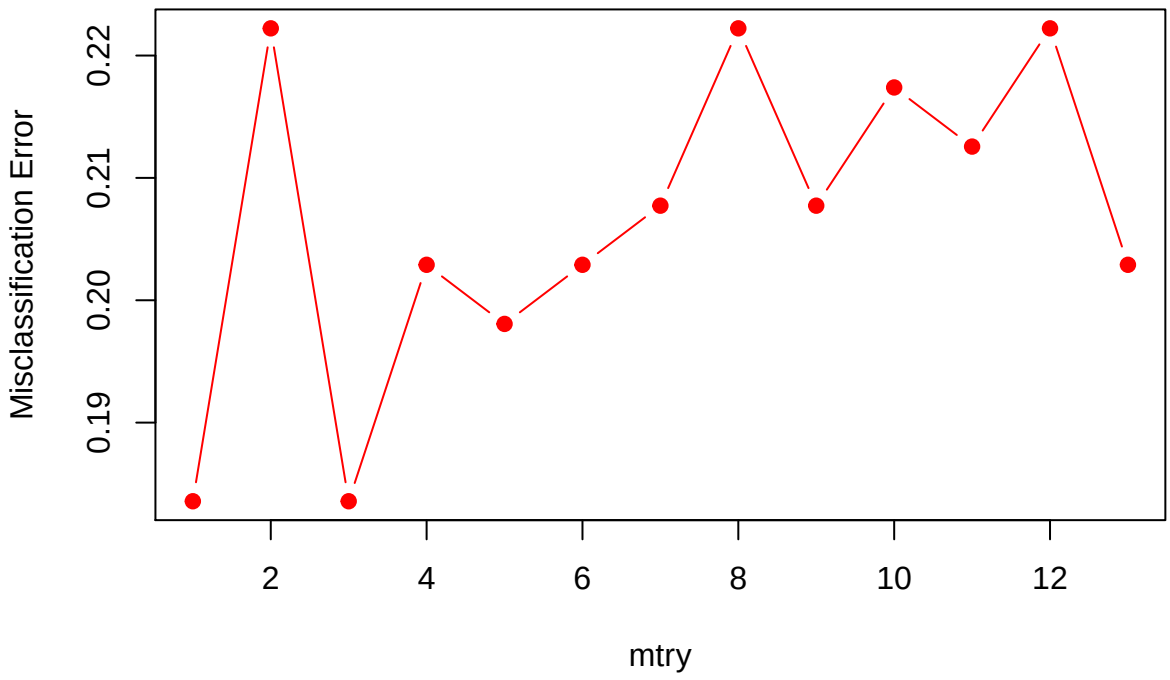
```
# setting mtry to sqrt(p) is a rule of thumb, this number can be set by
# k fold CV as well
rf.class <- randomForest(AHD~., data=train, mtry=round(sqrt(ncol(train) - 1)), importance=TRUE)
#importance(rf.class)
varImpPlot(rf.class)
```



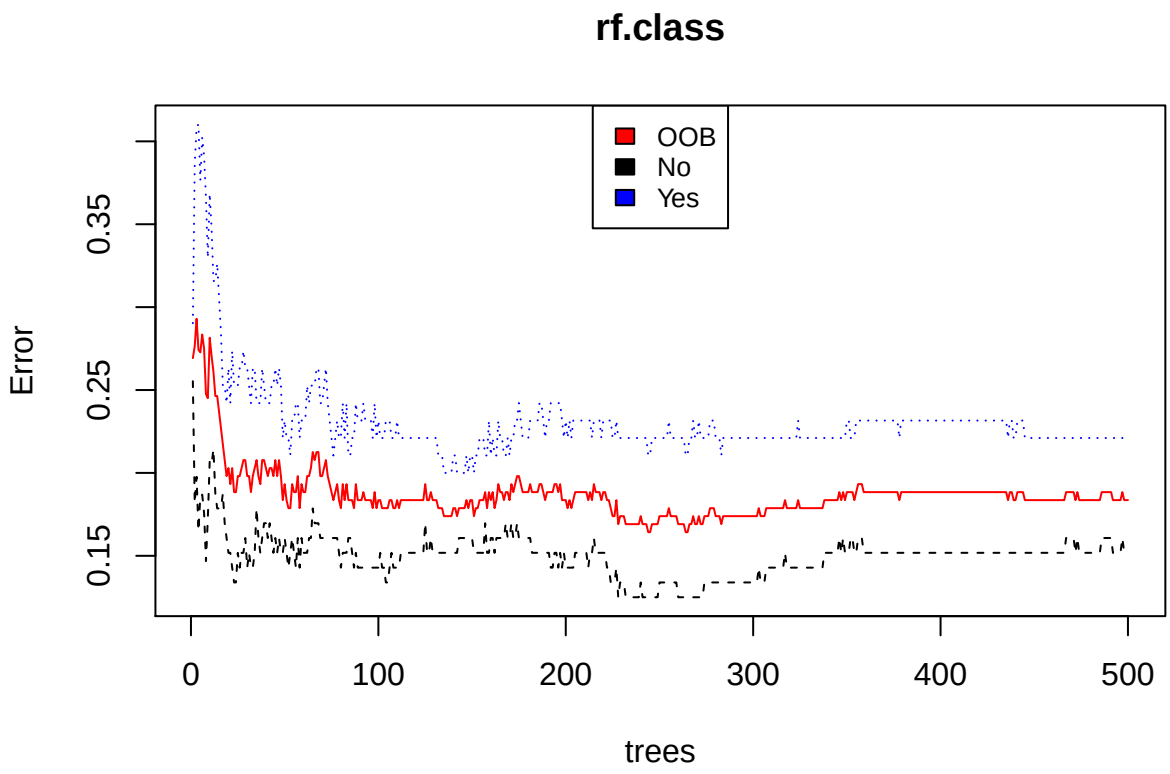

```
plot(rf.class, col=c("red", "black", "blue"))
legend("top", colnames(rf.class$err.rate) ,col=seq_len(3),cex=0.8,fill=c("red", "black", "blue"))
```




```
#Try choosing mtry by plotting the OOB error
p <- ncol(train) - 1
oob.error.class <- double(p) #initialize empty vector
set.seed(1)
for(m in 1:p) {
fit <- randomForest(AHD ~ ., data=train, mtry=m, ntree=175)
conf.mat <- fit$err.rate[175]
oob.error.class[m] <- fit$err.rate[175, 'OOB']
}
matplot(1:p, oob.error.class, pch=19, col="red", type="b", ylab="Misclassification Error", xlab="mtry")
```




```
# setting mtry to sqrt(p) is a rule of thumb, this number can be set by
# k fold CV as well
rf.class <- randomForest(AHD~., data=train, mtry=3, importance=TRUE)
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"))
```



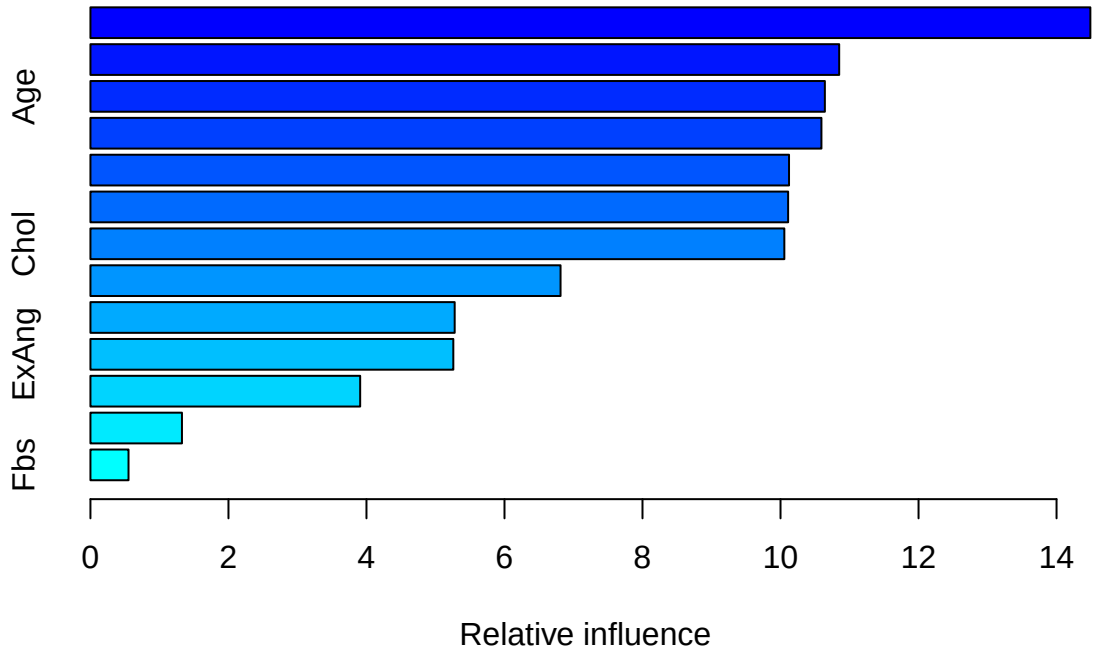
Boosting

```
library(gbm)
```

```
## Warning: package 'gbm' was built under R version 4.0.4
```

```
## Loaded gbm 2.1.8
```

```
set.seed(1)
#format y for gbm, must be 0/1
train$ChestPain = as.numeric(factor(train$ChestPain))
train$Thal = as.numeric(factor(train$Thal))
AHD.0.1 <- ifelse(train$AHD == 'Yes', 1, 0)
class.boost = gbm(AHD.0.1 ~ . - AHD, data = train, n.trees = 5000, distribution = "adaboost", shrinkage
summary(class.boost)
```




```
##                  var    rel.inf
## RestBP       RestBP 14.4899486
## Ca               Ca 10.8500863
## Age             Age 10.6427662
## Oldpeak     Oldpeak 10.5927425
## MaxHR         MaxHR 10.1234250
## Thal           Thal 10.1101401
## Chol           Chol 10.0551703
## ChestPain ChestPain  6.8123290
## Slope         Slope  5.2781342
## ExAng         ExAng  5.2586456
## Sex             Sex  3.9085508
## RestECG     RestECG  1.3262240
## Fbs             Fbs  0.5518371
```

Comparing Models

Citations