HeartDPred_PreAnalysis_F

Tyler Boudreau

2023-05-26

```
# Required packages
library(knitr)
library(caret)
```

```
## Loading required package: ggplot2
```

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

```
library(ggcorrplot)
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
library(randomForest)
```

```
## randomForest 4.7-1.1
```

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

```
##
## Attaching package: 'randomForest'
```

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

```
library(xgboost)
library(Metrics)
```

```
##
## Attaching package: 'Metrics'
```

```
## The following objects are masked from 'package:caret':
##
##     precision, recall
```

```
library(rpart)
library(rpart.plot)
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following object is masked from 'package:Metrics':
##
##     auc
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(ggplot2)
library(plotly)
```

```
##
## Attaching package: 'plotly'
```

```
## The following object is masked from 'package:xgboost':
##
##     slice
```

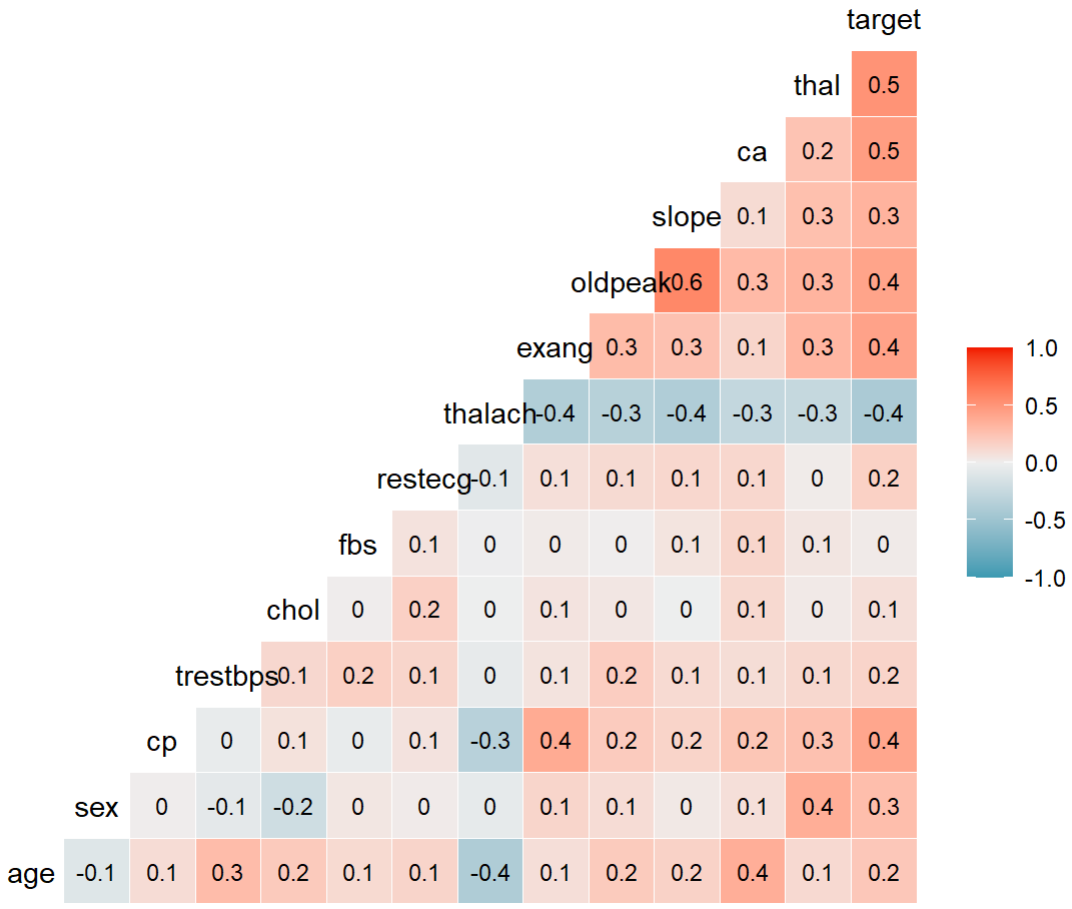
```
## The following object is masked from 'package:ggplot2':
##
##     last_plot
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

```
## The following object is masked from 'package:graphics':
##
##     layout
```

```
Clev_HeartD <- read.csv('C:\\Users\\Tyler\\Downloads\\Heart_disease_cleveland_new.csv') # File l
ocatiom for Data
set.seed(3573)
```

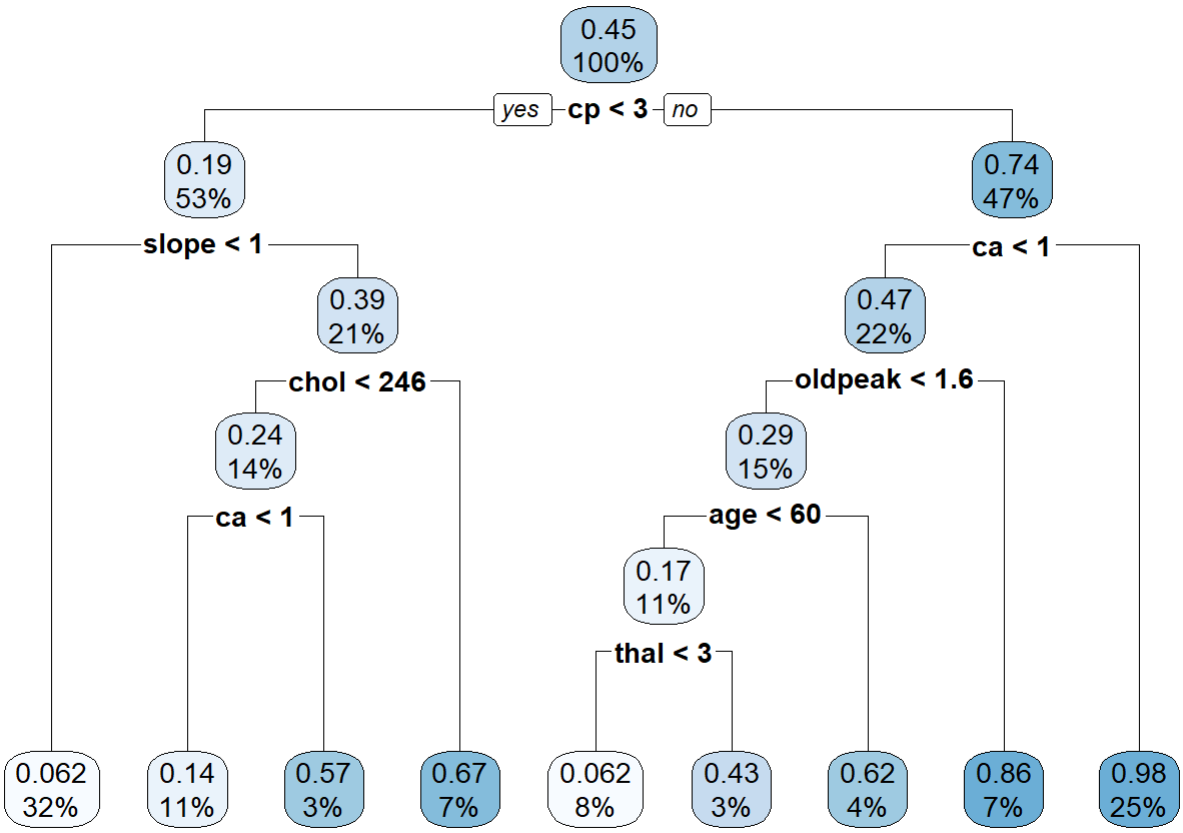
```
#plot correlation matrix to check for Multicollinearity
ggcorr(Clev_HeartD, label = TRUE, digits = 3, label_size = 3)
```




```
# Plot shows possible Multicollinearity, for Slope and Oldpeak variables

# Could remove least important one to possibly improve model if desired
```

```
set.seed(3573) # sets seed
sample1 <- sample(c(TRUE,FALSE),nrow(Clev_HeartD),replace=TRUE, prob = c(0.70,0.30)) # 70 percen
t of data for train / supervised learning, 30 percent for test / unsupervised learning.

train <- Clev_HeartD[sample1, ]#Supervised learning Train partition
test <- Clev_HeartD[!sample1, ]#Unsupervised learning Test Partition
```

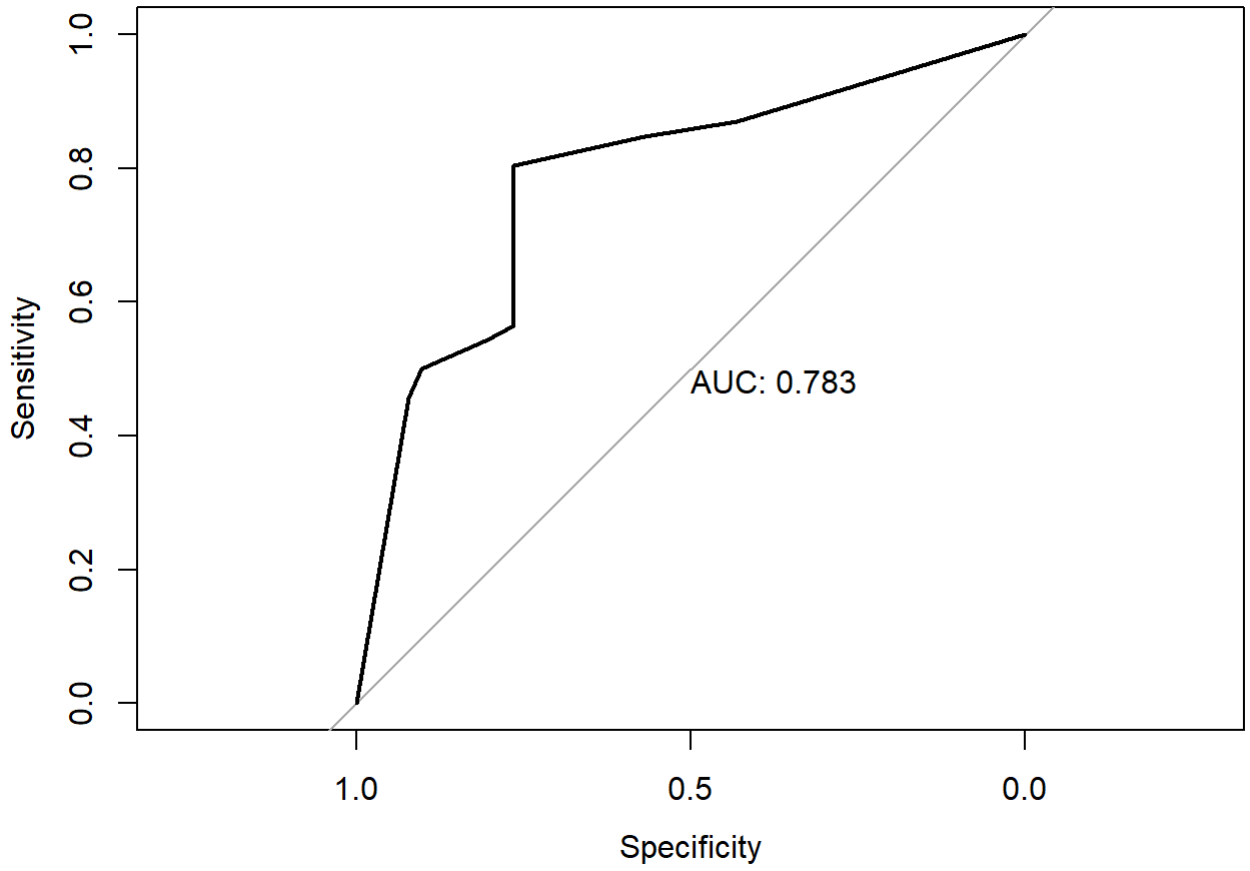
```
# Basic Decision tree model
HeartDTree <- rpart(target~.,data = train)
rpart.plot(HeartDTree)
```




```
pred_dt <- predict(HeartDTree,test)
test_rocDt = roc(test$target ~pred_dt,plot = TRUE, print.auc = TRUE) # ROC curve for Decision tr
ee model
```

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

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




```
# Random Forest Model
HeartDForest <- randomForest(target~., data = train, importance = TRUE)
```

```
## Warning in randomForest.default(m, y, ...): The response has five or fewer
## unique values.  Are you sure you want to do regression?
```

```
print(HeartDForest)
```

```
##
## Call:
##  randomForest(formula = target ~ ., data = train, importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##           Mean of squared residuals: 0.1211605
##                     % Var explained: 51.07
```

```
print(importance(HeartDForest,type=2))
```

```
##          IncNodePurity
## age          3.1732858
## sex          1.2589712
## cp           8.3457790
## trestbps     3.1119651
## chol         3.4695756
## fbs          0.1844395
## restecg      0.5484329
## thalach      4.1161402
## exang        2.1688242
## oldpeak      4.7264442
## slope        3.6461113
## ca           7.0058265
## thal         4.7640627
```

```
pred2 <- predict(HeartDForest,test)
print(mean((pred2-test$target)))^2
```

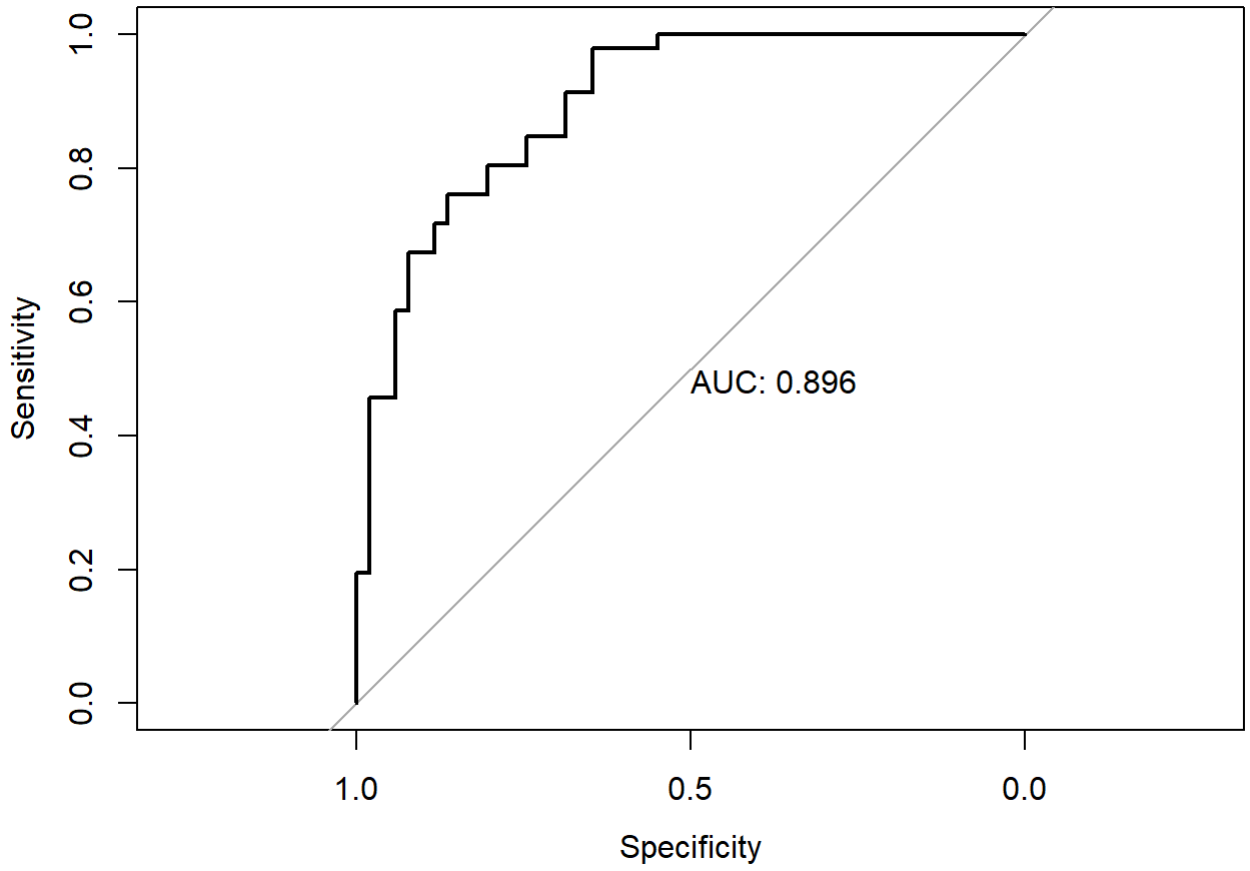
```
## [1] -0.05418585
```

```
## [1] 0.002936107
```

```
#Evaluate Reciver operator Curve ROC for model to assess accuracy for Random Forest model
test_roc = roc(test$target ~pred2,plot = TRUE, print.auc = TRUE)
```

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

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




```
train_x = data.matrix(train[,-14]) # 14 is number of predictors
train_y = train[,14]
test_x = data.matrix(test[,-14])
test_y = data.matrix(test[,14])
#HeartDTest1 <- data.frame(age = c(63), sex = c(1), cp = c(0), trestbps = c(145), chol = c(233),
fbs = c(1), restecg = c(2), thalach = (150), exang = c(0), oldpeak = (2.3), slope = c(2), ca= c
(0), thal = c(2)) example values to test
#predict(HeartDForest, HeartDTest1)
xgb_train <- xgb.DMatrix(data = train_x, label = train_y)
xgb_test <- xgb.DMatrix(data = test_x, label = test_y)
watchlist = list(train=xgb_train, test=xgb_test)
#Observe and compare Train and test rmse and pick when test unsupervised learning error is lowes
t to avoid overfitting and to have best model accuracy
xgbmodel = xgb.train(data = xgb_train, max.depth = 3, watchlist=watchlist, nrounds = 20)
```

```
## [1]  train-rmse:0.417670 test-rmse:0.460593
## [2]  train-rmse:0.365167 test-rmse:0.433078
## [3]  train-rmse:0.326485 test-rmse:0.413149
## [4]  train-rmse:0.298295 test-rmse:0.412289
## [5]  train-rmse:0.277528 test-rmse:0.408390
## [6]  train-rmse:0.261432 test-rmse:0.407246
## [7]  train-rmse:0.248700 test-rmse:0.408043
## [8]  train-rmse:0.238663 test-rmse:0.406963
## [9]  train-rmse:0.232247 test-rmse:0.408396
## [10] train-rmse:0.227435 test-rmse:0.410779
## [11] train-rmse:0.218184 test-rmse:0.407272
## [12] train-rmse:0.210946 test-rmse:0.408363
## [13] train-rmse:0.208032 test-rmse:0.409706
## [14] train-rmse:0.205023 test-rmse:0.408026
## [15] train-rmse:0.203300 test-rmse:0.407510
## [16] train-rmse:0.201803 test-rmse:0.409155
## [17] train-rmse:0.193218 test-rmse:0.411230
## [18] train-rmse:0.188337 test-rmse:0.413642
## [19] train-rmse:0.186378 test-rmse:0.414184
## [20] train-rmse:0.182139 test-rmse:0.410007
```

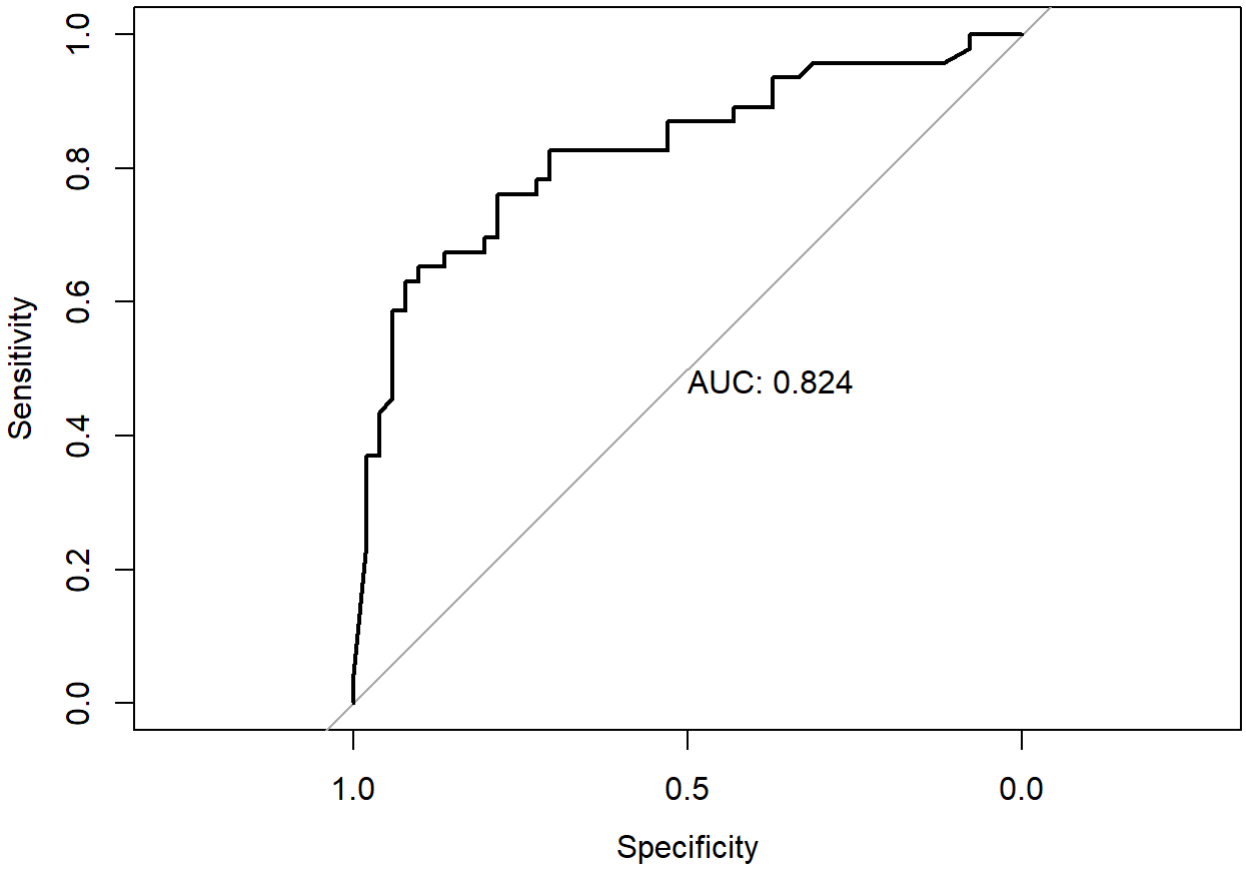
```
xgbfinal = xgboost(data = xgb_train, max.depth = 3, nround = 6, verbose = 0) #Final value for nr
ounds depends on your exact run, exaple for this is 6 rounds for lowest rmse
pred_y = predict(xgbfinal, xgb_test) # Make predictions with model

mse(test_y,pred_y) # evaluate error of model
```

```
## [1] 0.1658495
```

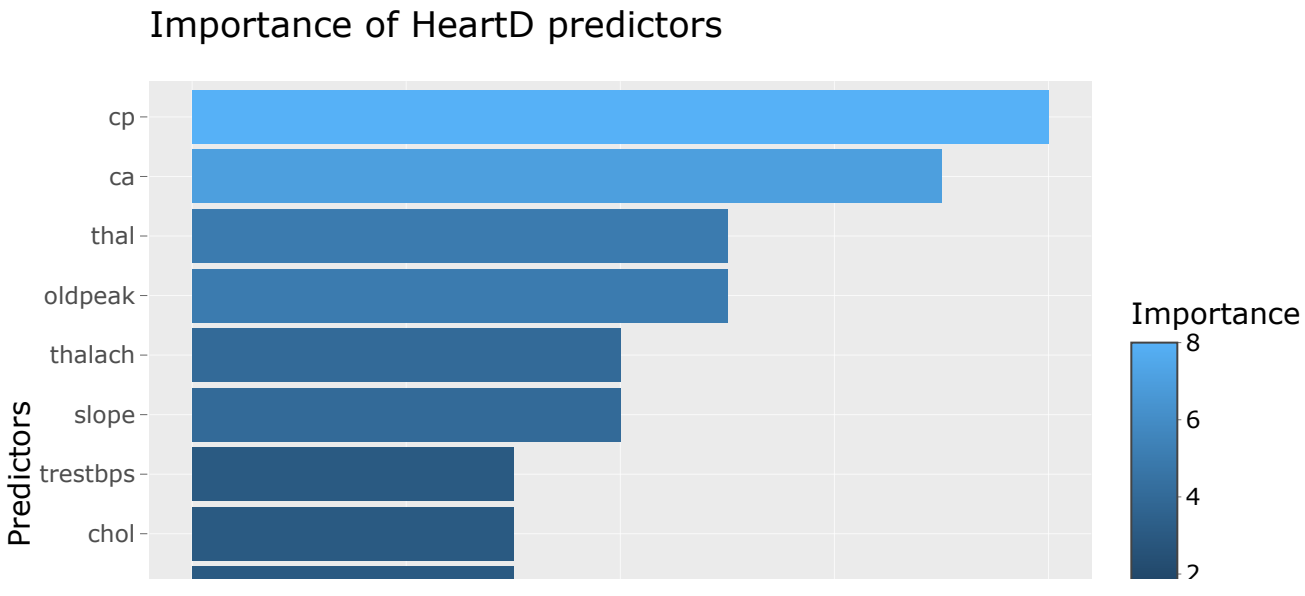
```
#Evaluate Reciver operator Curve ROC for model to assess accuracy for xgboost model
test_roc = roc(test_y~pred_y, plot = TRUE, print.auc = TRUE)
```

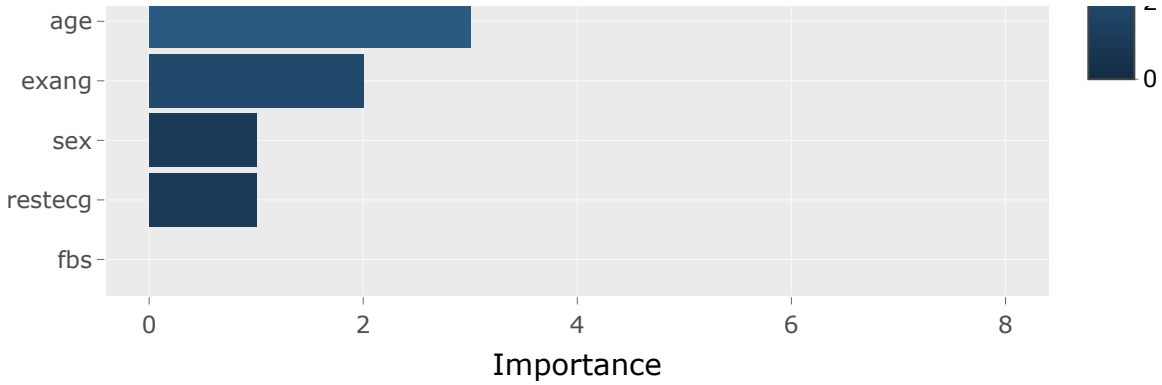
```
## Setting levels: control = 0, case = 1
```

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




```
#Create importance Bar Graph
importance <- importance(HeartDForest)
vImportance <- data.frame(Variables = row.names(importance),Importance =round(importance[,'IncNo
dePurity'], 0))

ggplotly(ggplot(vImportance, aes(x = reorder(Variables, Importance),
y = Importance, fill=Importance))+geom_bar(stat='identity') +
labs(title = 'Importance of HeartD predictors', x = 'Predictors', y = 'Importance') +
coord_flip())
```






```
#CP and ca are most important factors
```