Statistical Modelling Project

STAT9155: Statistical Modelling II

Fan Feng, Alyssa Gagne, Selam Mequanint and Anderson Or

April 13th, 2019

Introduction

The Cleveland heart disease dataset used for this project was donated to the Kaggle website by David W. Acha from the University of California, Irvine (*UCI*). The dataset includes 303 observations and a subset of 14 out of 76 attributes. Table 1 includes a description of the atributes used in the project.

For this project a tree-based method was used, which is a classifying algorithm. The method is used for this project to predict *heart disease* using the 14 predictors in the dataset. The basic idea of these tree-methods is to partition the space hierarchically and identify some representative centroids. Tree-based methods are simple for interpretation and can provide accurate predictions, particularly with the application of bagging, random forests, and boosting as will be implemented for this project. *Decision trees* are typically drawn upside down, in the sense that the leaves are at the bottom of the tree. *Classification trees* have a hierarchical way of partitioning the space. We start with the entire space and recursively divide it into smaller regions. In the end, every region is assigned to a class label; in this project the class label will be either a target of 0 or a target of 1. The advantages of using decision trees are firstly they implicitly perform variable screening or feature selection; secondly they require relatively little effort from users for data preparation and lastly the non-linearity of the data doesn't affect them. In addition, *decision trees* are easy to interpret by those without a background in statistical modelling.

Table 1: Cleveland Heart Disease Variables

	Description
age	Age in years
sex	Sex (1 = male; 0 = female)
cp	Chest pain type (1 = typical angina; 2 = atypical angina; 3 = non-anginal pain; 4 = asymptomatic)
trestbps	Resting blood pressure measured in mm Hg
chol	Level of serum cholesteral measure in mg/dl
fbs	Fasting blood sugar being below or above 120mg/dl (1 = true; 0 = false)
restecg	Resting electrocardiographic results (0 = normal; 1 = having ST-T; 2 = hypertrophy)
thalach	Maximum heart rate achieved measured in bps
exang	Exercise induced angina (1 = yes; 0 = no)
oldpeak	ST depression induced by exercise relative to rest
slope	Slope of the peak exercise ST segment (0 = downsloping; 1 = flat; 2 = upsloping)
ca	Number of major vessels (0-3) colored by flourosopy
thal	Thallium stress test (3 = normal; 6 = fixed defect; 7 = reversable defect)
target	Presences of heart disease (1=true;0=false)

Data Cleaning

The dataset published on Kaggle contained few missing values and was previously cleaned. The only adjustment made to the data was for the attribute `Thal` , which was originally coded in levels of 1,2 and 3. This was changed to 6,3 and 7 respectively to represent the true reading of a *Thallium stress test*, where 3 represents a *normal* stress level, 6 a *fixed defect* and 7 a *reversable defect.*

Data Preprocessing

Our dataset contains categorical variables initially labelled as integer variables in *R*. To facilitate modelling, they were converted to factor variables. In addition, there were two cases with *zero* values for thallium stress test (`thal`) and five records with invalid categorical numbers for the number of major vessels (`ca`). Therefore, a total of seven cases were removed from the dataset.

Exploratory Data Analysis

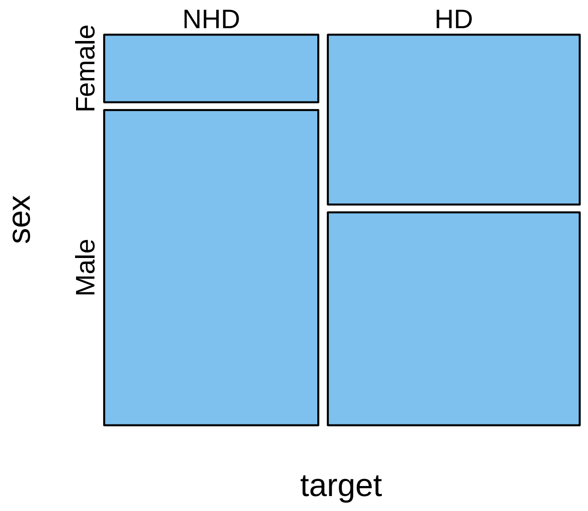


Figure 1: Mosaic plot of patients by sex

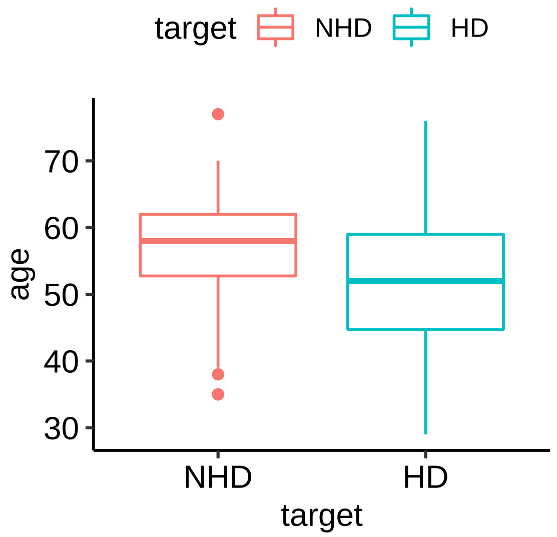


Figure 2: Boxplot of patients with and without heart disease by age

Figure 1 describes that of the patients with *heart disease*, there is no signifigant difference in numbers between men and women. Nonetheless, in Figure 2, the mean age of the patients with *heart disease* is lower compared to the patients without *heart disease*. Figure 3 showcases that the average cholesterol level in people with *heart disease* is slightly lower than those without however, there are a few outlier patients without *heart disease* with very high levels of cholesterol.

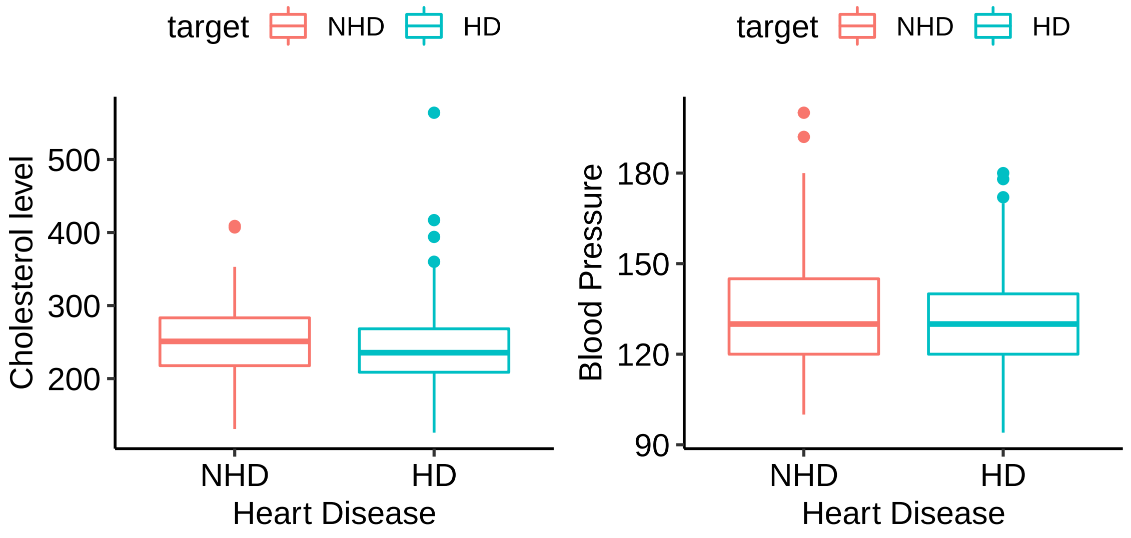


Figure 3: Boxplots of patients with heart disease against their cholesterol level and blood pressure

Correlation matrices are useful to explore if there is any collinearity among predicting variables, particularity for continuous variables. Figure 4 showcases that there was no strong correlation among the continuous predictors.

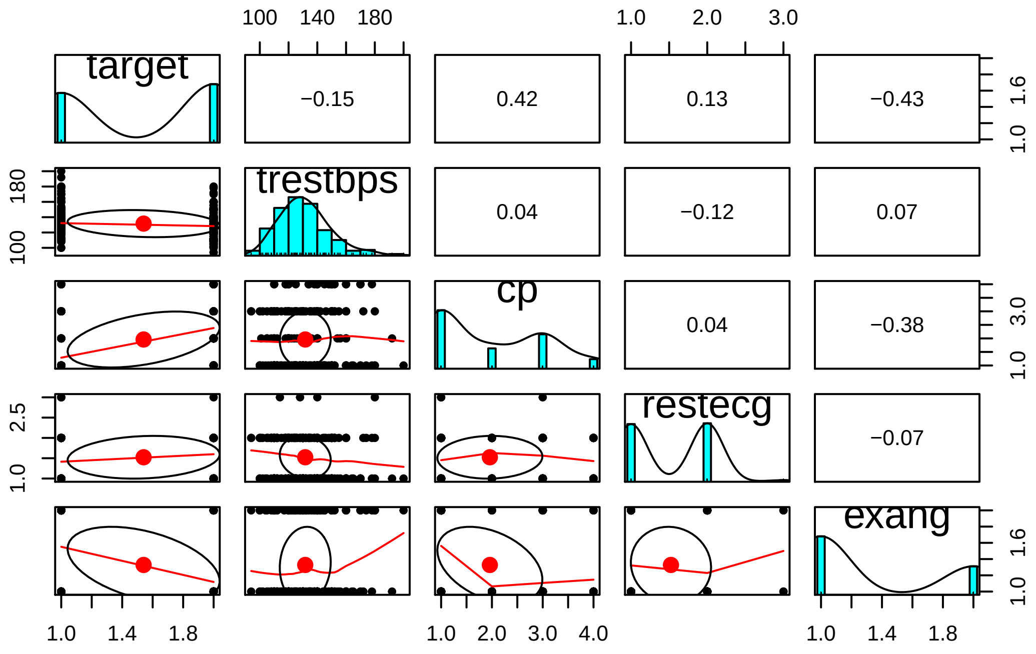
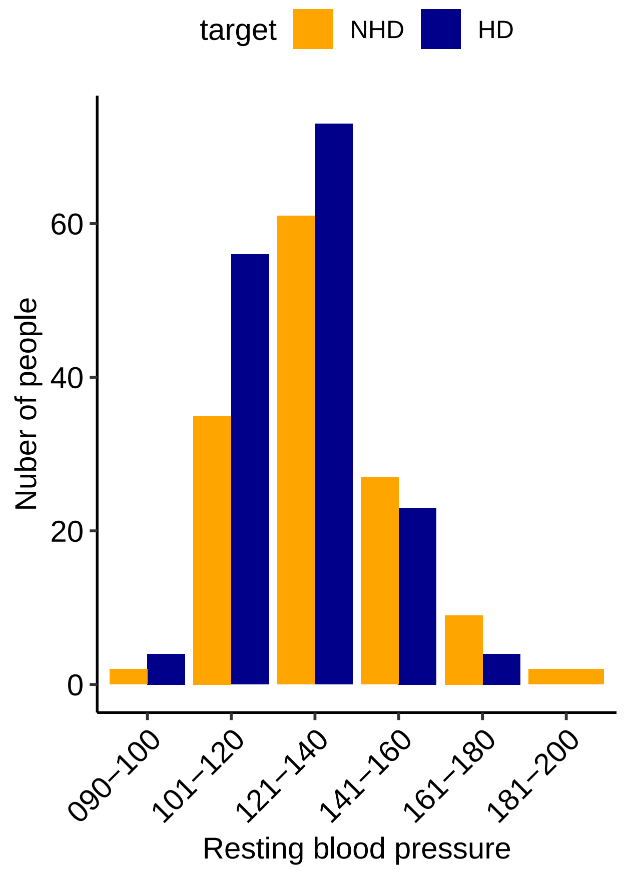


Figure 4: Correlation Matrix of predictors in dataset



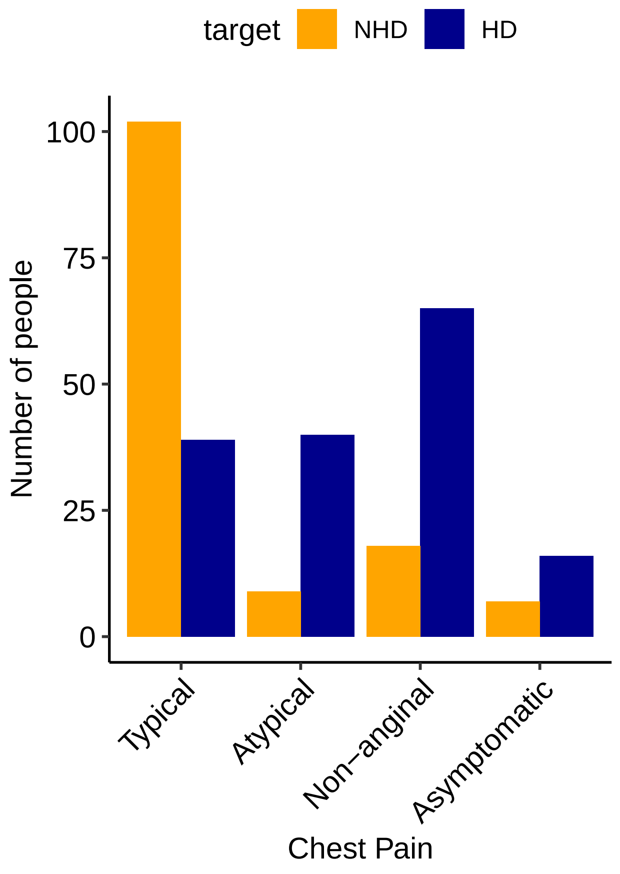


Figure 5: Number of patients with hearth disease by resting blood pressure and categories of chest pain

Both plots in Figure 5, describe how the number of patients with *heart disease* varies within each level of *resting blood pressure* and categorization of experienced *chest pain*. However, within both plots it can see what category have a much larger group of patients with *heart disease* indicating that *chest pain* and *resting blood pressure* are strong predictors of *heart disease*.

Methods

Tree-based Methods

Tree-based methods were first introduced by Breiman et al. (1984). Tree-based methods can be used both for regression and classification problems. A tree involves a root node, branches and leaves. The *root* node contains all the observations in the dataset. From the *root* node observations are sent to one of two descendant nodes split by the most important predictor variable. The level of predictor variables importance is determined based on some criterion including the sum of squared errors (SSE) for regression problems, and Gini index and entropy for classification problems. The trees are grown until all nodes contain fewer than some fixed number of cases, then “pruned” back to prevent overfitting(Breiman et al., 1984).

In addition, tree-based ensemble approach can be used to improve the accuracy and performance of models.

Tree-based Ensembles

Tree-based ensembles are models that combine the predictions of many different trees with low bias and high variance, which are weak models, to give an aggregated prediction with lower variance than the individual trees, while maintaining low bias (Hastie et al., 2001). Ensemble methods use randomness tree-fitting procedures or fit non-random trees to different versions of the dataset or do both procedures (Dietterich,2000). Ensembles used different methods to aggregate predictions. The average of the individual tree predictions is most commonly used for the regression problem and the *voting* trees approach is prominent in classification problems. Individual classifiers make prediction errors in different regions of the predictor space. Thus, ensembles provide an improved prediction accuracy over individual trees by *out voting* the incorrect predictors with the correct ones. In this project, the most common tree-basedensembles: Random forest and Gradient boosting models were implemented to obtain accurate predictions. *Random forests* (Breiman, 2001) are tree-based ensembles that use bootstrap samples and randomness in the tree-building procedure. For the regression problem, random forests build a large collection of de-correlated trees to obtain their prediction average. When used for classification, a random forest obtains a class vote from each tree and then classifies using majority vote. Similar to other tree-based methods,the random forest has a few tuning parameters, which can be chosen using cross-validation. *Random forests* are simpler to train and tune, which made them more popular and can be implemented in a variety of `Rpackages`. A conditional inference tree *cForest*, which is claimed to have better predictive power and variable importance measure was implemented in this project for better performance. Conditional inference trees (Hothornet al., 2006) is where the node split is selected based on the level of strength of its association with the observed value of the dependent variable helps to reduce bias due to categories and selects relevant variables. Just as popular as *Random Forests* and used in many other machine learning methods, *Gradient Boostedtrees* (GBT), invented by Jerome H. Friedman in 1999, are another type of ensemble method. Gradient boosting train a bunch of individual models in a sequential way and each member of the ensemble learns on the error of its predecessor. The final model is the aggregation of all individual predictions. GBT has many hyperparameters grouped into two categories: tree structure and regularization to set to achieve better performance. Like any other methods, GBT has strength and limitations as well. GBT handles data of mixed type, detects non-linear feature interaction and handles outliers very well. However, it can be slow to train and difficult to parallelize due to its sequential nature.

Data Splitting and Cross Validation

A big issue in *statistical learning* is *overfitting* a model to a dataset. Overfitting occurs when the model is so accurate at predicting the data from the initial dataset, that it is not generalizable to new data. To prevent *overfitting*, when the dataset size is adequate, the dataset is split into 2 subsets: a training set and test set. The model is then trained on the training set and tested on the test set. Testing on the test sets allows for an adequate measure of *accuracy* of the model. Good accuracy on a training set does not speak as loud as good *accuracy* from a test set, which contains observations the model has never seen. k-fold *cross validation* is used aswell to prevent overfitting. It works by shuffling the data and then splitting it into k groups. For each group, it holds it out as a test set and uses the other k-1 groups as a training set to fit a model. Accuracy of each model fitted is retained and the final model returned is the most *accurate* one. *Cross-validation* is commonly used when there is not enough data to split it into *two* initial subsets and have an adequate number of observations to both train and test on.

Evaluation Metrics

The *Receiver Operating Characteristic* (ROC) curve is used to measure the performance for a classification problem at different threshold levels. Based on the cutoff value, the curve plots the true positive rate against the false positive rate. The ROC is a probability curve, while *Area Under the Curve* (AUC) is the area under the ROC curve. AUC helps to measure the separability. An excellent model has an AUC close to 1, which indicates it has good measure of separability, while a poor model has an AUC close to 0, which indicates a poor measure of separability. When the AUC is 0.5, it means the model has no class separation capacity. ROC and AUC are used together to evaluate the strength of a model. As previously mentioned, *accuracy* is a measurement of performance of a model on a test set.

Data Modeling

Decision Tree

The first *decision tree* created was a basic unpruned classification *decision tree*. Quite simply, each observation (patient's data) moves down the tree going right if the expression at the split is true and left if it is false. The last stage of each branch of the tree determines the classification on whether the patient has *heart disease*. The test set was then run through the unpruned decision tree, rendering an *accuracy* of 81.81% for the tree in predicting if the patient has *heart disease*. The *accuracy* was reasonable,however pruning of the tree was done to improve the *accuracy* of the tree by reducing the complexity of the tree and reducing the overfitting of the model. As seen in the confusion matrix in Table 2, the *accuracy* with the pruning of the tree, did not increase. The *pruned decision tree* can be seen in the Figure 6.

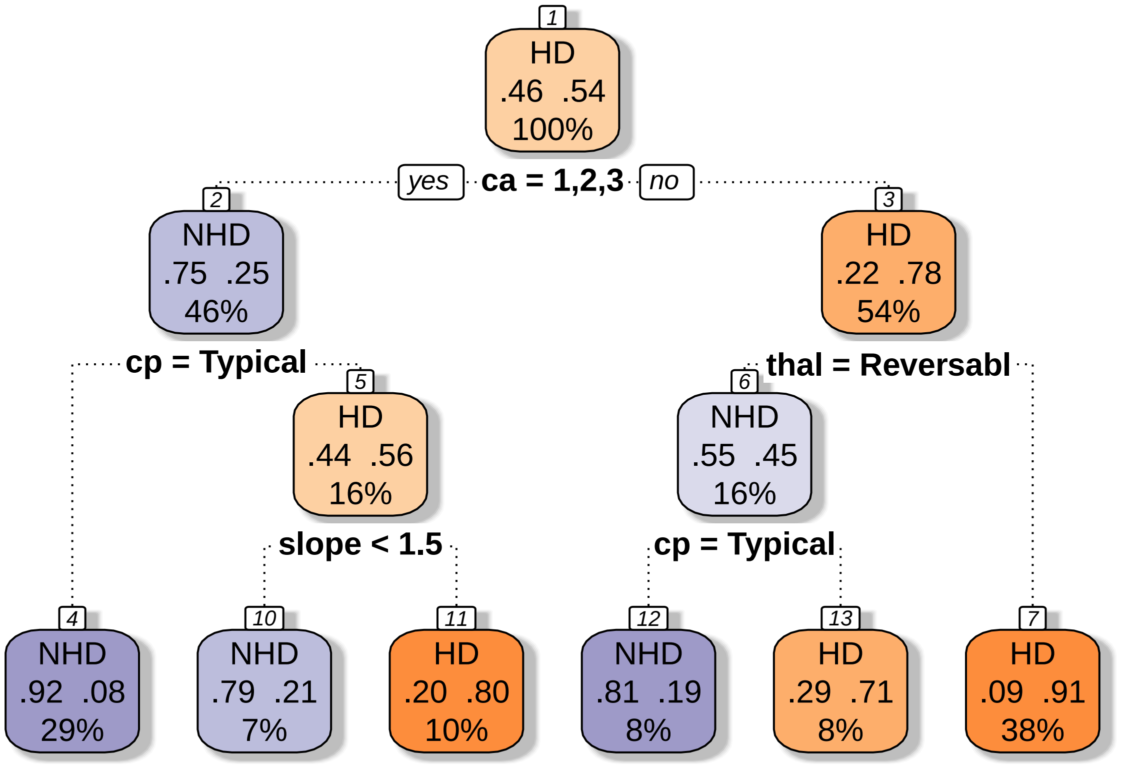


Figure 6: Pruned decision tree

Table 2: Confusion Matrix

Predicted	Observed	
	NHD	HD
NHD	28	4
HD	12	44

The next decision tree built was created using the `caret` package in *R*. There are two primary methods used to create decision trees with `caret`, the first splits the data, while the second performs a 10 fold cross validation.The predictor `cp` was the parameter used by the `caret` package to prune the tree.

For the first `caret` *decision tree* model, performance was measured by splitting the data into a training and a testing set with a proportion of 70/30. The model was fit to the training set, then it was measured how well the model could predict on the test set. The 10-fold cross-validation method, which is the resampling procedure by splitting the dataset to 10 number of groups evaluate the performance of the model was used. It used 9 10 folds to fit the model and then tested the model on the fold left out. This process was repeated 10 times using always one fold for testing. Cross-validation is an appropriate model performance evaluation technique for this study as it is widely used to estimate prediction accuracy and recommended for small dataset where splitting the data into training and test dataset compromise the fitness of the model.

The cross validation method returned a better accuracy of 76.67%, compared to an *accuracy* 72.70% for the model that split the data. Details on the `caret` *split method* and its code can be found in the *Appendix* . The results for the *cross validation* `caret` *decision tree* can be seen in Figure 7 and the size of the tree can be noted to be much smaller and simpler to understand compared to the pruned *decision tree* in Figure 6, as it has most noticeably only 2 levels.

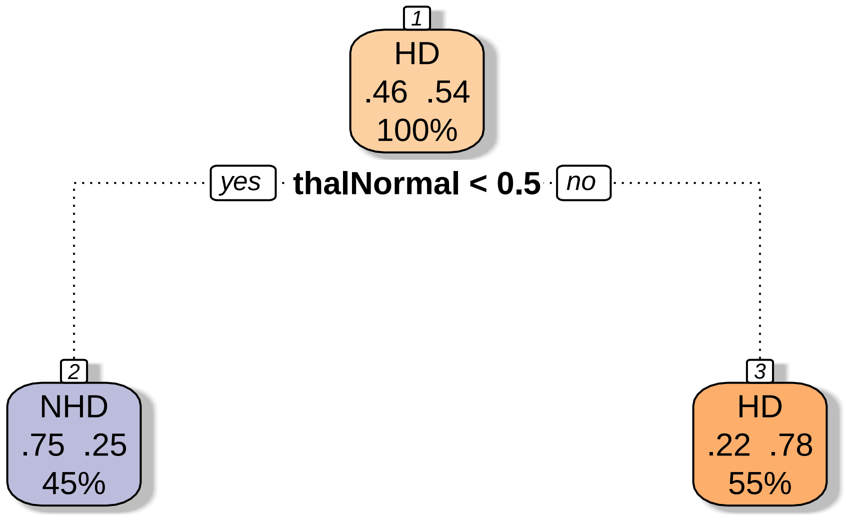


Figure 7: Decision tree with random search

Table 3: Accuracy of caret decision tree

	cp	Accuracy
1	0.00	0.74
2	0.05	0.77
3	0.11	0.77
4	0.16	0.77

Ensemble Methods

Random Forest

Similarly to the *decision tree* built previously, the `caret` package in *R* was used to create a *random forest.* The `caret` package uses two tuning parameters, which can seen in Table 4.

Two different types of search algorithms were used with the *Random Forest* cross-validation method. The first was a search-based algorithm and the second a grid-search algorithm to determine the best *mtry* tuning parameter. In addition, the default *ntree* value of 500 was used. Both *random Forests* were created using a 10-fold repeated cross-validation and both algorithms used with the *random Forest* method produced the same *accuracy* of 81.5%. Figre 8 showcases the *accuracy* produced.

Table 4: Tuning parameters for Random Forest

	Tuning.parameters	Description
1	mtry	number of variables randomly sampled at each split
2	ntree	number of trees to grow

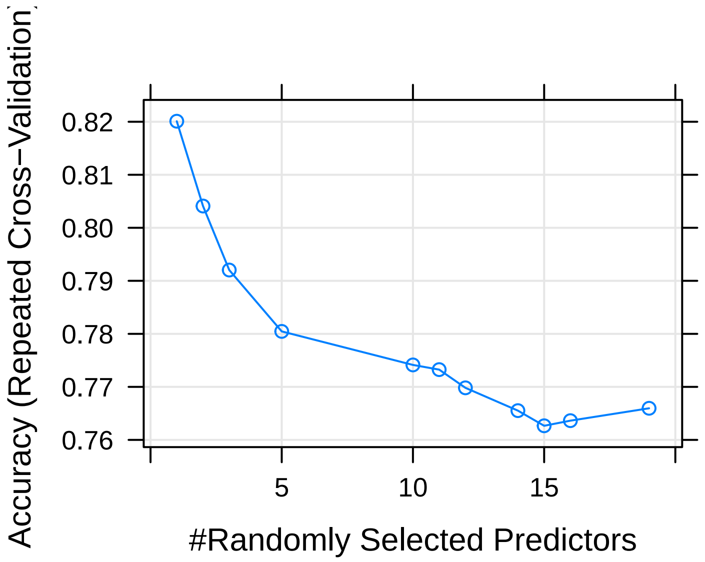


Figure 8: Accuracy of Random Forest

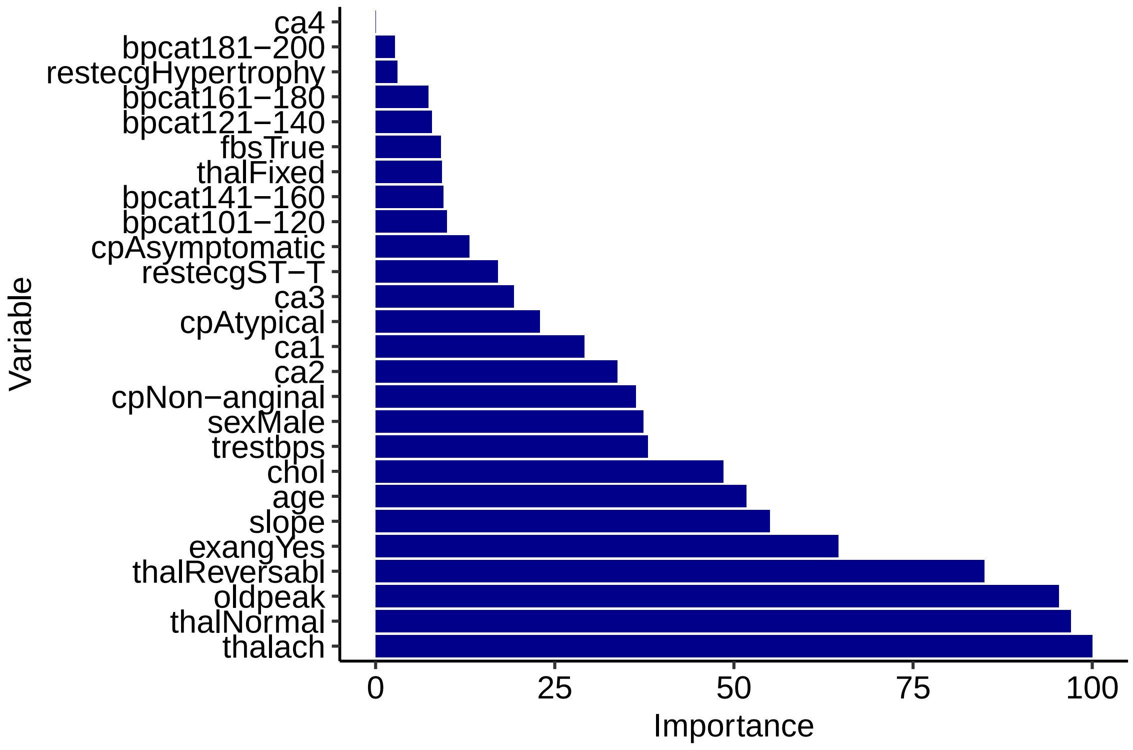


Figure 9: Random forest variable importance

Table 5: Accuracy for Random Forest

	mtry	Accuracy
1	1.00	0.81
2	2.00	0.80
3	4.00	0.78
4	8.00	0.78

Figure 9 describes the importance of the variables, decided by the *random Forest* produced. *Random Forest* indicates that the values the predictors such as *thalReversable* and *thalach* hold, heavily influence if the observed patient has *heart disease.*

The `cForest` package in *R* is computationally more expensive and better than the `randomForest` package in terms of accuracy, in terms of creating a *Random Forest.* `cforest` uses Out-Of-Bag (OOB) data which means, it calculates error by using the OOB data (data not used to make the forest) as a test set and calculates its classification error. It is important to note that it is slower to run and handles less data, however due to the size of this dataset, it will not be an issue. To get the final ensemble, it uses weighted average of the trees.

Table 6: Accuracy for cForest

	mtry	Accuracy
1	2.00	0.82
2	11.00	0.79
3	21.00	0.78

As Table 6 describes, the most optimal *mtry* parameter for the best *accuracy* of 0.8224 for the model, was 2.

The *cForest* ran on the test data revealed an accuracy of 79.55%, which is reasonable however lower than the accuracy of the *unpruned decision tree.*

Table 7: Confusion Matrix

Predicted	Observed	
	NHD	HD
NHD	29	10
HD	11	38

Gradient Boosted Tree

As Table 5 describes, the most optimal *n.trees* parameter for the best *accuracy* of 84.20% for the model, was 50. However, once tested on the test set, it performed with an *accuracy* of 77.284%.

For the ensemble methods, the *Receiver Operator Characteristic* (ROC) curve can also be used to compare the *Random Forest*, *cForest* and *gbm.* The *ROC* curve is a performance measurement for classification problems. It describes how well the model is able to distinguish between the classes. The larger the *Area Under the Curve* (AUC), the better the model is at making accurate predictions.

Figure 10 plots the 4 different ROC curves for the different ensemble models and the *decision tree.* It can be see the *random forest* performed the best, with the largest Area Under the Curve (AUC).

Table 8: Accuracy for Gradient boosting

	mtry	Accuracy
1	2.00	0.82
2	11.00	0.79
3	21.00	0.78

Table 9: Confusion Matrix

Predicted	Observed	
	NHD	HD
NHD	29	10
HD	11	38

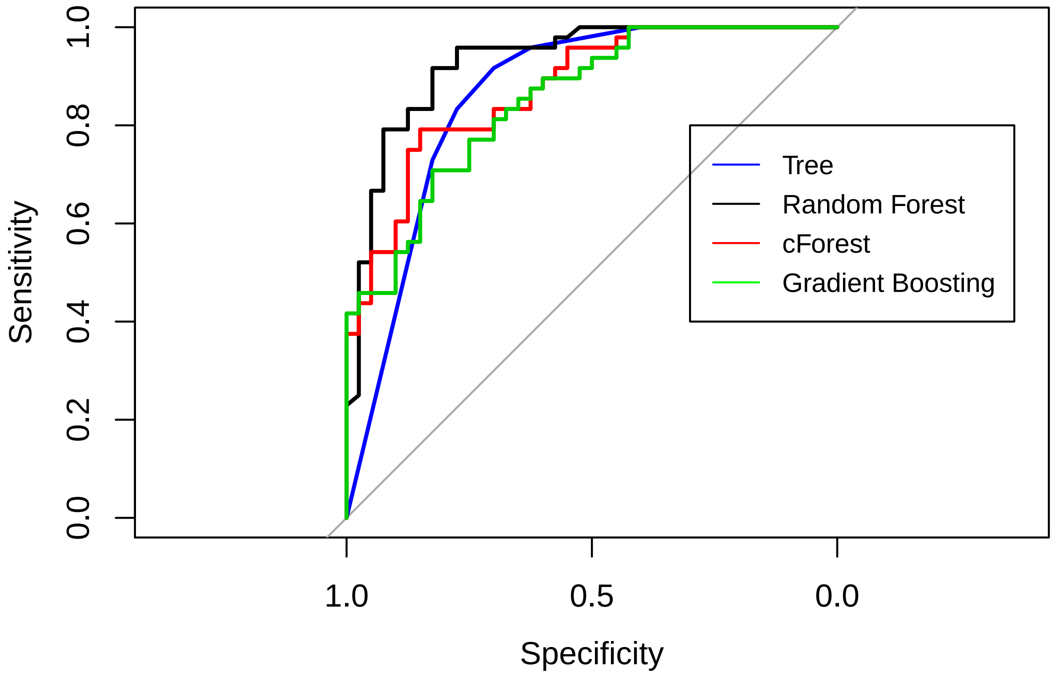


Figure 10: ROC curve

Conclusion

The main objective of this project was to find the best tree method that would predict whether a *patient* has a heart disease given different measurements and observations on the patient. Through boxplot diagrams in the exploratory data analysis section, it was seen that patients with *heart disease* were likely to be younger, with a lower level of cholesterol. All classification trees fitted methods had reasonable *accuracy* in predicting whether a *patient* has heart disease. Specifically, the * unpruned decision tree* and *random forest* were the two models with the highest *accuracy*, both above 80%. The Receiver Operator Characteristics (ROC) curve, allowed the *random forest* model to be chosen as the final model, as it had the largest *AUC* and one of the highest *accuracies*. The *random forest* method indicates that the *Thallium stress test* is the key predictor in identifying the presence of *heart disease*. This result was not surprising as the *Thallium stress test* is a nuclear imaging test that shows how well a *patient's* blood flows into the heart during exercise or at rest.

Table 10: Accuracy of all models

	Models	Accuracy
1	Unpruned decision tree	81.81%
2	Pruned decision tree	81.81%
3	Random forest with grid search	81.50%
4	Random forest with random search	81.50%
5	cForest	79.55%
6	Gradient boosted tree	77.73%
7	Caret split-data decision tree	76.67%
8	Caret cv decision tree	72.70%

Contributions

FF, AG, SM and AO contributed equally to this project by taking on on different roles and responsibilities. SM worked on data cleaning and preparation. AO and FF worked on developing graphs and plots in the explanatory data analysis phase. AG and SM reviewed AO and FF work and added more tables and plots. AG took the responsibility of creating, maintain and finalizing the R markdown. Each member had the responsibility of developing their *R* codes and summarize their work in writing to pass it to AG. AG collected all pieces of code and summarized them to develop the draft report. FF and AO ran the decision tree and random forest models. SM added more tree-based ensemble models using the 'caret' package for comparison. FF and AO drafted the PowerPoint slides. All members reviewed the draft report and slides and provided their feedback. Comments, inputs and suggestions obtained from all team members were incorporated into the final versions.

References

1.Carnegie Mellon Unviersity. (2011). Evaluating Statistical Models. Data Analysis 36-402. 2.Breiman, L. and Ihaka, R. (1984). Nonlinear discriminant analysis via scaling and ACE, Technical report, University of California, Berkeley. 3.Hastie, T., Tibshirani, R. and Friedman, J.(2001).The Elements of Statistical Learning. Data Mining, Inference, and Prediction. Springer, New York. 4.Dietterich, T. (2000). Ensemble methods in machine learning, Lecture Notes in Computer Science 1857: 1–15. 5.Breiman, L. (2001). Random forests, Machine Learning 45: 5–32. 6.Hothorn T, Hornik K, Zeileis A (2006). "Unbiased Recursive Partitioning: A ConditionalInference Framework." Journal of Computational and Graphical Statistics,15(3), 651–674 7.Friedman, J.H., 1999. Greedy function approximation: a gradient boosting machine. TechnicalReport, Department of Statistics, Stanford University.

Citations

1. Hungarian Institute of Cardiology. Budapest: Andras Janosi, M.D.
2. University Hospital, Zurich, Switzerland: William Steinbrunn, M.D.
3. University Hospital, Basel, Switzerland: Matthias Pfisterer, M.D.
4. V.A. Medical Center, Long Beach and Cleveland Clinic Foundation:Robert Detrano, M.D., Ph.D.

Appendix

```
library(ggplot2)
library(plyr)
library(vip)
library(knitr)
opts_chunk$set(echo = TRUE, fig.pos= "H")
library(tidyverse)
library(xtable)
library(vcd)
#formating tables
library(xtable)
library(RColorBrewer)
library(rattle)
#data wrangling
library(dplyr)

#plots
library(plotrix)
library(psych)
library(ggplot2)
library(ggpubr)
theme_set(theme_pubr())

#data modeling
library(rpart)
library(rpart.plot)
library(caret)
library(randomForest)
library(tree)
library(party)

library(plotrix)
library(psych)
library(kableExtra)
library(pROC)

# Preset some options for printing your xtables
options(xtable.caption.placement = 'top',
        xtable.include.rownames = TRUE,
        xtable.comment = FALSE,
        xtable.booktabs = TRUE)
heartdata <- read.csv(file = "heart.csv")
#data dictionary
datat1 <- tibble(
age = "Age in years",
sex = "Sex (1 = male; 0 = female)",
cp = "Chest pain type (1 = typical angina; 2 = atypical angina; 3 = non-anginal pain; 4 = asymptomatic)
trestbps = "Resting blood pressure measured in mm Hg",
chol = "Level of serum cholesteral measure in mg/dl",
fbs = "Fasting blood sugar being below or above 120mg/dl (1 = true; 0 = false)",
restecg = "Resting electrocardiographic results (0 = normal; 1 = having ST-T; 2 = hypertrophy)",
thalach = "Maximum heart rate achieved measured in bps",
```

```r
exang = "Exercise induced angina (1 = yes; 0 = no)",
oldpeak = "ST depression induced by exercise relative to rest",
slope = "Slope of the peak exercise ST segment (0 = downsloping; 1 = flat; 2 = upsloping)",
ca = "Number of major vessels (0-3) colored by flourosopy",
thal = "Thallium stress test (3 = normal; 6 = fixed defect; 7 = reversable defect)",
target = "Presences of heart disease (1=true;0=false)"
)
tb1 <- as.data.frame(datat1)
rownames(tb1) <- c("Description")
tb <- t(tb1)

print(xtable(tb,
        caption="Cleveland Heart Disease Variables"), type="latex")
#Data Cleaning
heartdata$thal[heartdata$thal == 3] <- 7
heartdata$thal[heartdata$thal == 2] <- 3
heartdata$thal[heartdata$thal == 1] <- 6
names <- c('sex' ,'cp', 'fbs', 'restecg', 'exang', 'ca', 'thal', 'target')
heartdata[,names] <- lapply(heartdata[,names] , factor)
heartdata<-subset(heartdata, thal!="0")
heartdata<-subset(heartdata, ca!="4")
droplevels(heartdata)#drops unused levels
nrow(heartdata)
levels(heartdata$sex)<-c("Female", "Male")
levels(heartdata$cp)<-c("Typical","Atypical","Non-anginal", "Asymptomatic")
levels(heartdata$fbs)<-c("False","True")
levels(heartdata$restecg)<-c("Normal", "ST-T","Hypertrophy")
levels(heartdata$exang)<-c("No","Yes")
levels(heartdata$slope)<-c("Down","Flat","Up")
levels(heartdata$thal)<-c("NA","Normal","Fixed","Reversabl")
levels(heartdata$target)<-c("NHD", "HD")

tbl2 <- xtabs(~target + sex, heartdata)
mosaicplot(tbl2,main = " ",cex=0.8, color = "skyblue2", cex.main=0.5)
ggplot(heartdata, aes(x=target, y=age, color=target)) +
  geom_boxplot()
ggplot(heartdata, aes(x=target, y=chol, color=target)) +
  labs(x="Heart Disease",
       y="Cholesterol level")+
  geom_boxplot()

ggplot(heartdata, aes(x=target, y=trestbps, color=target)) +
  labs(x="Heart Disease",
       y="Blood Pressure")+
  geom_boxplot()
subdata<-heartdata[, c(14,4,3,7, 9)]
pairs.panels(subdata,cex=0.5)
library(reshape)
attach(heartdata)
heartdata$bpcat[trestbps > 90 & trestbps<=100] <- "090-100"
heartdata$bpcat[trestbps > 100 & trestbps<=120] <- "101-120"
heartdata$bpcat[trestbps > 120 & trestbps<=140] <- "121-140"
heartdata$bpcat[trestbps > 140 & trestbps<=160] <- "141-160"
```

```r
heartdata$bpcat[trestbps > 160 & trestbps<=180] <- "161-180"
heartdata$bpcat[trestbps > 180 & trestbps<=200] <- "181-200"
mydata.summary = melt(xtabs(~target+bpcat,data=heartdata))
mydata.summary2 = mydata.summary[mydata.summary$target=="HD",]
mydata.summary2$percent <- mydata.summary2$value/melt(xtabs(~bpcat,data=heartdata))$value
p1<-ggplot(data=heartdata, aes(x=bpcat, fill=target)) +
geom_bar(stat="count", position=position_dodge()) +
labs(x="Resting blood pressure", y="Nuber of people")+
  scale_fill_manual(values = c("HD" = "darkblue", "NHD" = "orange")) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
p2<-ggplot(data=heartdata, aes(x=cp, fill=target)) +
geom_bar(stat="count", position=position_dodge())+
  labs(x="Chest Pain",
       y="Number of people") +
  scale_fill_manual(values = c("HD" = "darkblue", "NHD" = "orange")) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

grid.arrange(p1, p2, ncol = 2)
set.seed(111)
#trainIndex = sample(1:nrow(heartdata), 200)
#heartTrain = heartdata[trainIndex,]
#heartTest = heartdata[-trainIndex,]

intrain <- createDataPartition(y = heartdata$target, p= 0.7, list = FALSE)
heartTrain <- heartdata[intrain,]
heartTest <- heartdata[-intrain,]

tree1 = rpart(target~.,data=heartTrain,method="class")
#summary(tree1)
#plot(tree1)
#fancyRpartPlot(tree1, palettes=c("Blues", "Greens"))

tree1_pred = predict(tree1,heartTest, type="class")
table1 <-table(Predicted=tree1_pred,Observed=heartTest$target) #Accuracy:81.8%
kable(table1, "latex", booktabs = T, caption = "Confusion Matrix")%>%
  add_header_above(c("Predicted", "Observed" = 2)) %>%
kable_styling(latex_options = c("striped", "hold_position"))
set.seed(111)
kt=rpart(target~.,data=heartTrain,method="class", cp=0.001)
#printcp(kt)
ktp = prune(kt,cp=0.0011)
fancyRpartPlot(ktp, palettes=c("Purples", "Oranges"), sub = "")
rf_pred<-(predict(ktp,heartTest,type="class"))
ConfusionMatrix<-table(Predicted=rf_pred, Observed=heartTest$target)
#ConfusionMatrix
kable(ConfusionMatrix, "latex", booktabs = T, caption = "Confusion Matrix")%>%
  add_header_above(c("Predicted", "Observed" = 2)) %>%
kable_styling(latex_options = c("striped", "hold_position"))
Xy <- heartdata
x <- Xy[,1:14]
y <- Xy$target
#Setting the random seed for replication
set.seed(111)
```

```r
#setting up cross-validation
ctrl<- trainControl(method="repeatedcv", number = 10,
                          allowParallel=TRUE, classProbs = TRUE,
                     savePredictions = TRUE) #needed for ROC
metric="Accuracy"
dtree_fit <- train(target ~., data = Xy, method = "rpart",metric=metric,
                   trControl=ctrl,
                   tuneLength = 10,
                   parms=list(split='information'))
#plot(dtree_fit$finalModel)
fancyRpartPlot(dtree_fit$finalModel, palettes=c("Purples", "Oranges"),sub="")
#print(dtree_fit)#Accuracy=74%
dat8 <- data.frame("cp" = c(0.0000,0.0547,0.1094,0.1642), "Accuracy" = c(0.740,0.766,0.766,0.766))
xtb5 <-xtable(dat8,caption="Accuracy of caret decision tree ")
print(xtb5, type="latex")
#caret package with split data
dtree_fit <- train(target ~., data = heartTrain, method = "rpart",
                   parms = list(split = "information"),
                   trControl=ctrl,
                   tuneLength = 10)
prp(dtree_fit$finalModel, box.palette = "Reds", tweak = 1.2)

test_pred <- predict(dtree_fit, newdata = heartTest)
#confusionMatrix(test_pred, heartTest$target )#Accuracy 72.7%
ConfusionMatrix2<-table(Predicted=test_pred, Observed=heartTest$target)
ConfusionMatrix2

dat9 <- data.frame("Tuning parameters" = c("mtry","ntree"), "Description" = c("number of variables rand
xtb9 <-xtable(dat9,caption="Tuning parameters for Random Forest ")
print(xtb9, type="latex")
set.seed(111)
mtry <- sqrt(ncol(x))
metric <-"Accuracy"
rfctrl <- trainControl(method="repeatedcv", number=10, repeats=7, search ="random",
                     classProbs = TRUE,
                     savePredictions = TRUE) #needed for ROC
rforest_fit <- train(target ~., data = Xy, method = "rf", metric=metric, tuneLength=15,
                   trControl=rfctrl)
#print(rforest_fit) # Accuracy=81.8, ROC 0.9088771  Sens:0.7341444  Spec:0.8892857

t <- varImp(rforest_fit)$importance
ind <- rev(order(unlist(t)))
imp <- unlist(t)[ind]
var <- row.names(t)[ind]
g1 <- tibble(
   var = ordered(var, levels=var), #need ordered for Pareto
   imp = imp
 ) %>%
  ggplot(aes(x = var, y = imp)) +
  geom_bar(stat = "identity", fill="darkblue") +
  xlab("Variable") +
  ylab("Importance") +
  coord_flip()
```

```r
plot(rforest_fit)
print(g1)

dat12 <- data.frame("mtry" = c(1,2,4,8), "Accuracy" = c(0.815,0.800,0.784,0.777))

xtb12 <-xtable(dat12,caption="Accuracy for Random Forest")
print(xtb12, type="latex")
set.seed(111)
tunegrid <- expand.grid(.mtry=c(1:15))
metric <- "Accuracy"
grctrl <- trainControl(method="repeatedcv", number=10, repeats=7, search ="grid",
                     summaryFunction=twoClassSummary,
                     classProbs = TRUE,
                     savePredictions = TRUE) #needed for ROC
rforest_grid <- train(target ~., data = Xy, method = "rf", metric="ROC", tuneGrid= tunegrid,
                   trControl=grctrl)

print(rforest_grid) #Accuracy 81.5%
plot(rforest_grid)#mtry=1    0.9075108 Sens:0.7424647  Spec:0.8785714

train.cf <- train(target ~ .,   #cforest knows the outcome is binary (unlike rf)
                   data=heartTrain,
                   method="cforest",
                   trControl=ctrl)  #Note that importance not available here
#train.cf
dat13 <- data.frame("mtry" = c(2,11,21), "Accuracy" = c(0.822,0.788,0.779))

xtb13 <-xtable(dat13,caption="Accuracy for cForest")
print(xtb13, type="latex")
#obtaining class predictions
cf.classTest <-  predict(train.cf,
                         newdata = heartTest,
                          type="raw")
#computing confusion matrix
#cf3 <- confusionMatrix(heartTest$target,cf.classTest)#Accuracy 79%
Ccf3<-table(Predicted=cf.classTest, Observed=heartTest$target)
kable(Ccf3, "latex", booktabs = T, caption = "Confusion Matrix")%>%
  add_header_above(c("Predicted", "Observed" = 2)) %>%
kable_styling(latex_options = c("striped", "hold_position"))
train.gbm <- train(as.factor(target) ~ .,
                   data=heartTrain,
                   method="gbm",
                   verbose=F,
                   trControl=ctrl)
#train.gbm
dat14 <- data.frame("n.trees" = c(50,100,150,50), "Accuracy" = c(0.817,0.832,0.831,0.842))

xtb13 <-xtable(dat13,caption="Accuracy for Gradient boosting")
print(xtb13, type="latex")
#obtaining class predictions
gbm.classTest <-  predict(train.gbm,
                         newdata = heartTest,
                          type="raw")
```

```r
#confusionMatrix(heartTest$target,gbm.classTest)


gbm4<-table(Predicted=gbm.classTest, Observed=heartTest$target)
kable(Ccf3, "latex", booktabs = T, caption = "Confusion Matrix")%>%
  add_header_above(c("Predicted", "Observed" = 2)) %>%
kable_styling(latex_options = c("striped", "hold_position"))
rf.probs=predict(rforest_fit,
                 newdata=heartTest,
                 type="prob")
cf.probs=predict(train.cf,
                 newdata=heartTest,
                 type="prob")
gbm.probs=predict(train.gbm,
                 newdata=heartTest,
                 type="prob")
tree.probs=predict(ktp,
                 newdata=heartTest,
                 type="prob")
rocCurve.rf <- roc(heartTest$target,rf.probs[,"HD"])
rocCurve.cf <- roc(heartTest$target,cf.probs[,"HD"])
rocCurve.gbm <- roc(heartTest$target,gbm.probs[,"HD"])
rocCurve.tree <- roc(heartTest$target,tree.probs[,"HD"])
plot(rocCurve.tree,col=c(4))
plot(rocCurve.rf,add=TRUE,col=c(1)) # color black is rf
plot(rocCurve.cf,add=TRUE,col=c(2)) # color red is cforest
plot(rocCurve.gbm,add=TRUE,col=c(3)) # color green is gbm
legend(0.3,0.8,legend=c("Tree", "Random Forest", "cForest","Gradient Boosting "),
       col=c("blue", "black","red","green"), lty=1:1, cex=0.8)
library(xtable)
datafinal <- data.frame("Models" = c("Unpruned decision tree","Pruned decision tree","Random forest wit

xtbfinal <-xtable(datafinal,caption="Accuracy of all models")
print(xtbfinal, type="latex")
```