Applied Predictive Modeling - Predicting Heart Disease

Filipp Krasovsky & Rudy Fasano

6/10/2021

This report focuses on analyzing a collection of anonymized medical records (n=303) with the intent of designing a model that can accurately predict the incidence of heart disease given an array of biological factors such as age, sex, blood cholestrol, etc. The modeling question before us stands as follows: "Given some combination of medical information about a patient, is a patient more likely than not to contract heart disease?"

Data was collected from Kaggle (citation needed here) and contains a total of 303 data points with 13 predictors and a single response variable (output), which serves as a binary indicator of whether or not an individual has contracted heart disease. Because of the nature of our response variable, we have no choice but to frame the problem as a classification challenge rather than pure regression - that is, OLS and similar models are excluded, but we're still able to utilize alternative classifiers such as logistic regression, SVM, KNN, etc. As a preliminary note, our data does not have any missing values.

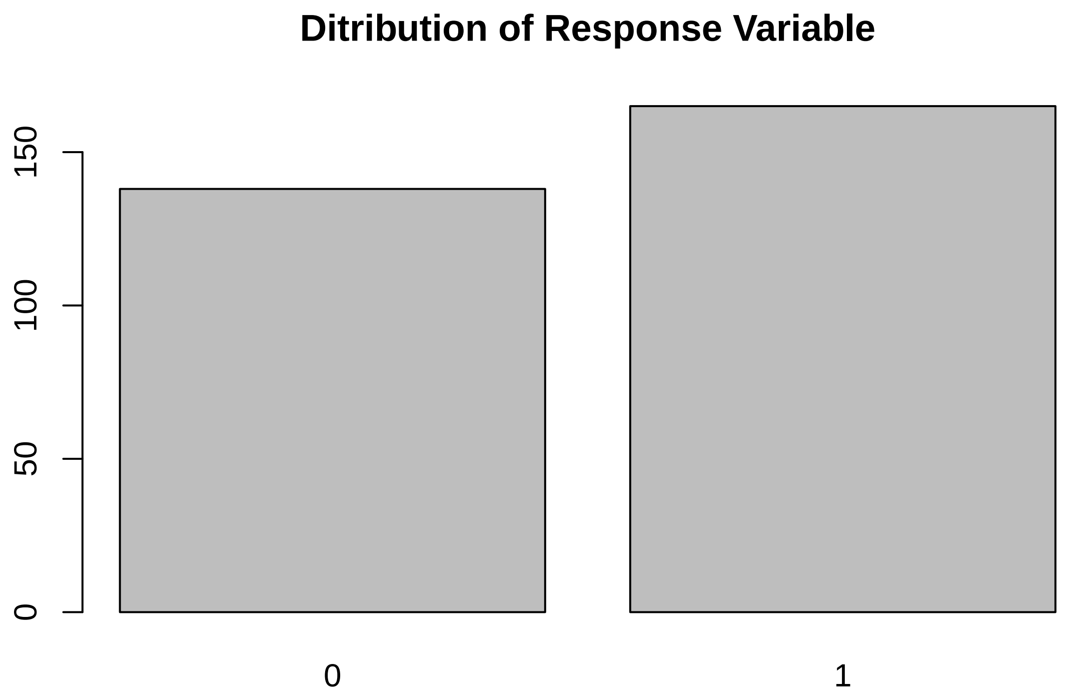
We begin with an overview of each variable's interpretation:

Age: A discrete ordinal variable which indicates the age of the patient. Sex: A binary categorical variable where 0=female and 1=male. cp: Chest Pain Type, categorical variable: 1 = typical angina, 2 = atypical angina, 3 = non-anginal pain, 4 = asymptomatic trtbps: resting blood pressure in mm Hg - continuous ratio variable. chol: cholestrol in mg/dl - continuous ratio variable fbs: a categorical variable where 1 indicates that fasting blood sugar > 120 mg/dl. restecg: resting electrocardiographic results where 0 = normal, 1 = wave abnormality, 2 = probably ventricular hypertrophy. thalachh: maximum heart rate achieved (ratio continuous variable) exng: an indicator variable where 1 indicates the presence of exercise induced angina. oldpeak: Previous Peak - a ratio continuous variable with no context available in the dataset. slp: slope - no context was available for this dataset. caa: number of major vessels. thall: the thal rate - a discrete ratio variable with no context provided.

```r
#load in our dataset
require(caret)
require(pROC)
require(ggplot2)
df = read.csv('heart.csv',TRUE)
df$output = as.factor(df$output)
df$cp = as.factor(df$cp)
df$exng = as.factor(df$exng)
df$thall = as.factor(df$thall)
df$caa = as.factor(df$caa)
df$restecg = as.factor(df$restecg)
df$sex = as.factor(df$sex)
df$fbs = as.factor(df$sex)
df$slp = as.factor(df$slp)
df$caa = as.factor(df$caa)
```

We begin by noting that there is no class imbalance in our dataset:

```r
#plot a histogram of the response variable
plot(df$output,main="Ditribution of Response Variable")
```

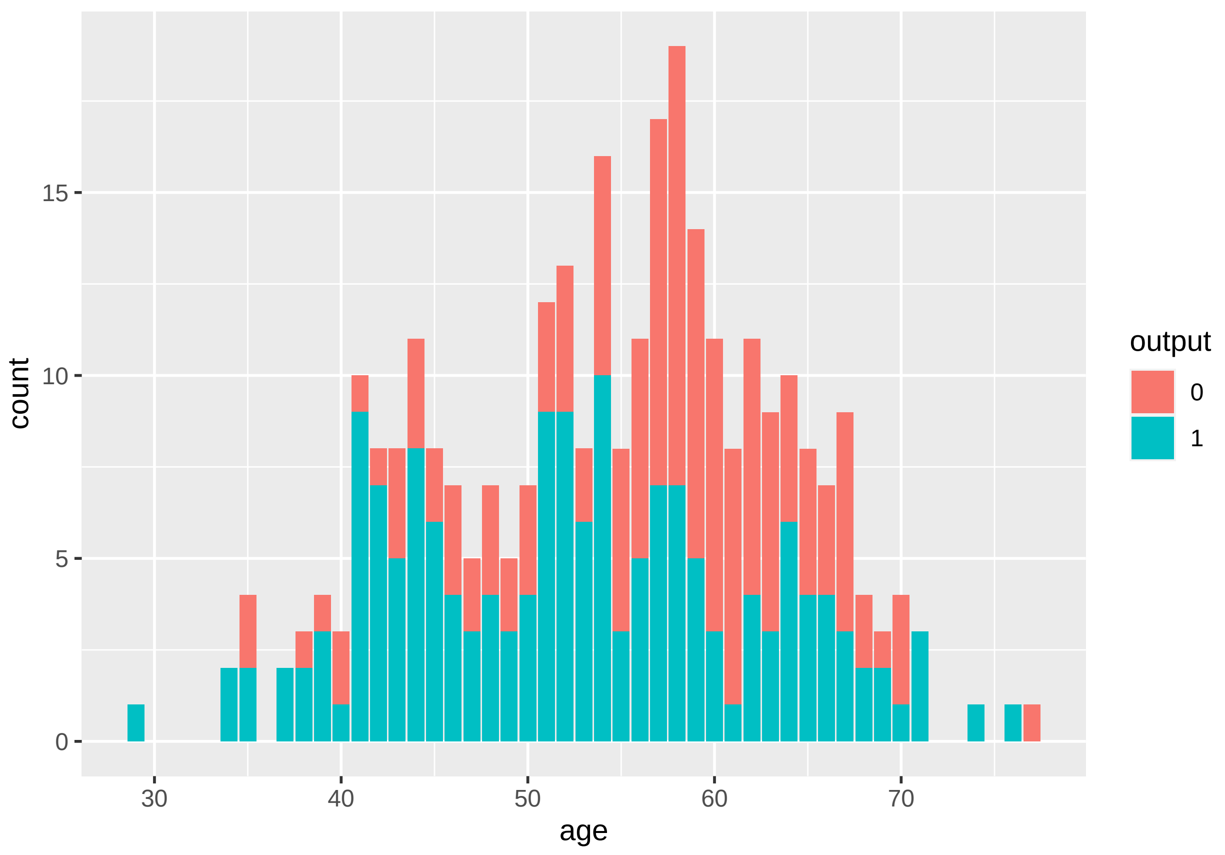


```r
#print distribution by percentage
print((table(df$output)/nrow(df))*100)
```

```
##
##        0        1
## 45.54455 54.45545
```

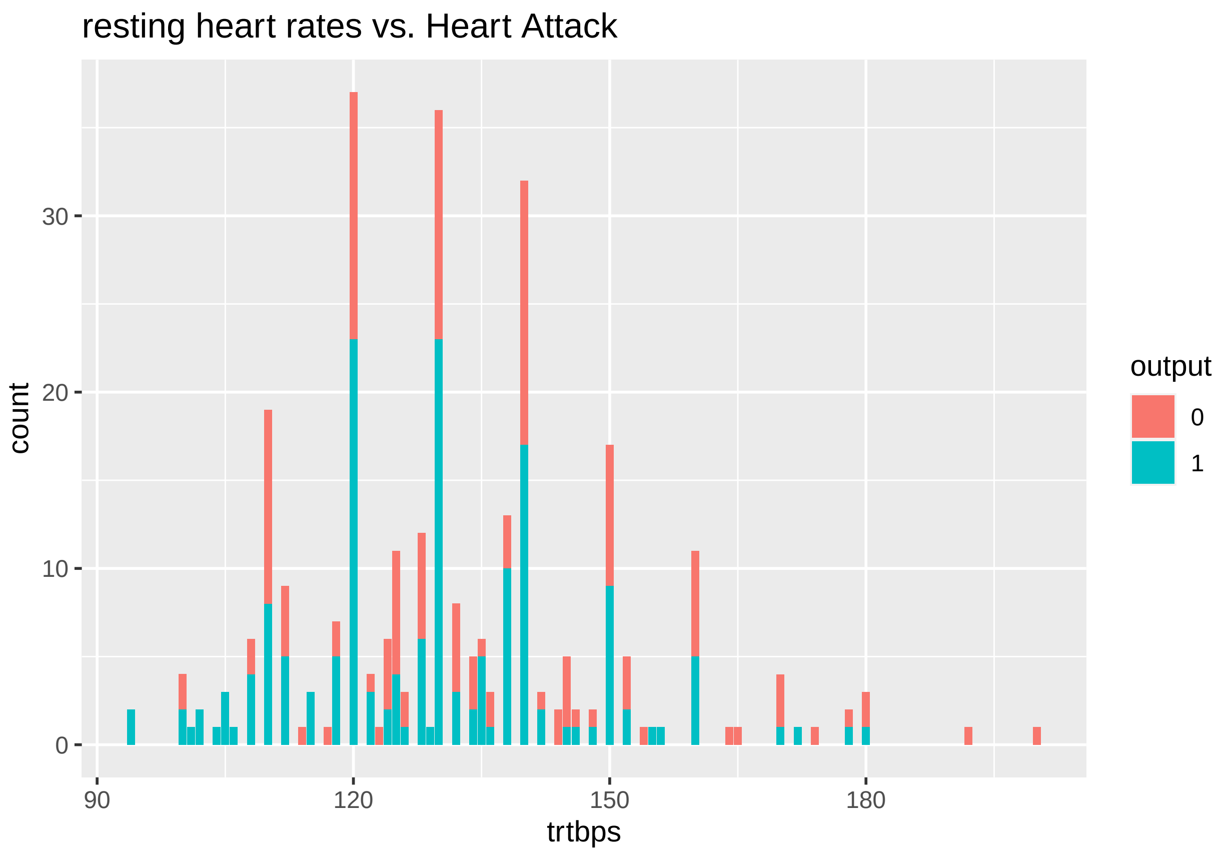
As demonstrated, we have a low class imbalance and probably will not need any sampling methods that require us to balance classes across samples, which allows us to move straight into k-fold cross validation, bagging, and bootstrapping. Of our predictor space, five variables are non-categorical, so we can proceed by examining their distributions relative to the response variable as well as any possible between-predictor relationships:

```
ggplot(df,aes(x=age,group=output,fill=output))+geom_bar()
```



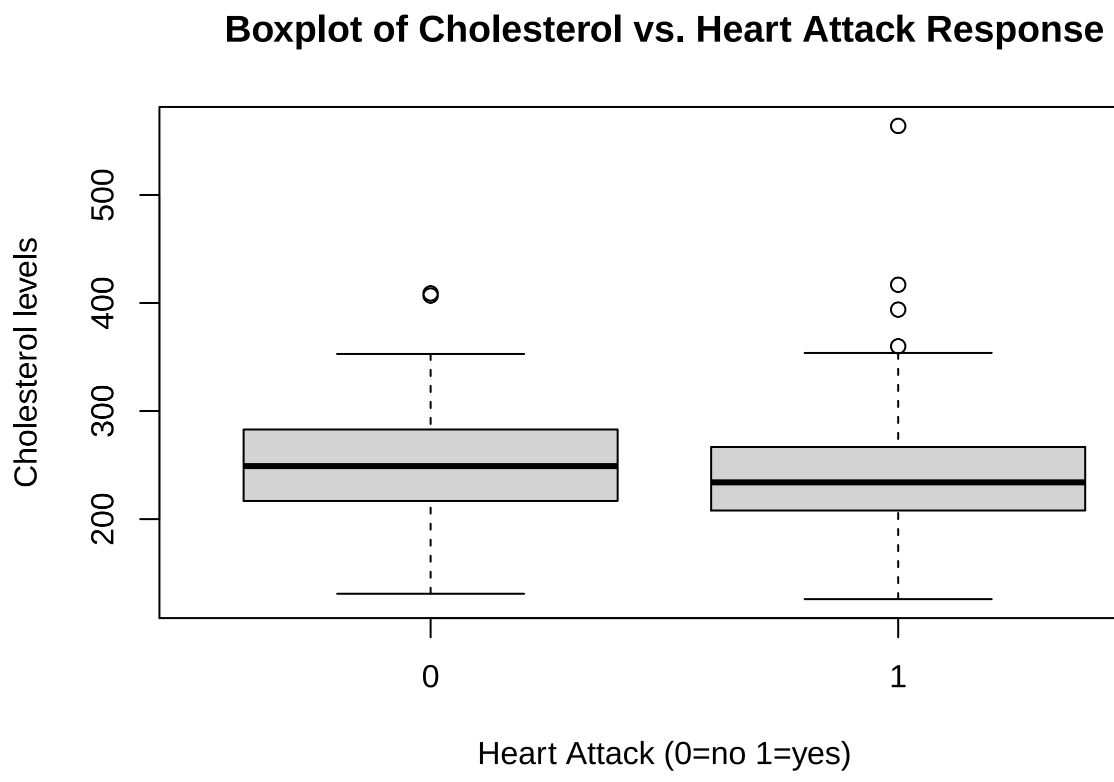
Starting with age, we observe that that the incidence of heart dominates samples with ages <= 55, while being more or less balanced for larger age values. When we observe this relationship for resting heart rates (trtbps), the most obvious face-value observation is that resting heart rate values smaller than 120 bps are disproportionately associated with the risk of heart attack (output=1):

```
ggplot(df,aes(x=trtbps,group=output,fill=output))+geom_bar(stat = 'count')+ggtitle("resting heart rates
```



Another intuitive hypothesis is that higher rates of cholesterol are strongly associated with a higher incidence of heart attack. For this examination, we can use a box and whisker plot. Our findings suggest that those samples with an incidence of heart attack had a smaller average cholesterol value but more outliers gravitating towards the higher end of cholesterol levels.

```
plot(df$output,df$chol,main="Boxplot of Cholesterol vs. Heart Attack Response",xlab="Heart Attack (0=no
```



However, given the motivation to explore this relationship further, we can consider a t-test to determine if the mean cholesterol levels are significantly different between those who did and did not have a heart attack:

```
response.1 <- which(df$output==1)
x1 = df[response.1,]$chol
x2 = df[-response.1,]$chol
#conduct a t test for difference of means
xtest = t.test(x1,x2)
print(xtest$p.value)
```

```
## [1] 0.1360182
```

Therefore, at a significance level of alpha = 0.05 that we do not have enough evidence to reject the null hypothesis that the difference in average cholesterol levels between patients who did and did not have a heart attack is zero.

Notable findings about the imbalance of heart attack patients, however, include the gender disparity. We first begin by observing the imbalance in data between men and women (0=women, 1=men):

```
tb.sex = (as.table(table(df$sex,df$output)))
print(tb.sex)
```

```
##
##       0   1
##   0  24  72
##   1 114  93
```

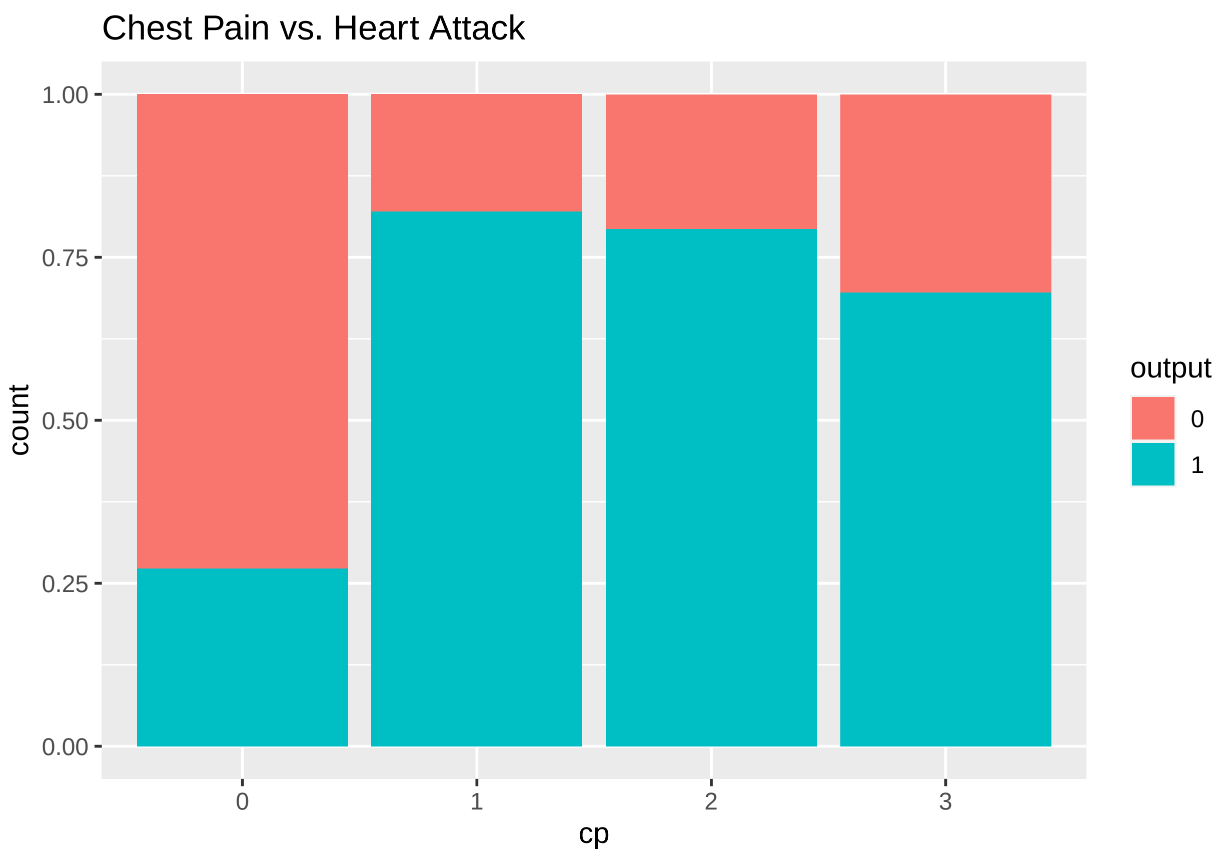
We find that while there are 96 female sample points, there are 207 male points, more than twice the number of female participants. Notwithstanding this, we find that the ratio of women who get heart attacks is much higher than that of men:

```
print(prop.table(tb.sex,margin=1))
```

```
##
##             0         1
##   0 0.2500000 0.7500000
##   1 0.5507246 0.4492754
```

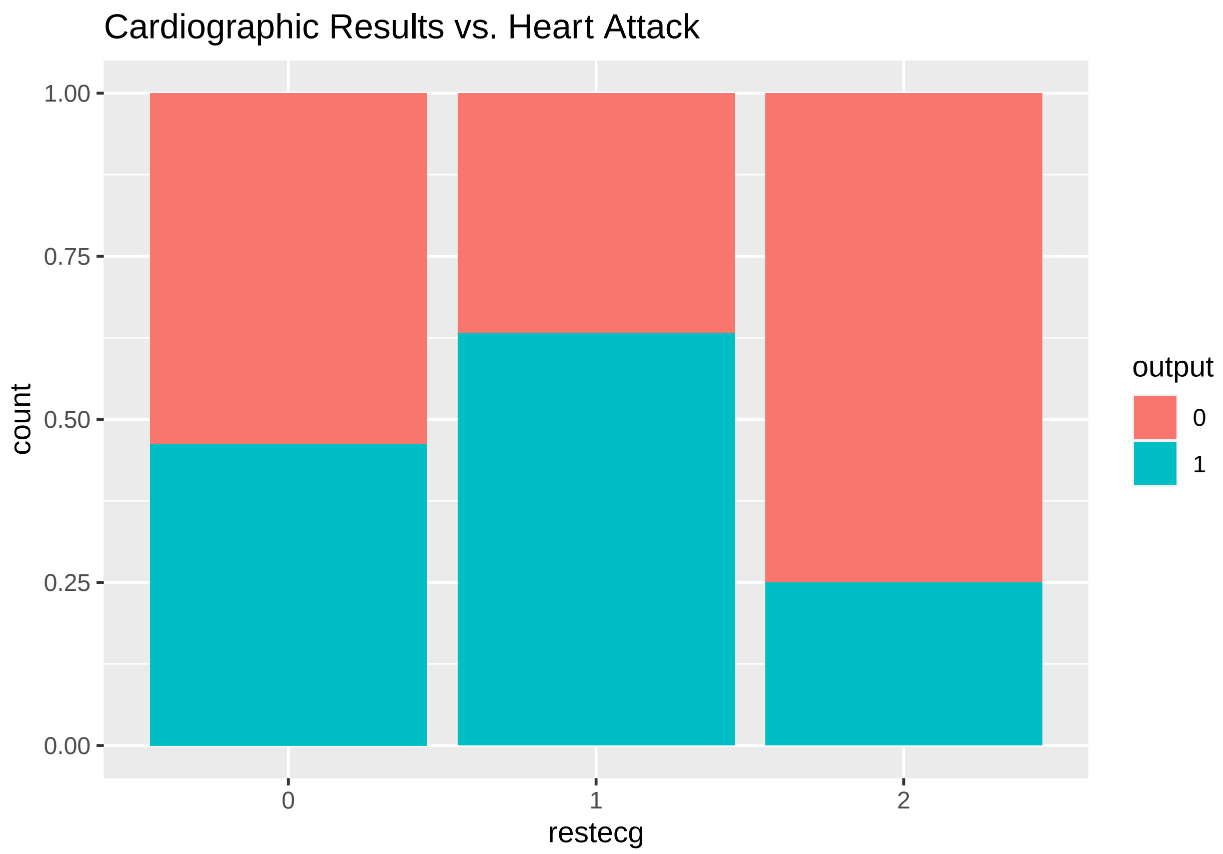
Other notable findings include the observation that heart attacks are significantly less likely for the lowest possible cp level. Although the description of this dataset labels data values 1-4 as increasing levels of angina pain, there doesn't seem to be any presence of values higher than 3. Consequently, we have to assume that the lowest level of the cp column is either the absence of angina entirely or an indicator of the presence of normal angina. Either way, the intuition still stands that non-acute/non-irregular levels of Angina (as well as the absence of angina entirely) are less likely to be associated with the incidence of a heart attack:

```
ggplot(df,aes(x=cp,group=output,fill=output))+geom_bar(stat='count', position = 'fill')+ggtitle("Chest
```



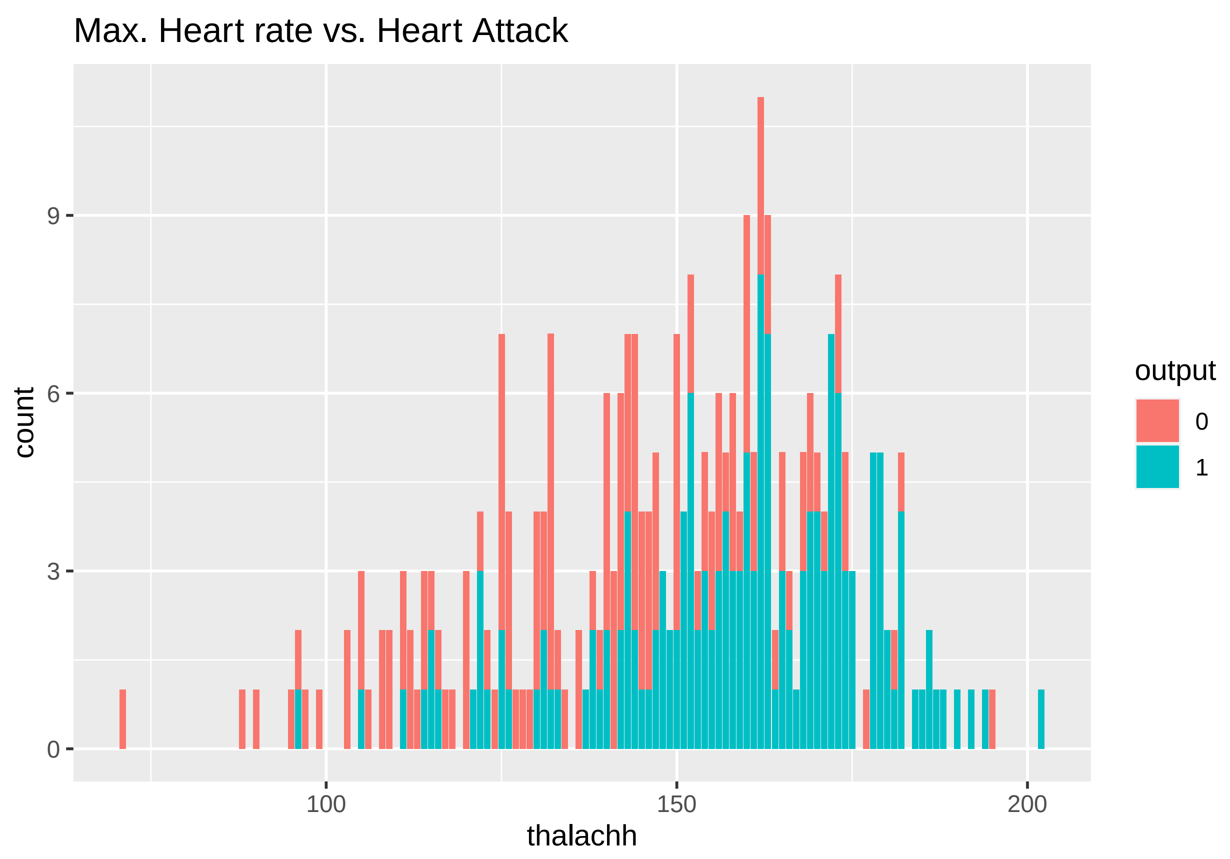
Detectiong of cardiographic results from the restecg variable suggest that the incidence of wave abnormality is significantly more associated with the risk of heart attack than normal results, and significantly moreso compared to incidences of ventricular hypertrophy. While this teams lacks the medical background to understand the implications of hypertrophy or wave abnormality, we can still understand that these conditions are abnormal, and so, intuitively affect heart functions.

```
ggplot(df,aes(x=restecg,group=output,fill=output))+geom_bar(stat='count', position = 'fill')+ggtitle("Ca
```



Carrying on in the same vein, our team also identified an intuitive relationship between the response variable and the thalachh variable, which corresponds to the maximum heart rate achieved. We find that higher maximum heart rates correspond more frequently to a higher incidence of heart attack:

```
ggplot(df,aes(x=thalachh,group=output,fill=output))+geom_bar()+ggtitle("Max. Heart rate vs. Heart Attac
```



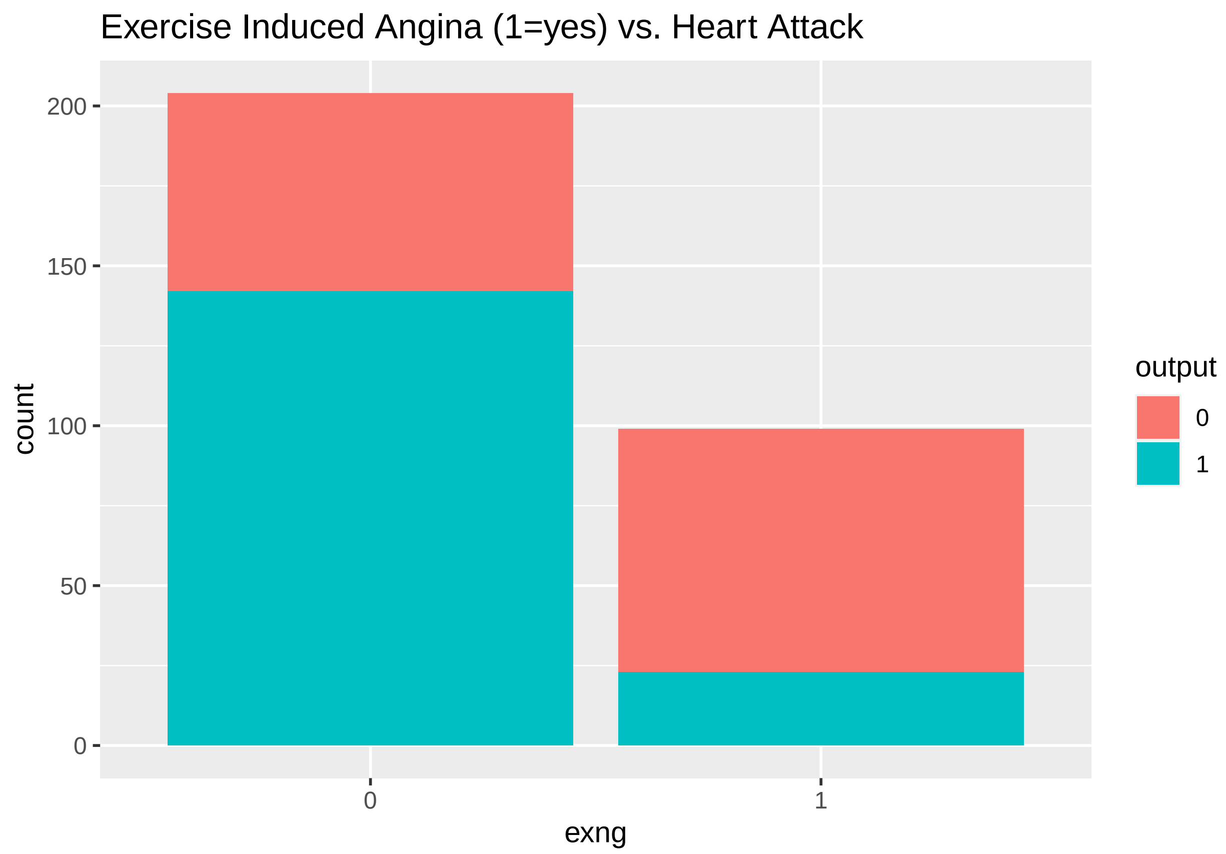
We further confirm this intuition with a t-test at a significance level of 0.05:

```
x = subset(df,output==1)$thalachh
y = subset(df,output==0)$thalachh
thaltest = t.test(x,y)
print(thaltest)
```

```
##
##  Welch Two Sample t-test
##
## data:  x and y
## t = 7.953, df = 269.9, p-value = 5.019e-14
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  14.57132 24.15912
## sample estimates:
## mean of x mean of y
##  158.4667  139.1014
```

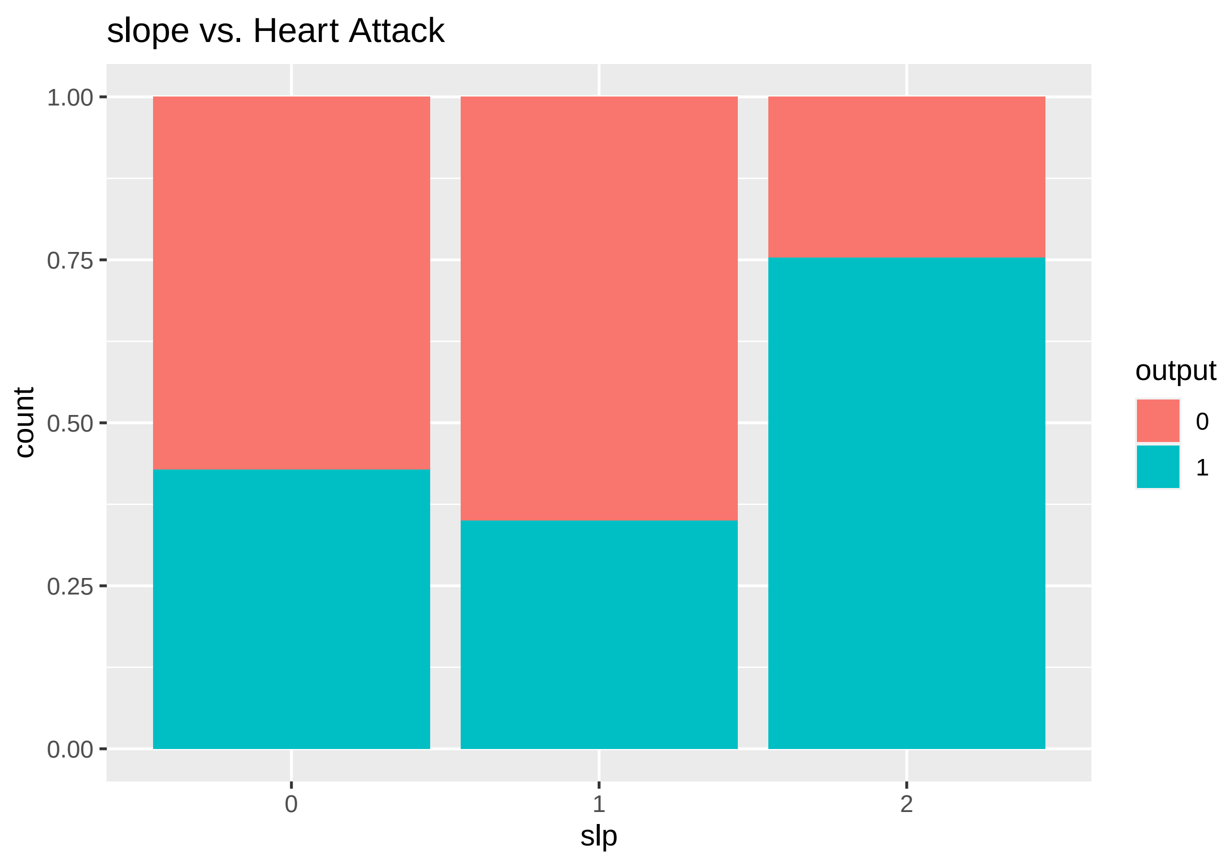
The mean max. heart rate for those who experienced a heart attack is significantly higher than their counterparts, confirming our intuition. Conversely, the data supports the claim that individuals who suffer from exercise induced angina are less likely to suffer a heart attack. While this was largely surprising during EDA, an intuitive explanation might include that individuals suffering from an exercise-induced condition may be more resistant to a heart attack based on their athletic activity:

```
ggplot(df,aes(x=exng,group=output,fill=output))+geom_bar()+ggtitle("Exercise Induced Angina (1=yes) vs.
```



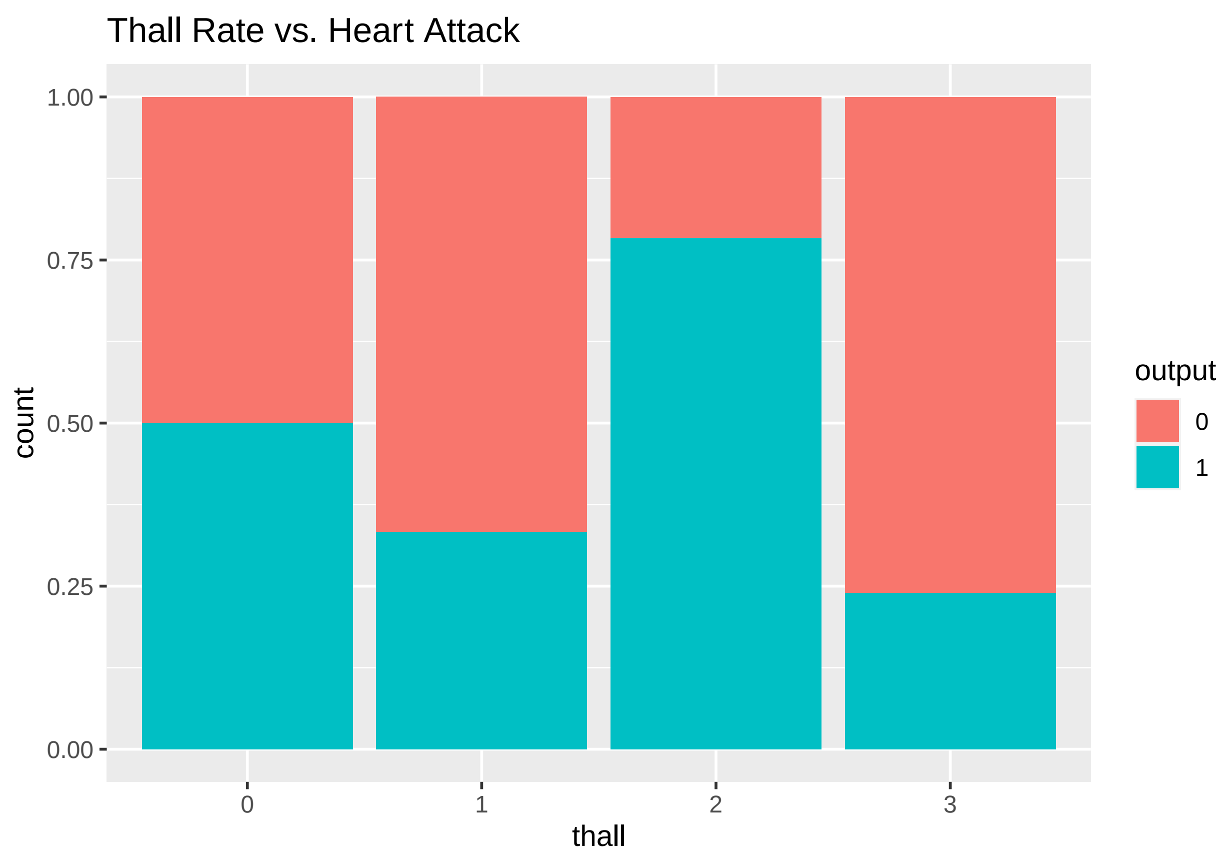
Furthermore, while we were unable to determine the context for the slope variable (slp) in the dataset, we did find that it significantly separates heart attack incidence based on its three unique values (0-2). In particular, we find that patients with a slope of 2 are significantly more likely to experience a heart attack:

```
df$slp = as.factor(df$slp)
ggplot(df,aes(x=slp,group=output,fill=output))+geom_bar(stat='count',position='fill')+ggtitle("slope vs
```



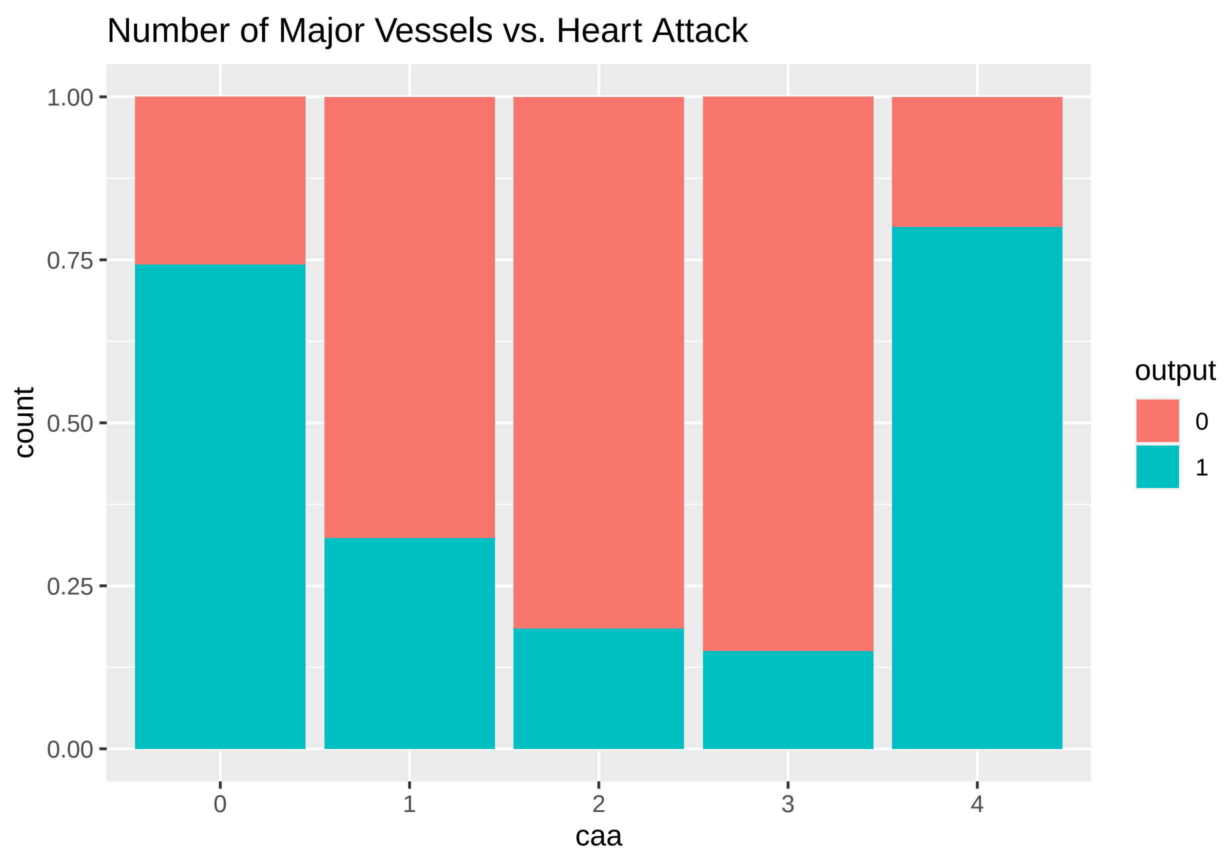
Similarly, while we found no background explanation on how to interpret the thall rate, we did find that those patients with a value of 0 and 2 had a significantly higher association with heart attack than values of 1 and 3:

```
ggplot(df,aes(x=thall,group=output,fill=output))+geom_bar(stat='count',position='fill')+ggtitle("Thall 
```



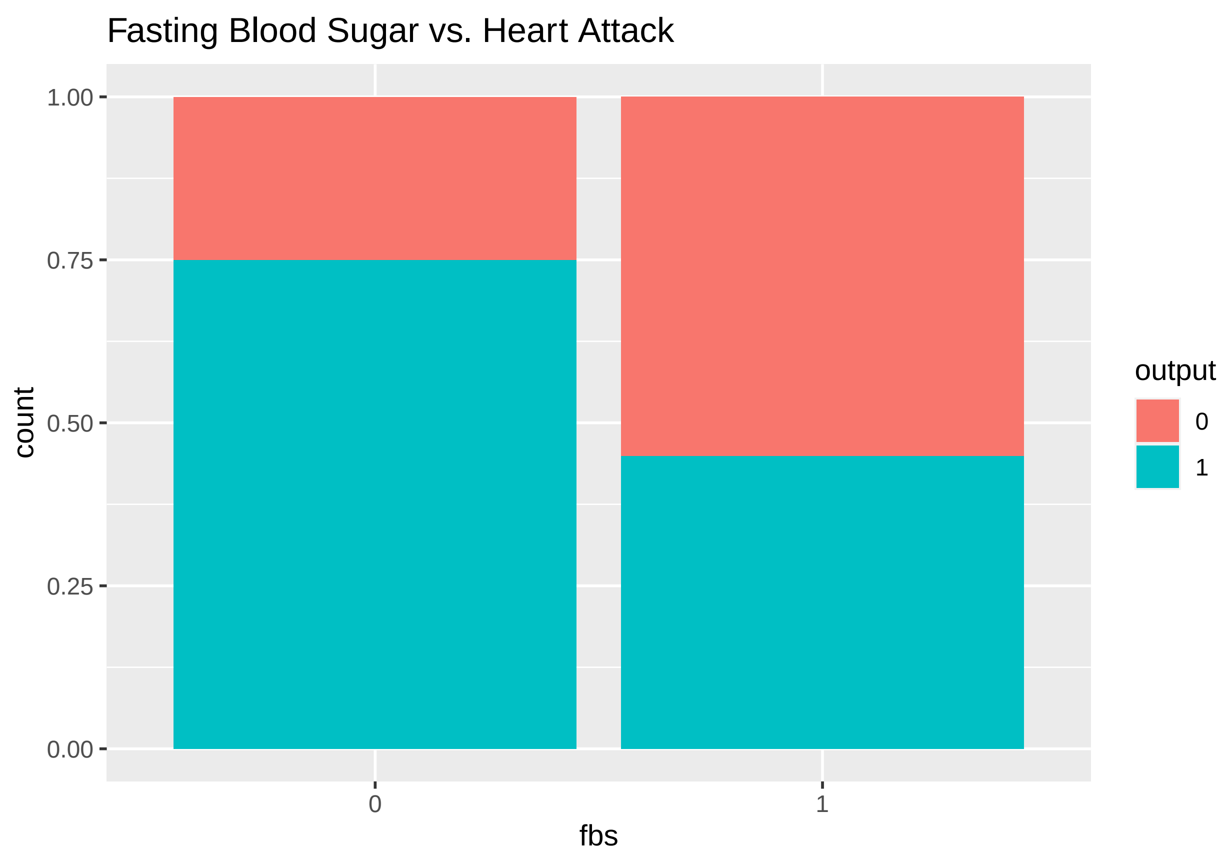
The number of major vessels also seems to have a significant effect on separating our response variable - individuals with zero and four major vessels are disproportionately associated with the incidence of heart attack:

```
ggplot(df,aes(x=caa,group=output,fill=output))+geom_bar(stat='count',position='fill')+ggtitle("Number o
```



Finally, we find that individuals with a lower resting blood sugar seem to have a higher incidence of heart attacks.

```
ggplot(df,aes(x=fbs,group=output,fill=output))+geom_bar(stat='count', position = 'fill')+ggtitle("Fasti
```



Moving forward with data pre-processing, we can center and scale our continuous predictors: age, trtbps, chol, thalachh, and oldpeak.

Because our predictor space is only 13 variables and n = 303, we can afford a 90-10 split between our training and testing set as well. We will also engage in a 10-fold cross validation for model training.

```
cont_vars = c("age","trtbps","chol","thalachh","oldpeak")
df.cont <- df[,cont_vars]
print(findCorrelation(cor(df.cont))) #no correlated predictors for our continuous variables.
```

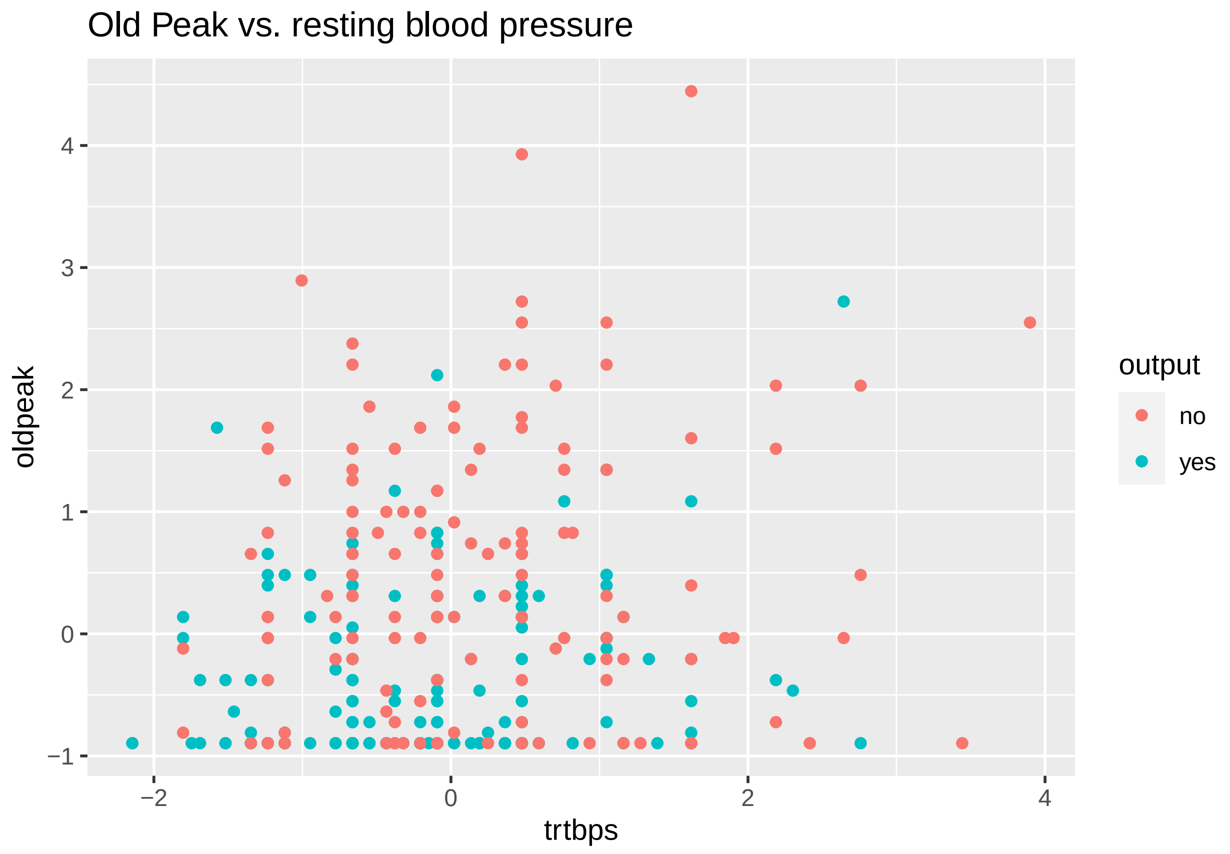
```
## integer(0)
```

```
df$output = ifelse(df$output==1,"yes","no")

#center and scale the continuous variables
df = predict(preProcess(df,method=c("center","scale")),df)

#partition 90-10
set.seed(10)
trainingRows <- createDataPartition(df$output,p=0.9,list=FALSE)

predictors = subset(df,select=-c(output))
response   = data.frame(df$output)
names(response) = "response"

predictors.train = predictors[trainingRows,]
```

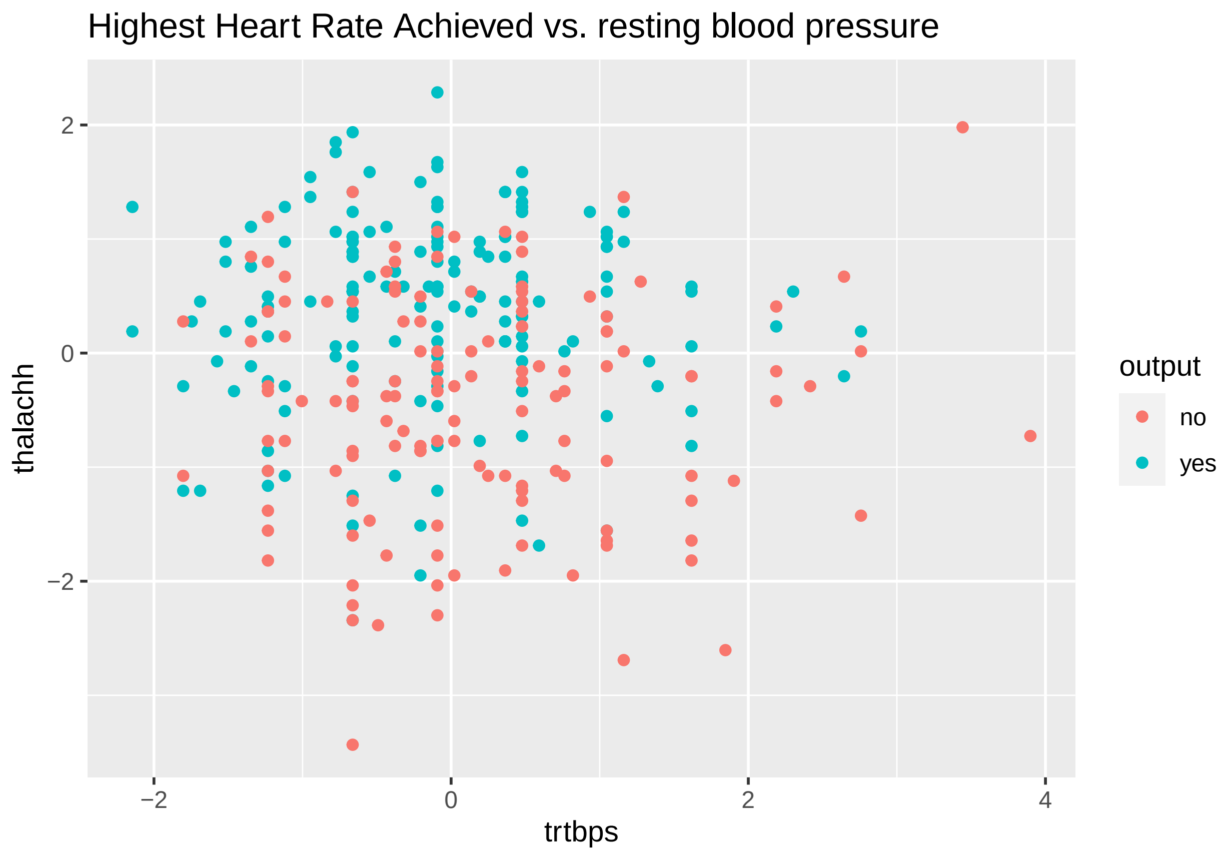
```
response.train    = response[trainingRows]

predictors.test   = predictors[-trainingRows,]
response.test     = response[-trainingRows]

reduced.train     = df.cont[trainingRows,]
reduced.test      = df.cont[-trainingRows,]
```

Because this is a classification challenge, we can consider classification models only (ie Logistc regression, QDA/LDA, Knn, etc.) We can create a hunch for whether or not the business problem is linearly separable by looking at scatter plots of our continuous data:

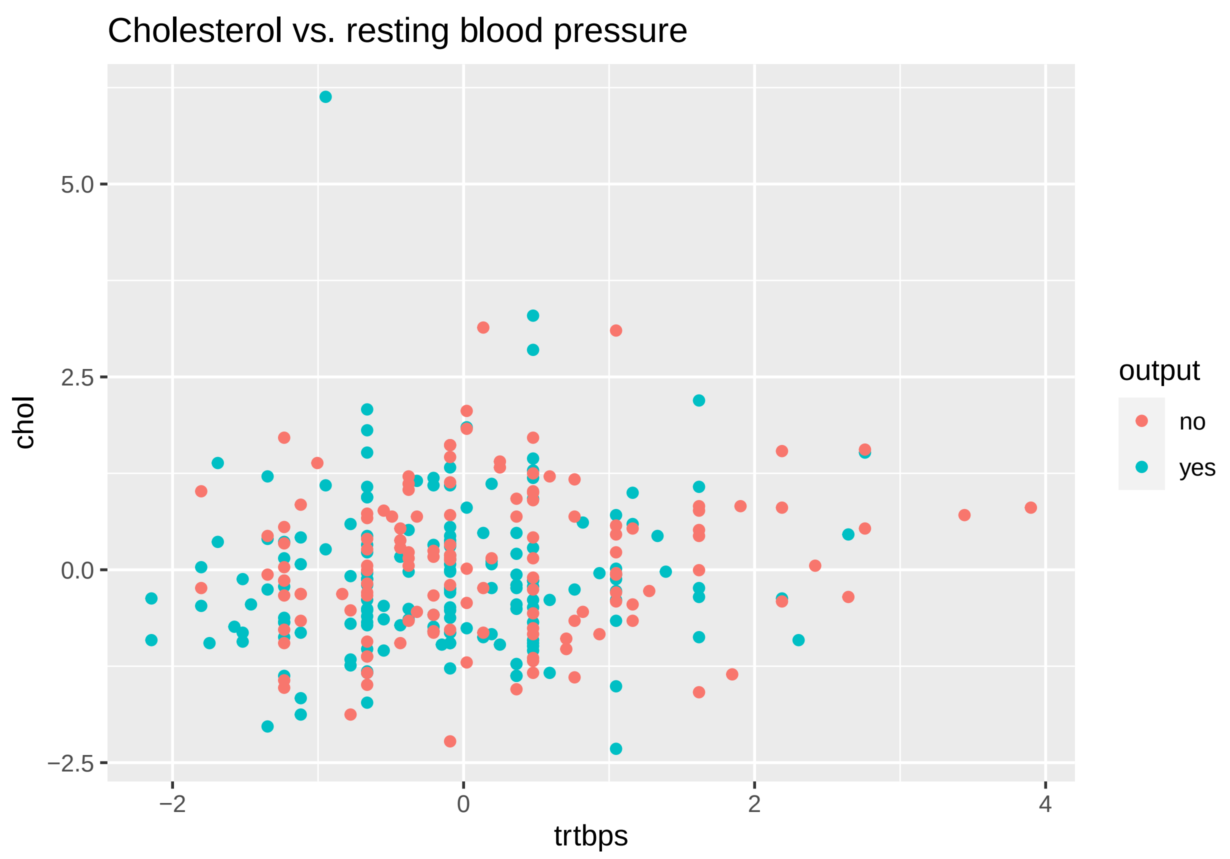
```
ggplot(df,aes(x=trtbps,y=oldpeak,group=output))+geom_point(aes(color=output))+ggtitle("Old Peak vs. res
```



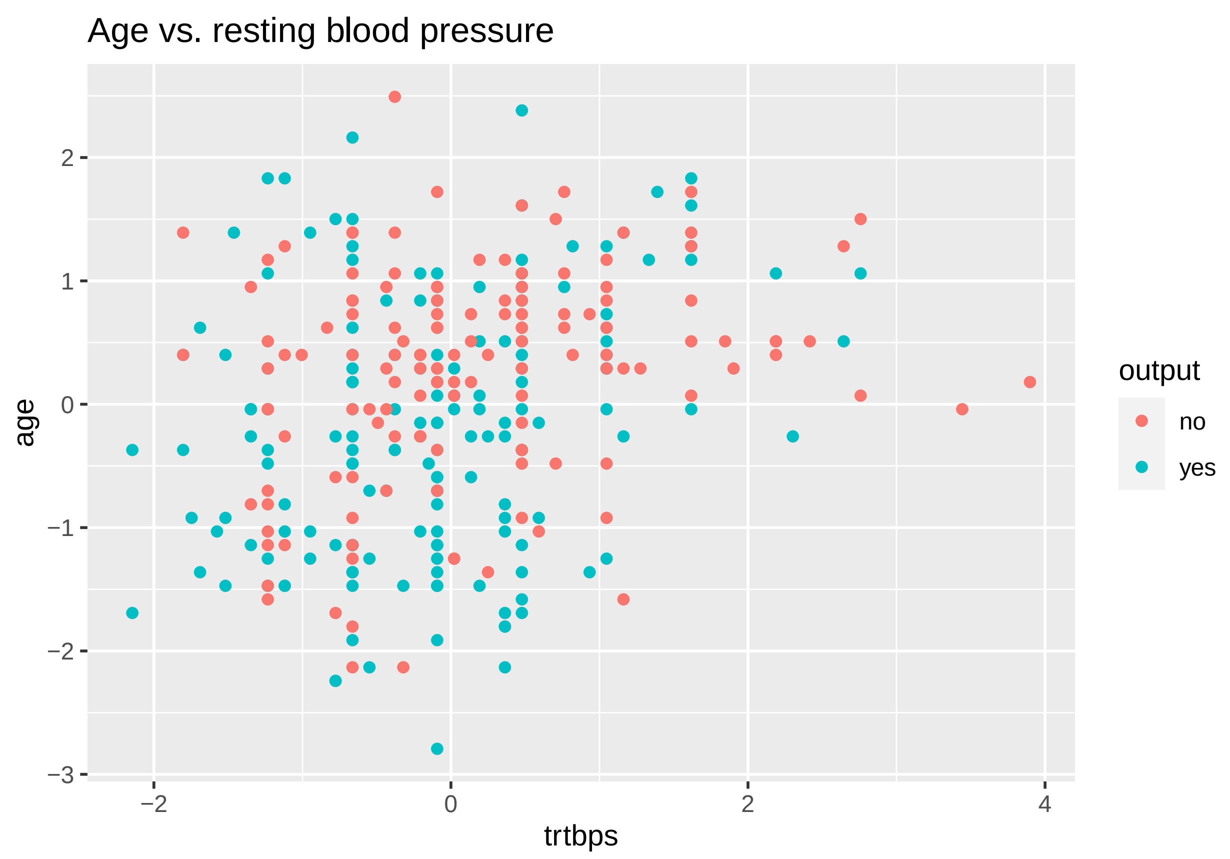
```
ggplot(df,aes(x=trtbps,y=thalachh,group=output))+geom_point(aes(color=output))+ggtitle("Highest Heart Ra
```



```
ggplot(df,aes(x=trtbps,y=chol,group=output))+geom_point(aes(color=output))+ggtitle("Cholesterol vs. res
```



```
ggplot(df,aes(x=trtbps,y=age,group=output))+geom_point(aes(color=output))+ggtitle("Age vs. resting blood
```



Based on a preliminary assessment, it seems unlikely that the problem would be linearly separable between the two response values. From here, we would favor nonlinear separation methods like logistic regression, QDA, NN, SVM, KNN, NSC, and Random Forests. Given that only five our predictors are numeric, we would have to limit the application of qda/svm/etc to only those predictors.

```
ctrl <- trainControl(summaryFunction = twoClassSummary, classProbs = TRUE)
knngrid <- data.frame(.k=c(1:40))

response.train = as.factor(response.train)
response.test =  as.factor(response.test)

#KNN
set.seed(476)
knnfit <- train(predictors.train, as.factor(response.train),
method = "knn",
metric = "ROC",
tuneGrid = knngrid,
trControl = ctrl)

knn.predict <- predict(knnfit,predictors.test,type="prob")
knn.roc <- roc(predictor = knn.predict$yes,response=response.test,levels=rev(levels(response.test)))
print(knn.roc)
```

```
##
## Call:
## roc.default(response = response.test, predictor = knn.predict$yes,      levels = rev(levels(response.
##
## Data: knn.predict$yes in 16 controls (response.test yes) > 13 cases (response.test no).
## Area under the curve: 0.9495
```

Next, we move on to our SVM model.

```
#SVM - set tuning param grid
set.seed(600)
library(kernlab)
sigmarange <- sigest(as.matrix(reduced.train))
svmgrid    <- expand.grid(.sigma = sigmarange[1],.C = 2 ** (seq(-4,4)))

#randomize, train, predict
set.seed(601)
svmfit <- train(x=reduced.train,y=response.train,method="svmRadial",tuneGrid = svmgrid,metric="ROC", fi
svm.predict <- predict(svmfit,reduced.test,type='prob')
svm.roc <-  roc(predictor = svm.predict$yes,response=response.test,levels=rev(levels(response.test)))
print(svm.roc)
```

```
##
## Call:
## roc.default(response = response.test, predictor = svm.predict$yes,     levels = rev(levels(response.
##
## Data: svm.predict$yes in 16 controls (response.test yes) > 13 cases (response.test no).
## Area under the curve: 0.8365
```

Given our heavy dependence on categorical variable, it’s intuitive that our SVM model did not perform as well as expected.

```
#NSC
set.seed(529)
nscGrid <- data.frame(.threshold = 0:25)
nscTuned <- train(x = reduced.train,y = response.train,method = "pam",tuneGrid = nscGrid,metric= "ROC",
```

```
## 11111111111111111111111111
```

```
nsc.predict <- predict(nscTuned,reduced.test,type='prob')
nsc.roc <- roc(predictor=nsc.predict$yes,response=response.test,levels=rev(levels(response.test)))
print(nsc.roc)
```

```
##
## Call:
## roc.default(response = response.test, predictor = nsc.predict$yes,     levels = rev(levels(response.
##
## Data: nsc.predict$yes in 16 controls (response.test yes) > 13 cases (response.test no).
## Area under the curve: 0.7115
```

Next, we consider logistic regression both with continuous only predictors and categorical predictors.

```r
#glm using only continuous vars
set.seed(289)
glmTuned <- train(x=reduced.train,y=response.train,method="glm",metric="ROC",trControl=ctrl)
glm.predict <- predict(glmTuned,reduced.test,type='prob')
glm.roc <- roc(predictor=glm.predict$yes,response=response.test,levels=rev(levels(response.test)))
print(glm.roc)
```

```
##
## Call:
## roc.default(response = response.test, predictor = glm.predict$yes,     levels = rev(levels(response.t
##
## Data: glm.predict$yes in 16 controls (response.test yes) > 13 cases (response.test no).
## Area under the curve: 0.8365
```

```r
#glm using the entire set
set.seed(628)
train <- cbind(predictors.train,response.train)
train$response.train=as.factor(train$response.train)
glm2 <- glm(response.train ~ .,data=train,family=binomial)
sucProb <-1 - predict(glm2,newdata=predictors.test,type="response")
glm2.roc = roc(predictor = sucProb,response=response.test,levels=rev(levels(response.test)))
print(glm2.roc)
```

```
##
## Call:
## roc.default(response = response.test, predictor = sucProb, levels = rev(levels(response.test)))
##
## Data: sucProb in 16 controls (response.test yes) < 13 cases (response.test no).
## Area under the curve: 0.9663
```

We now explore NN:

```r
#NN
set.seed(1028)
nnetgrid = expand.grid(.size = 5:10, .decay  = c(0,.1,1,2))
maxsize = max(nnetgrid$.size)
numwts   = (maxsize *(length(predictors.train)+1) + maxsize + 1)
nnetfit = train(x=predictors.train,y=response.train,method="nnet",metric="ROC",preProcess = c("spatialS

#predict, calculate roc
nnetfit.pred = predict(nnetfit,predictors.test,type='prob')
nnetfit.roc = roc(predictor=nnetfit.pred$yes,response=response.test,levels=rev(levels(response.test)))
print(nnetfit.roc)
```

```
##
## Call:
## roc.default(response = response.test, predictor = nnetfit.pred$yes,     levels = rev(levels(response
##
## Data: nnetfit.pred$yes in 16 controls (response.test yes) > 13 cases (response.test no).
## Area under the curve: 0.9231
```

Our Neural Net also performs very well, but faces technical limitations in an R setting. We now consider RF implementation:

```
#RF
set.seed(972)
library(randomForest)
rfmodel<- train(x=predictors.train,y=response.train,method="rf",metric="ROC",trace=FALSE,trControl=ctrl
rfpredict <- predict(rfmodel,predictors.test,type='prob')
rf.roc <- roc(predictor=rfpredict$yes,response=response.test,levels=rev(levels(response.test)))
print(rf.roc)
```

```
##
## Call:
## roc.default(response = response.test, predictor = rfpredict$yes,     levels = rev(levels(response.te
##
## Data: rfpredict$yes in 16 controls (response.test yes) > 13 cases (response.test no).
## Area under the curve: 0.9712
```

Along the same vein of decision tree use, we can also implement gbm:

```
#Boosted trees
set.seed(2055)
gbmGrid <- expand.grid(.interaction.depth = seq(1, 7, by = 2), .n.trees = seq(100, 1000, by = 100), .sh
gbmTune <- train(predictors.train, response.train, method = "gbm", metric="ROC",tuneGrid = gbmGrid, ver
gbm.predict = predict(gbmTune,predictors.test,type='prob')
gbm.roc <- roc(predictor=gbm.predict$yes,response=response.test,levels=rev(levels(response.test)))
print(gbm.roc)
```

```
##
## Call:
## roc.default(response = response.test, predictor = gbm.predict$yes,     levels = rev(levels(response.
##
## Data: gbm.predict$yes in 16 controls (response.test yes) > 13 cases (response.test no).
## Area under the curve: 0.9471
```

Finally, we print the optimal tuning for each model:

```
#print tuning params for each model except glm
print(gbmTune$bestTune)
```

```
##    n.trees interaction.depth shrinkage n.minobsinnode
## 10    1000                 1      0.01             30
```

```
print(knnfit$bestTune)
```

```
##     k
## 40 40
```

```
print(nnetfit$bestTune)
```

```
##   size decay
## 3    5     1
```

```
print(nscTuned$bestTune)
```

```
##   threshold
## 1         0
```

```
print(rfmodel$bestTune)
```

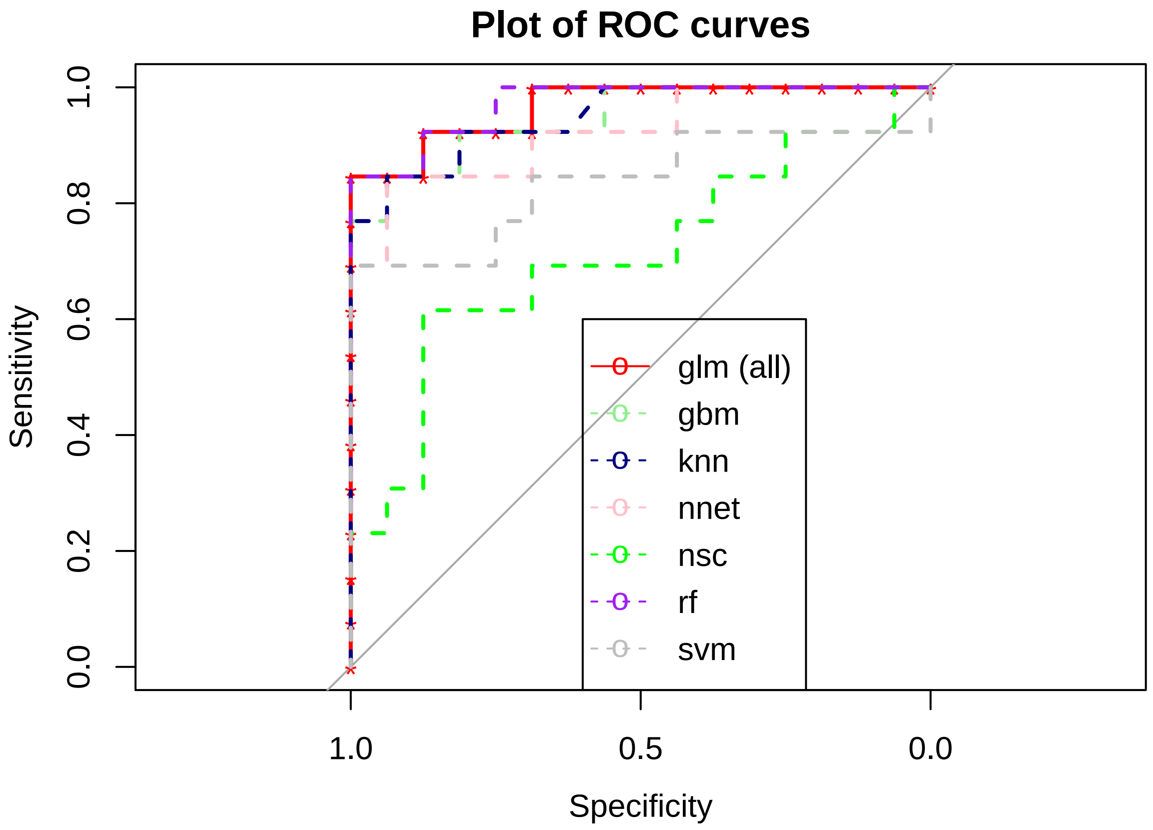
```
##   mtry
## 1    2
```

```
print(svmfit$bestTune)
```

```
##       sigma      C
## 1 0.0541278 0.0625
```

We plot the ROC curves for our models together to determine which model best satisfied our ROC expectations:

```
#plot ROC curves for each model
plot(glm2.roc,col='red',main="Plot of ROC curves",lty=1,type="o",pch="*")
lines(gbm.roc,col='lightgreen',lty=2)
lines(knn.roc,col='navyblue',lty=2)
lines(nnetfit.roc,col='pink',lty=2)
lines(nsc.roc,col='green',lty=2)
lines(rf.roc,col='purple',lty=2)
lines(svm.roc,col='gray',lty=2)
legend(0.6,0.6,legend=c("glm (all)","gbm","knn","nnet","nsc","rf","svm"), col=c("red","lightgreen","navy
```


Plot of ROC curves
Sensitivity
Specificity
1.0
0.8
0.6
0.4
0.2
0.0
1.0
0.5
0.0
glm (all)
gbm
knn
nnet
nsc
rf
svm