

# CS 221 Project Progress R Code

This R-markdown file contains the R-code for the project report of my CS 221 project.

```
library(caTools)
library(caret)

## Loading required package: lattice
## Loading required package: ggplot2
library(e1071)
library(MASS)
library(dplyr)

##
## Attaching package: 'dplyr'

## The following object is masked from 'package:MASS':
##
##      select

## The following objects are masked from 'package:stats':
##
##      filter, lag

## The following objects are masked from 'package:base':
##
##      intersect, setdiff, setequal, union

library(ggplot2)
library(naniar)
#library(randomForest)
```

This “convert” helper function is due to Brigitte Mueller, Predicting Heart Disease UCI. I’m borrowing this helper here to change the data type of the Cleveland data read from the website.

```
convert = function(obj,types){
  for (i in 1:length(obj)){
    FUN = switch(types[i],character = as.character,
                  numeric = as.numeric,
                  factor = as.factor)

    obj[,i] = FUN(obj[,i])
  }
  obj
}
```

The heart disease data are directly read from the UCI website.

```
cleveland.data = read.csv("https://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/processed.v1.data")
va.data = read.csv("http://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/processed.v2.data")
switzerland.data = read.csv("http://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/processed.v3.data")
hungarian.data = read.csv("http://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/processed.v4.data")
df_list <- list(cleveland.data, va.data, switzerland.data, hungarian.data)
combined.data = Reduce(function(x, y) merge(x, y, all=TRUE), df_list, accumulate=FALSE)
names(combined.data) = c("age", "sex", "cp", "trestbps", "chol", "fbs", "restecg",
                        "thalach", "exang", "oldpeak", "slope", "ca", "thal", "status")
combined.data$status[combined.data$status > 0] = 1
```

```

chclass = c("numeric","factor","factor","numeric","numeric","factor","factor","numeric","factor","numeric")
combined.data = convert(combined.data,chclass)
levels(combined.data$status) = c("healthy", "disease")
levels(combined.data$sex) = c("female", "male")
summary(combined.data)

```

```

##      age      sex      cp      trestbps      chol
##  Min.   :28.00  female:194  1: 46  Min.    : 0.0  Min.    : 0.0
##  1st Qu.:47.00  male  :726  2:174  1st Qu.:120.0  1st Qu.:175.0
##  Median :54.00                3:204  Median :130.0  Median :223.0
##  Mean   :53.51                4:496  Mean   :132.1  Mean   :199.1
##  3rd Qu.:60.00                3rd Qu.:140.0  3rd Qu.:268.0
##  Max.   :77.00                Max.    :200.0  Max.    :603.0
##                                     NA's   :59    NA's   :30
##      fbs      restecg      thalach      exang      oldpeak
##  0   :692    0   :551  Min.    : 60.0    0   :528  Min.    : -2.6000
##  1   :138    1   :179  1st Qu.:120.0    1   :337  1st Qu.: 0.0000
##  NA's: 90    2   :188  Median :140.0   NA's: 55  Median : 0.5000
##                NA's: 2  Mean    :137.5                Mean    : 0.8788
##                3rd Qu.:157.0                3rd Qu.: 1.5000
##                Max.    :202.0                Max.    : 6.2000
##                NA's    :55                NA's    :62
##      slope      ca      thal      status
##  1   :203    0   :181  3   :196  healthy:411
##  2   :345    1   : 67  6   : 46  disease:509
##  3   : 63    2   : 41  7   :192
##  NA's:309    3   : 20  NA's:486
##                NA's:611
##
##

```

```
str(combined.data)
```

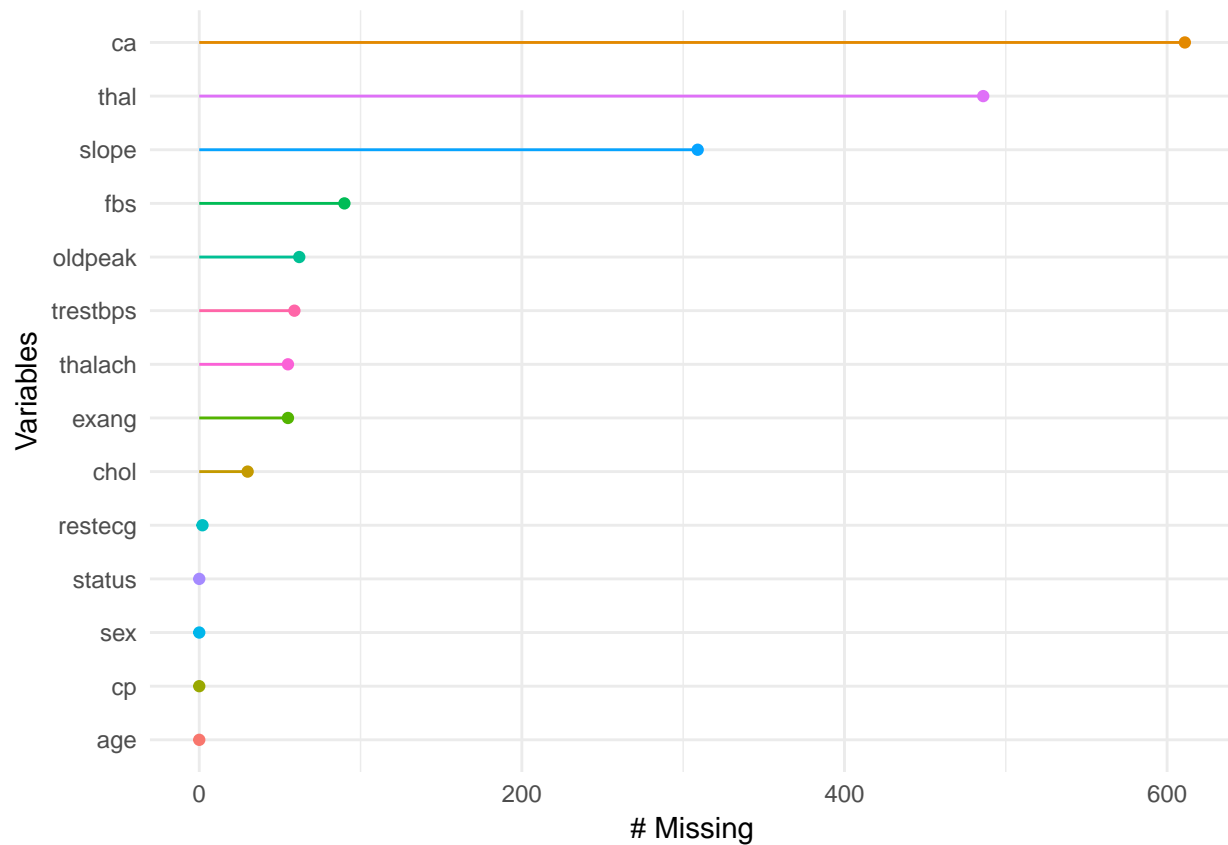
```

## 'data.frame':   920 obs. of  14 variables:
##  $ age      : num  28 29 29 29 30 31 31 32 32 32 ...
##  $ sex      : Factor w/ 2 levels "female","male": 2 2 2 2 1 1 2 1 2 2 ...
##  $ cp       : Factor w/ 4 levels "1","2","3","4": 2 2 2 2 1 2 4 2 1 2 ...
##  $ trestbps : num  130 120 130 140 170 100 120 105 95 110 ...
##  $ chol     : num  132 243 204 NA 237 219 270 198 0 225 ...
##  $ fbs      : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 NA 1 ...
##  $ restecg  : Factor w/ 3 levels "0","1","2": 3 1 3 1 2 2 1 1 1 1 ...
##  $ thalach  : num  185 160 202 170 170 150 153 165 127 184 ...
##  $ exang    : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 2 1 1 ...
##  $ oldpeak  : num  0 0 0 0 0 0 1.5 0 0.7 0 ...
##  $ slope    : Factor w/ 3 levels "1","2","3": NA NA 1 NA NA NA 2 NA 1 NA ...
##  $ ca       : Factor w/ 4 levels "0","1","2","3": NA NA 1 NA NA NA NA NA NA ...
##  $ thal     : Factor w/ 3 levels "3","6","7": NA NA 1 NA 2 NA NA NA NA ...
##  $ status   : Factor w/ 2 levels "healthy","disease": 1 1 1 1 1 1 2 1 2 1 ...

```

Graphing missing data

```
gg_miss_var(combined.data)
```

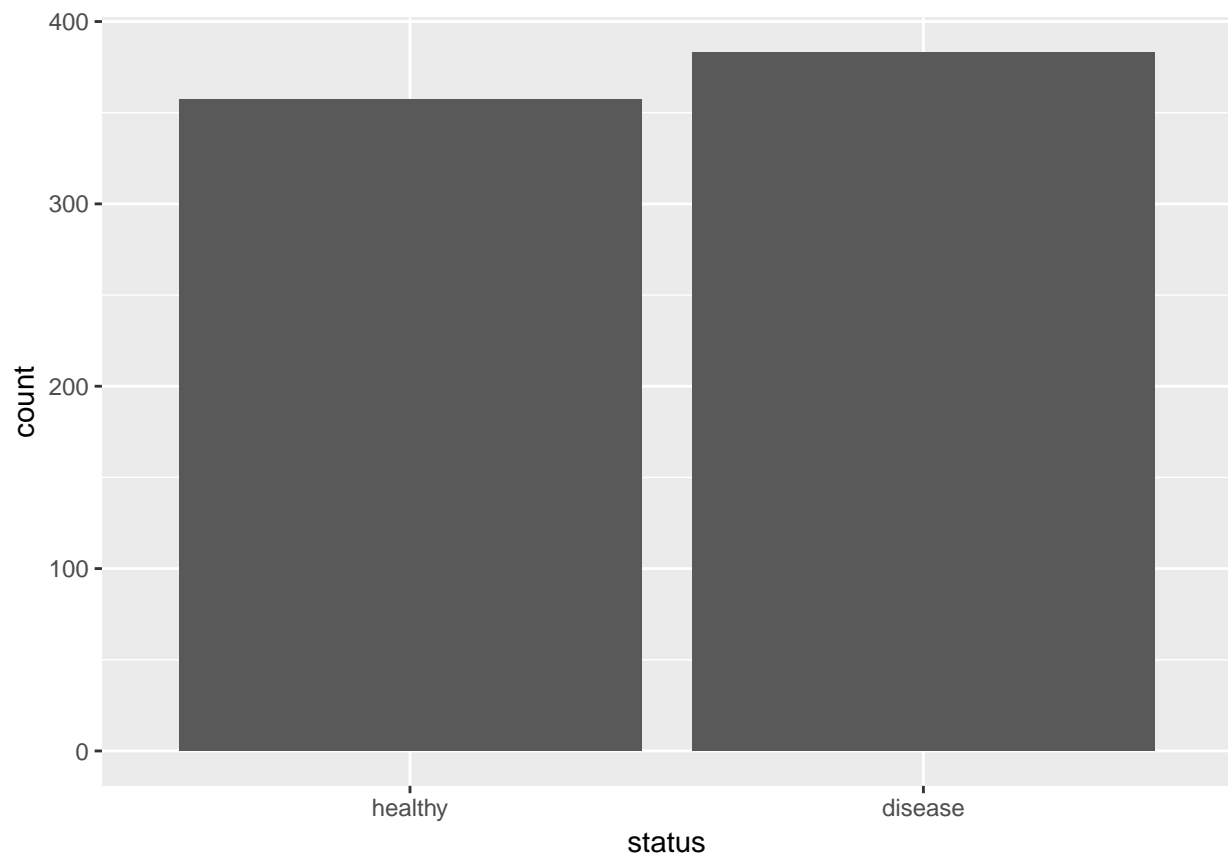


Processing missing data

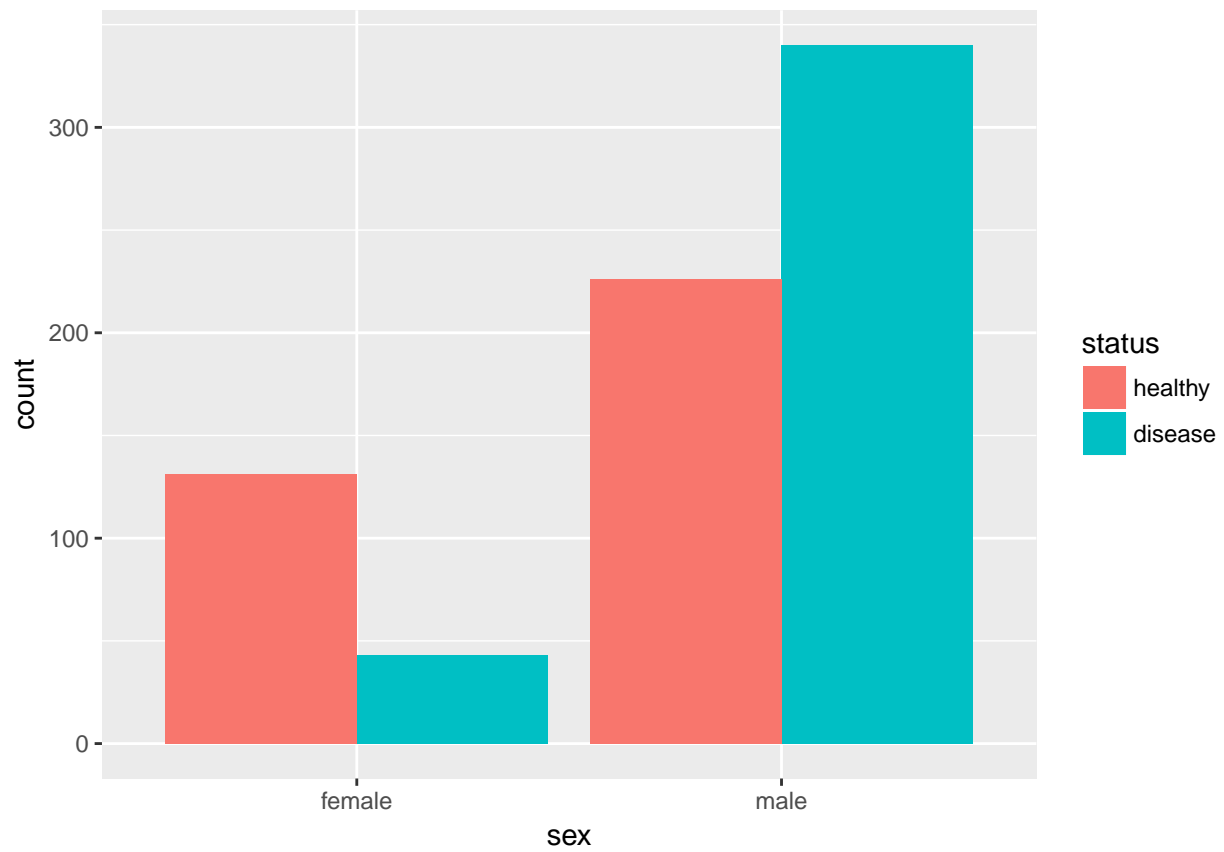
```
combined.data = subset(combined.data, select = -c(11, 12, 13))
combined.data = na.omit(combined.data)
```

Graphics for data exploration

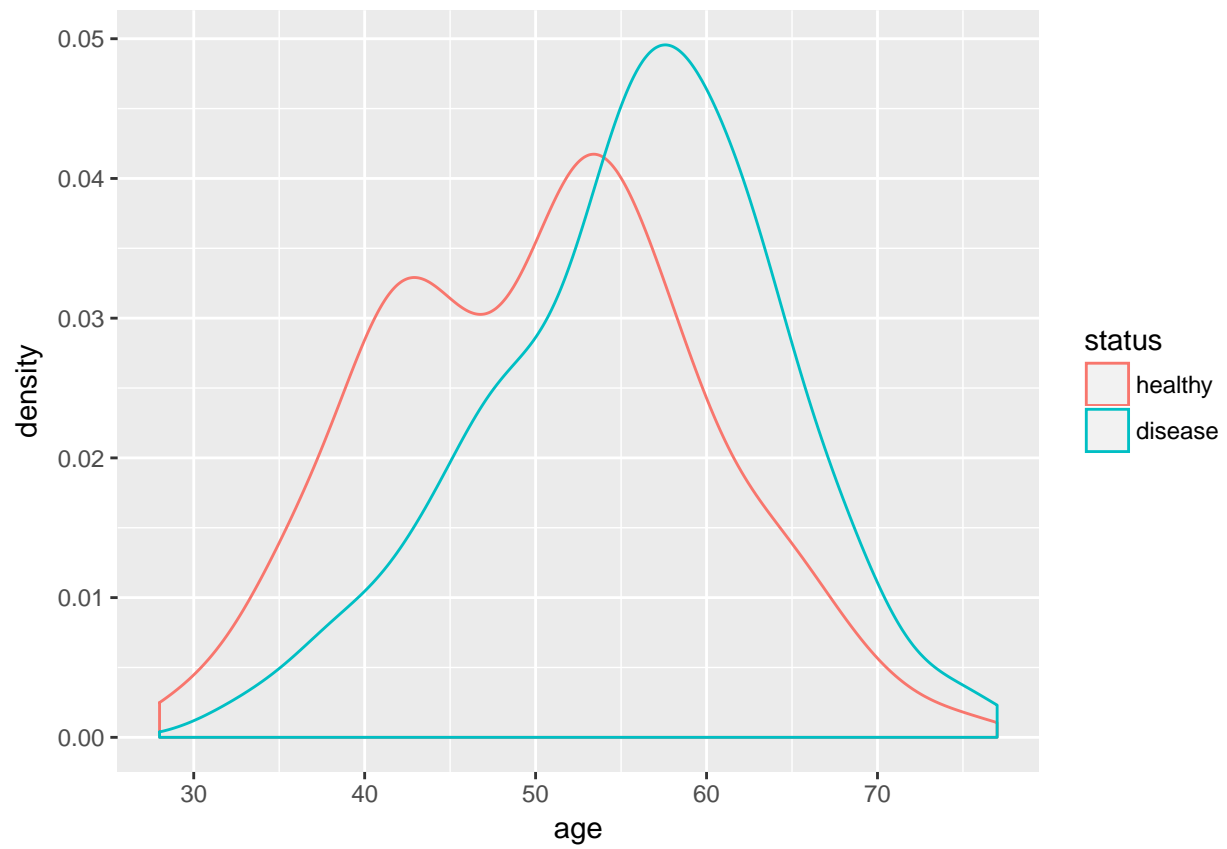
```
ggplot(combined.data, aes(x = status)) + geom_bar()
```



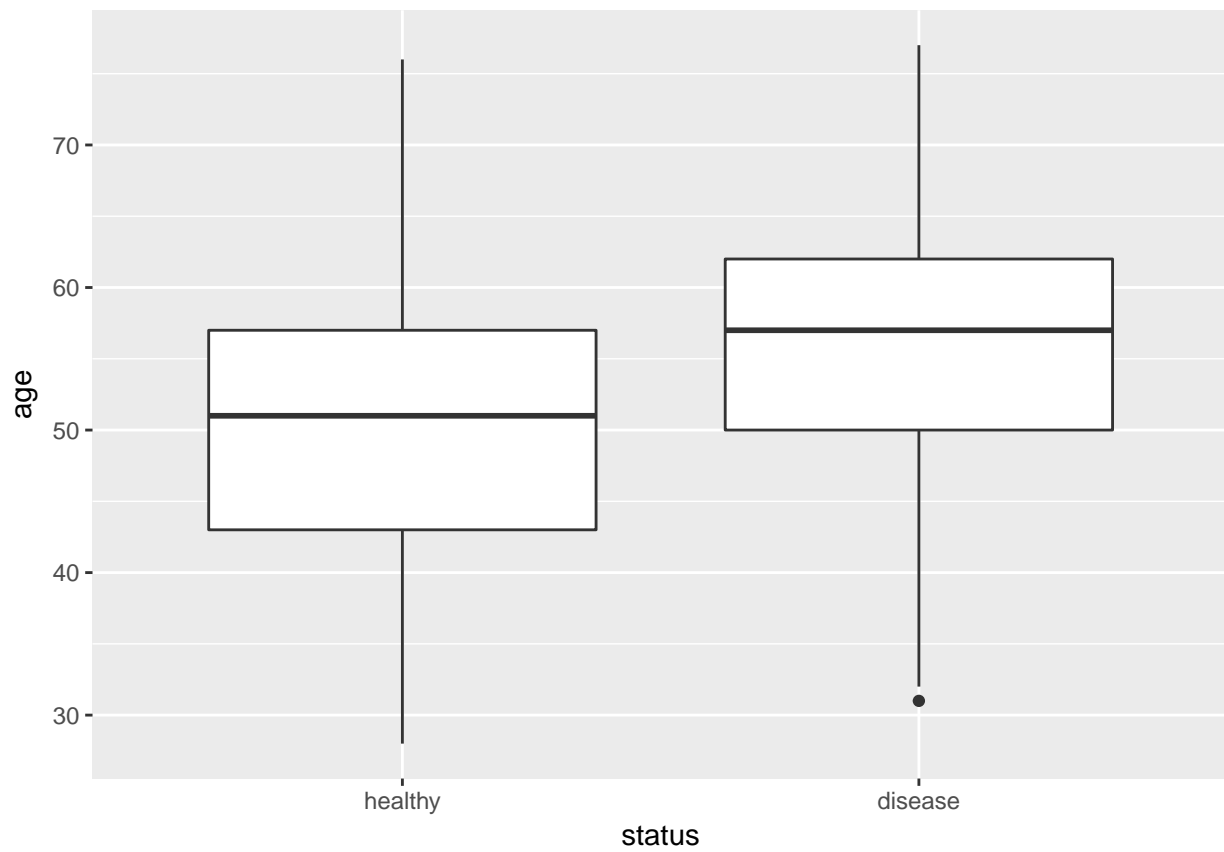
```
ggplot(combined.data, aes(x = sex, fill=status)) + geom_bar(position="dodge")
```



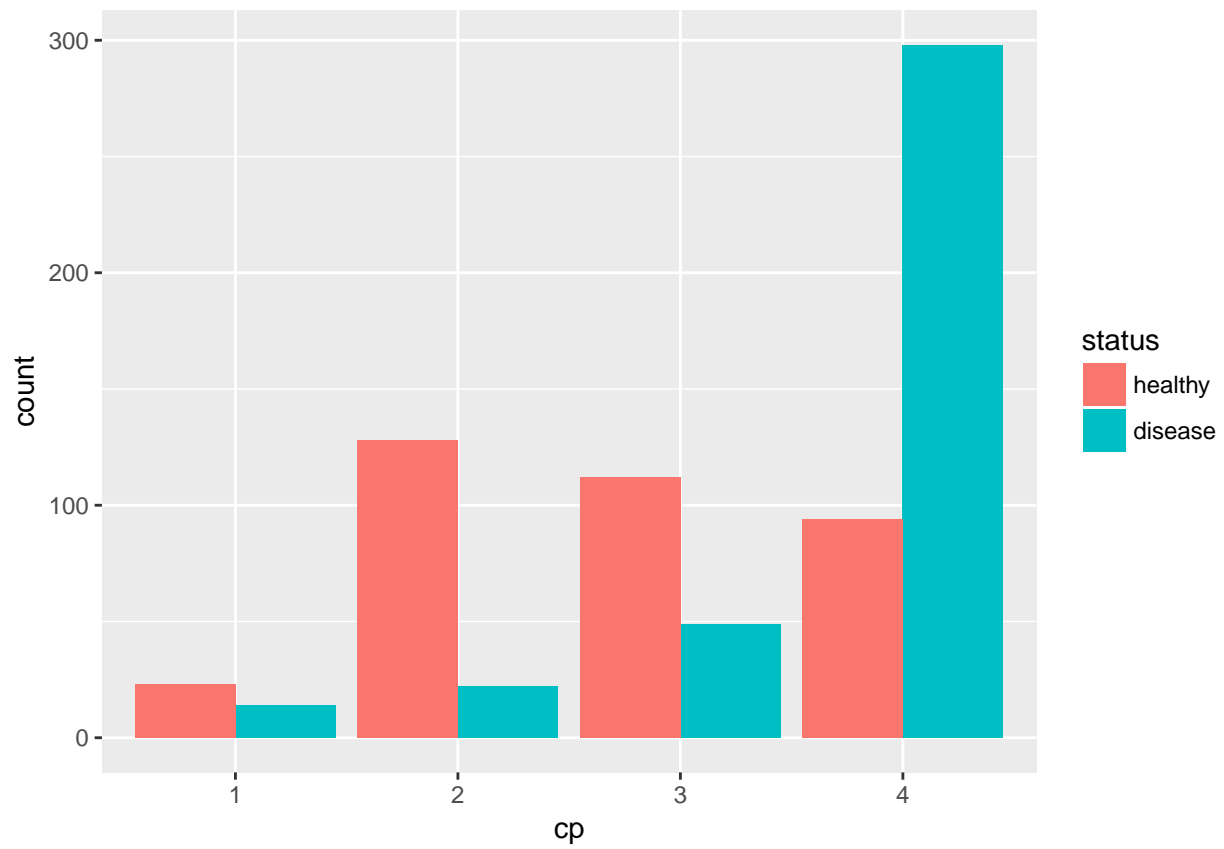
```
ggplot(combined.data, aes(x=age, color = status)) + geom_density()
```



```
ggplot(combined.data, aes(x=status, y = age)) + geom_boxplot()
```

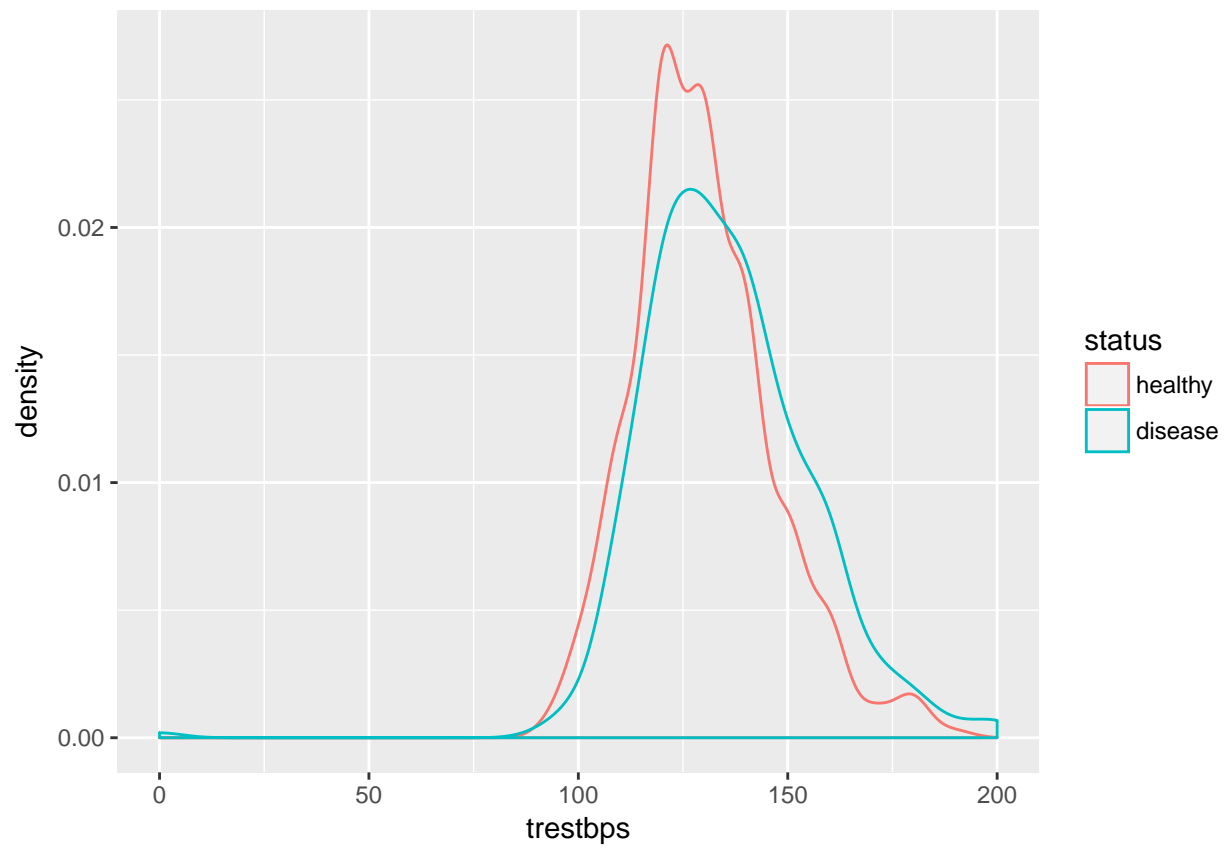


```
ggplot(combined.data, aes(x = cp, fill=status)) + geom_bar(position="dodge")
```

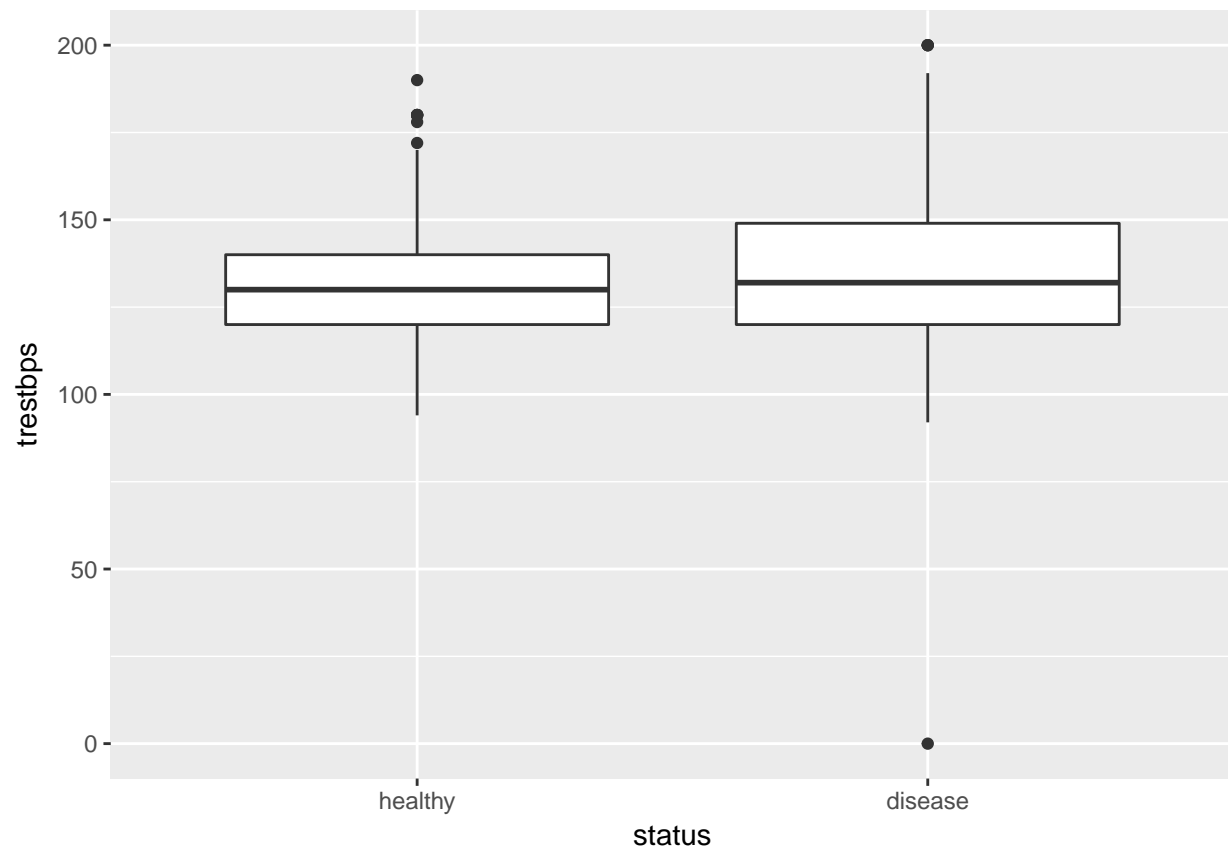


```
ggplot(combined.data, aes(x=trestbps, color = status)) + geom_density()
```

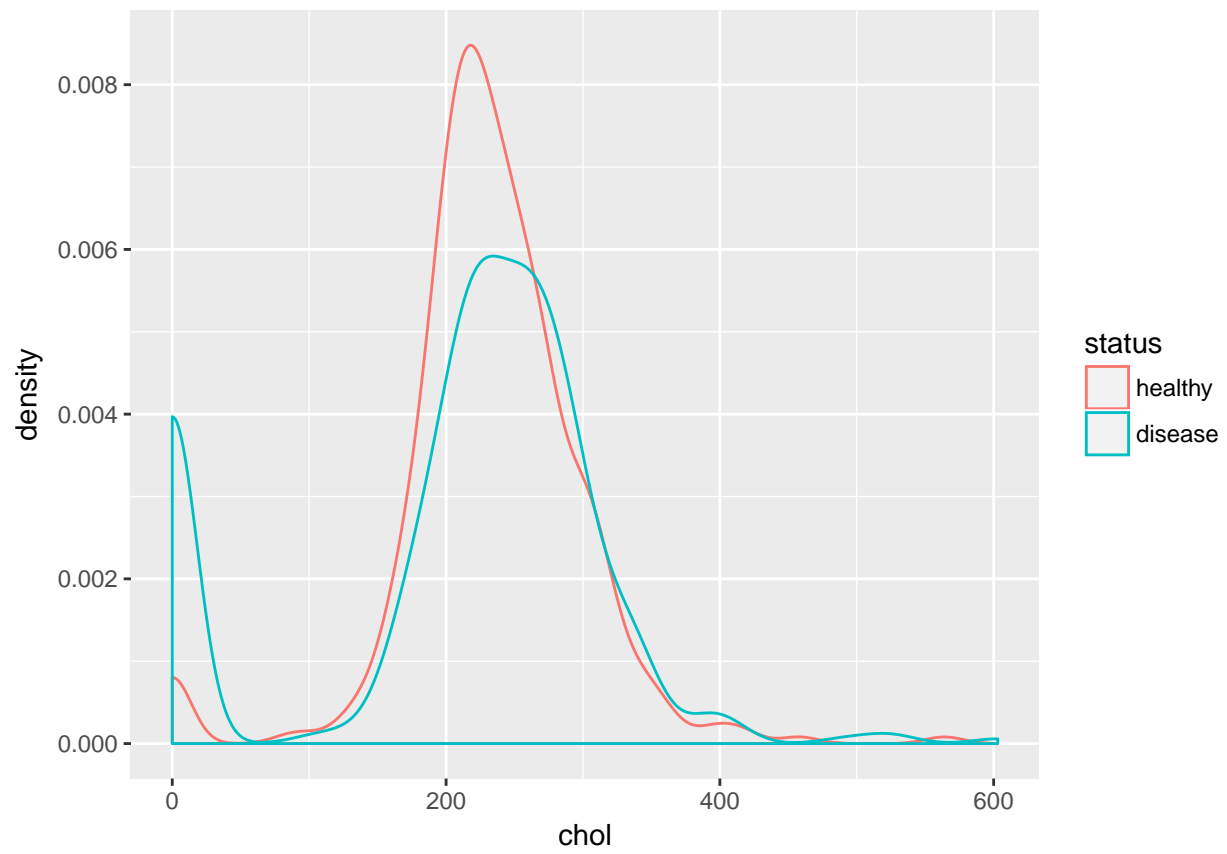




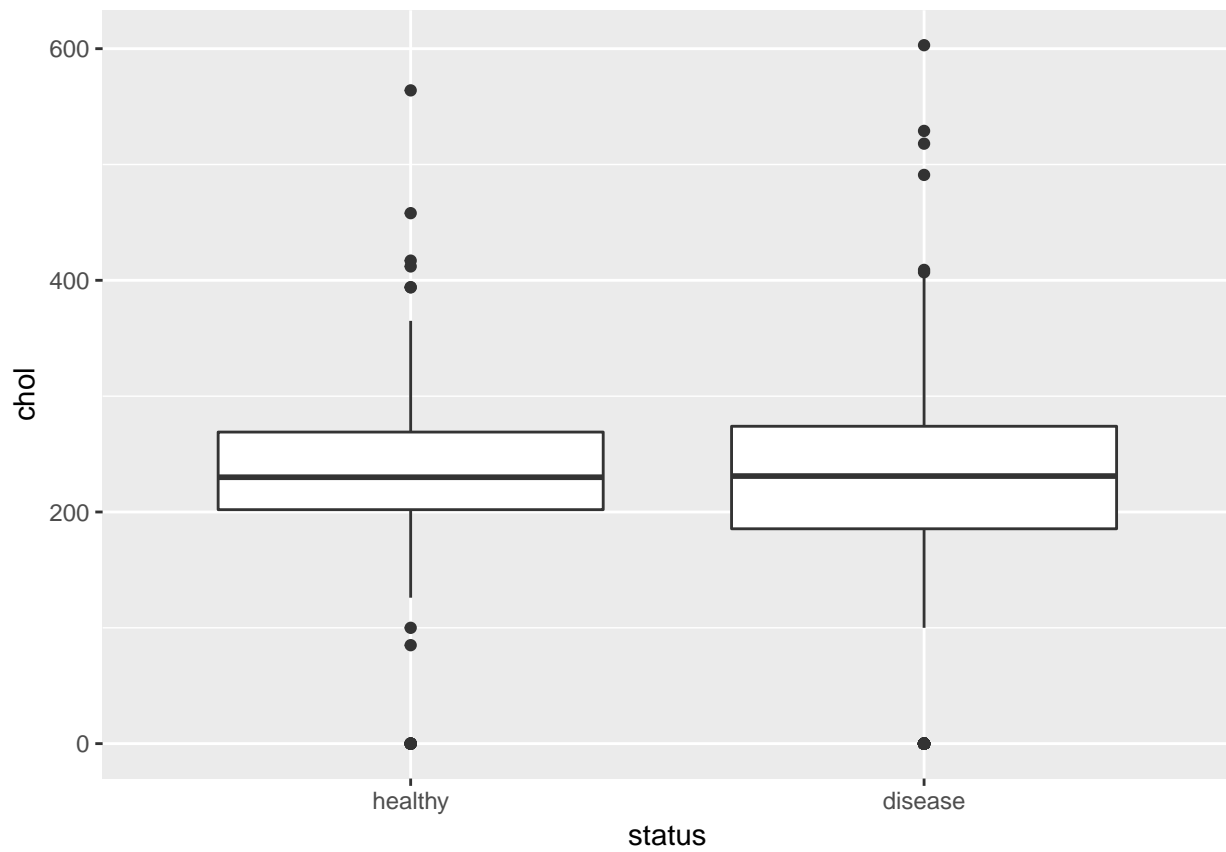
```
ggplot(combined.data, aes(x=status, y = trestbps)) + geom_boxplot()
```



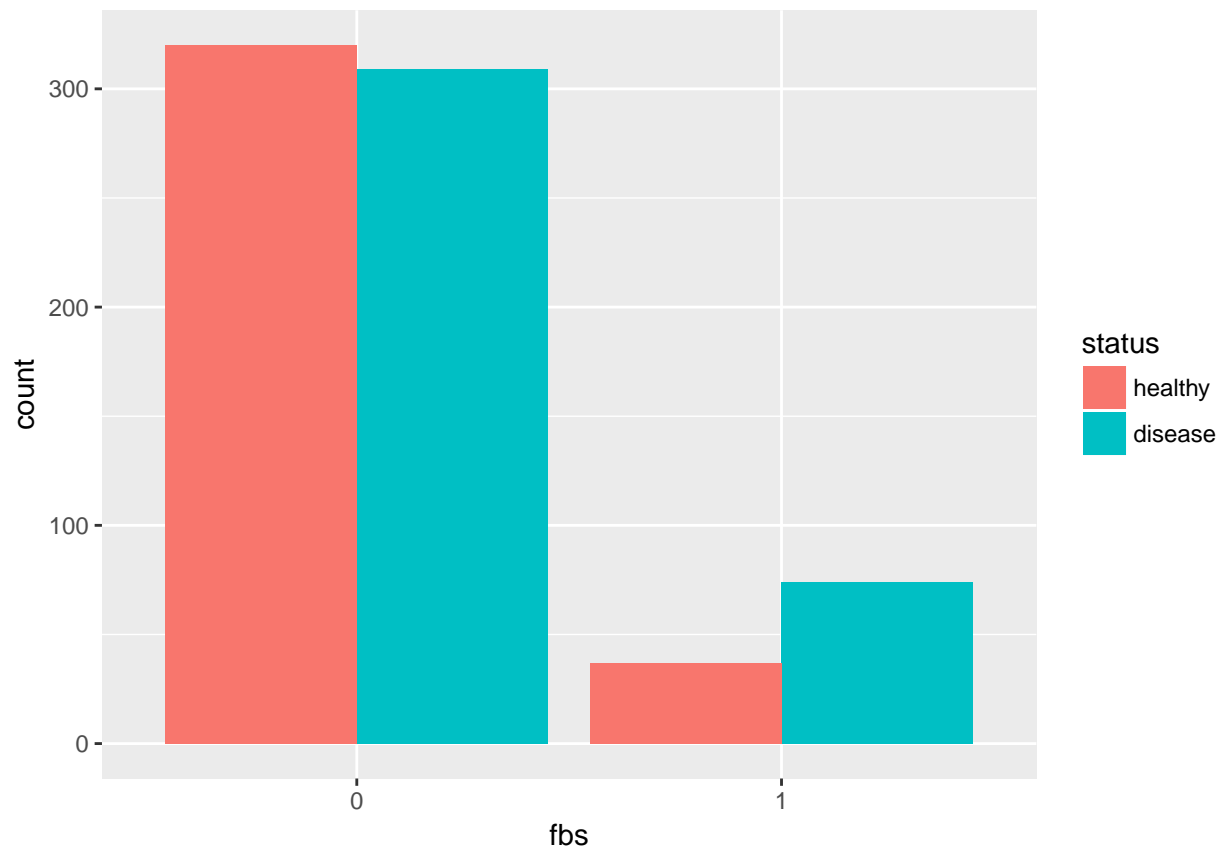
```
ggplot(combined.data, aes(x=chol, color = status)) + geom_density()
```



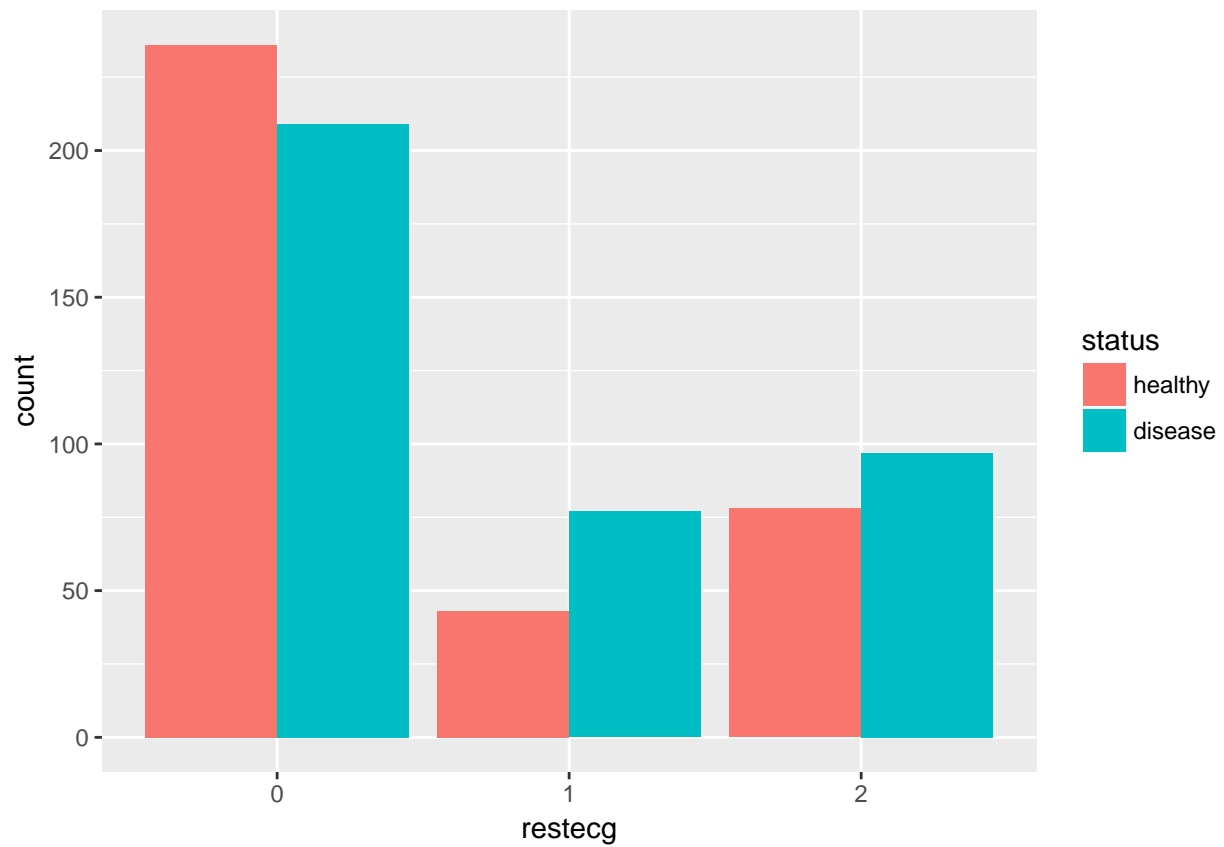
```
ggplot(combined.data, aes(x=status, y = chol)) + geom_boxplot()
```



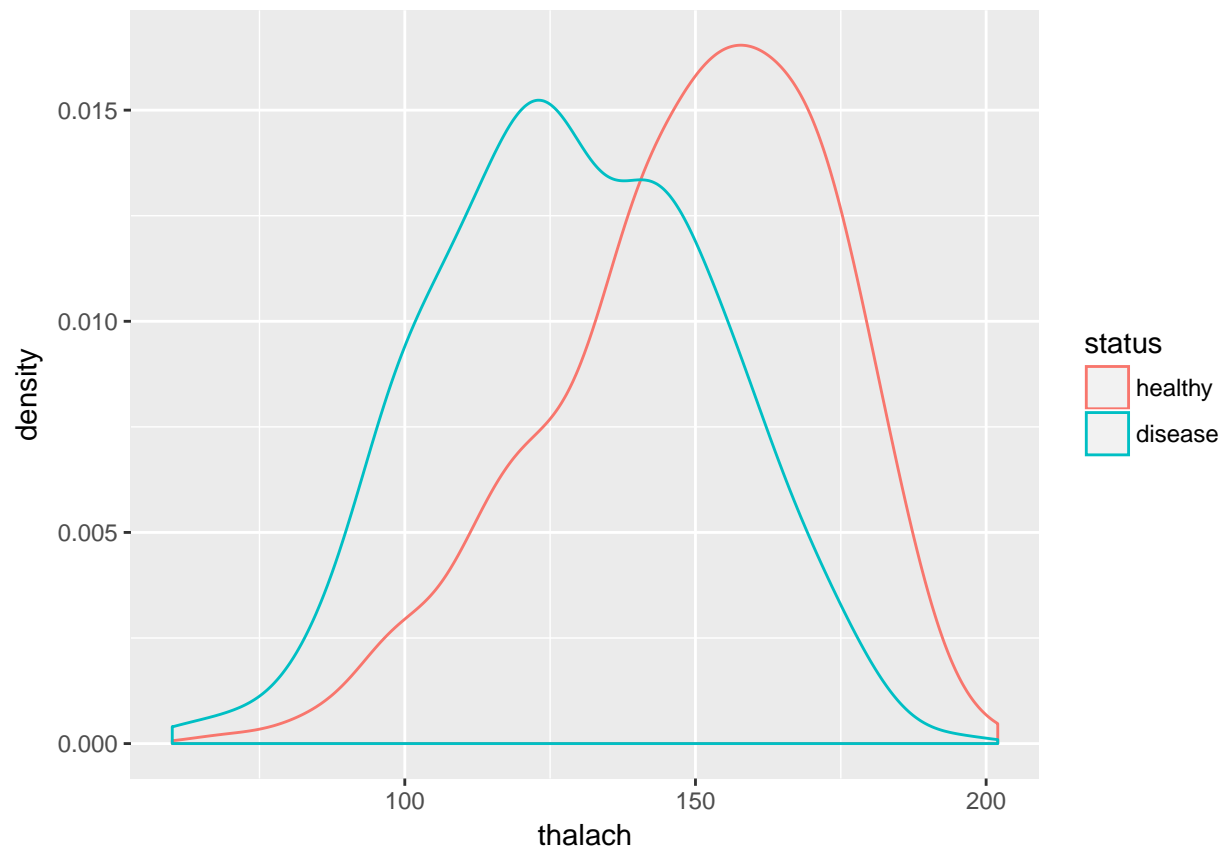
```
ggplot(combined.data, aes(x = fbs, fill=status)) + geom_bar(position="dodge")
```



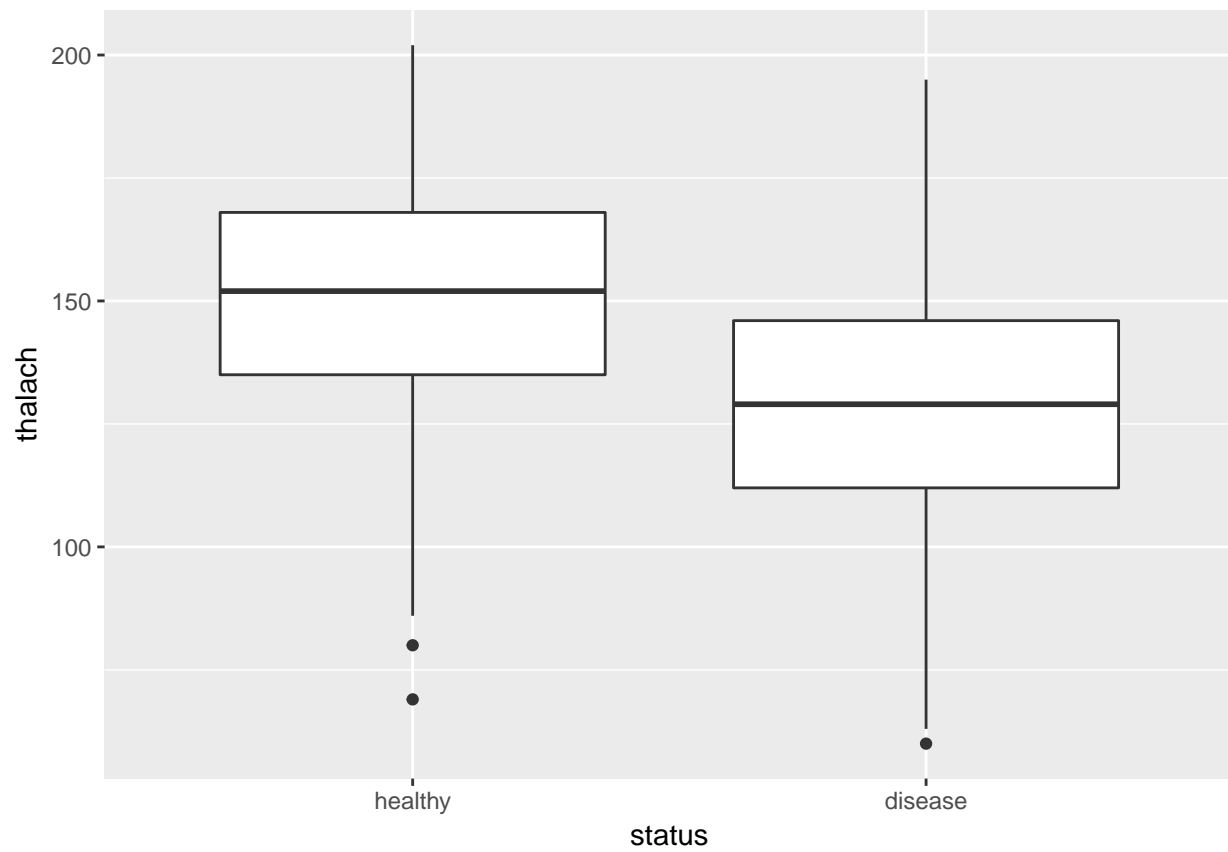
```
ggplot(combined.data, aes(x = restecg, fill=status)) + geom_bar(position="dodge")
```



```
ggplot(combined.data, aes(x=thalach, color = status)) + geom_density()
```

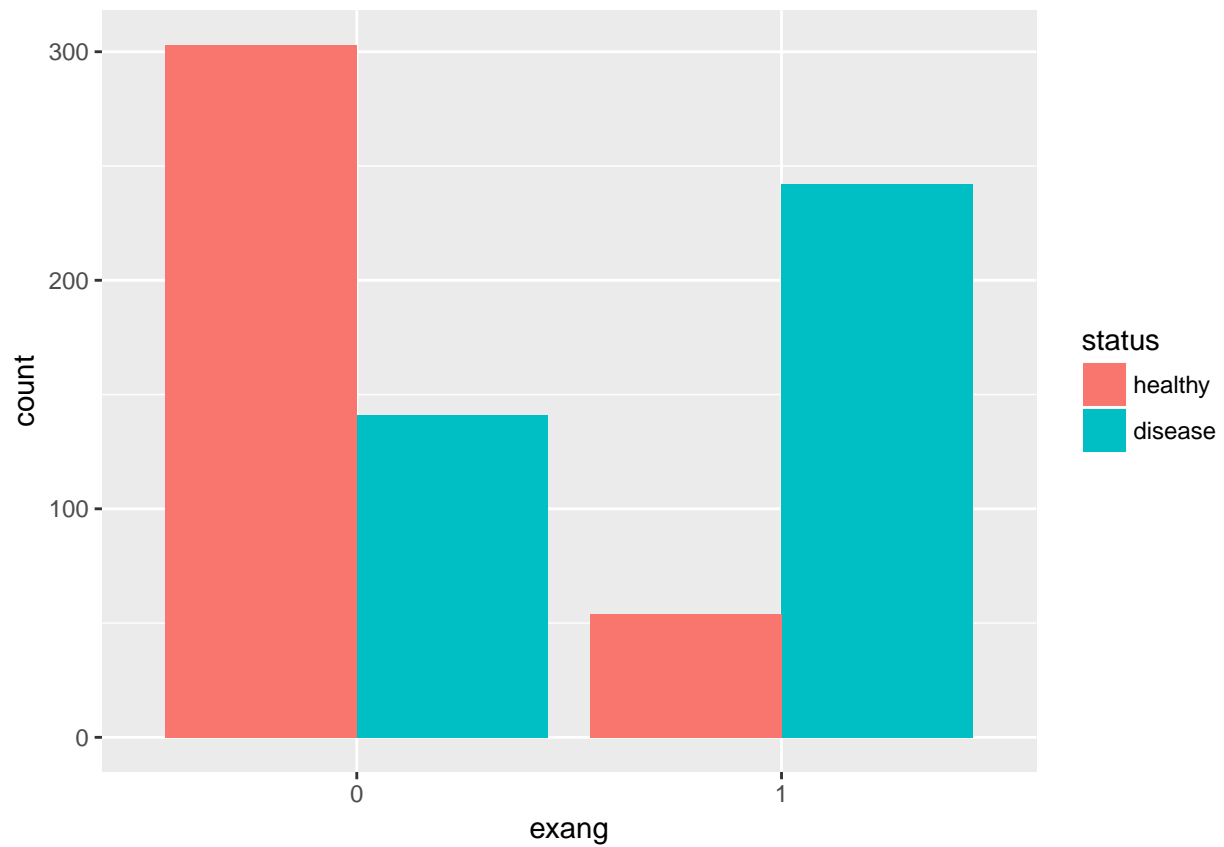


```
ggplot(combined.data, aes(x=status, y = thalach)) + geom_boxplot()
```

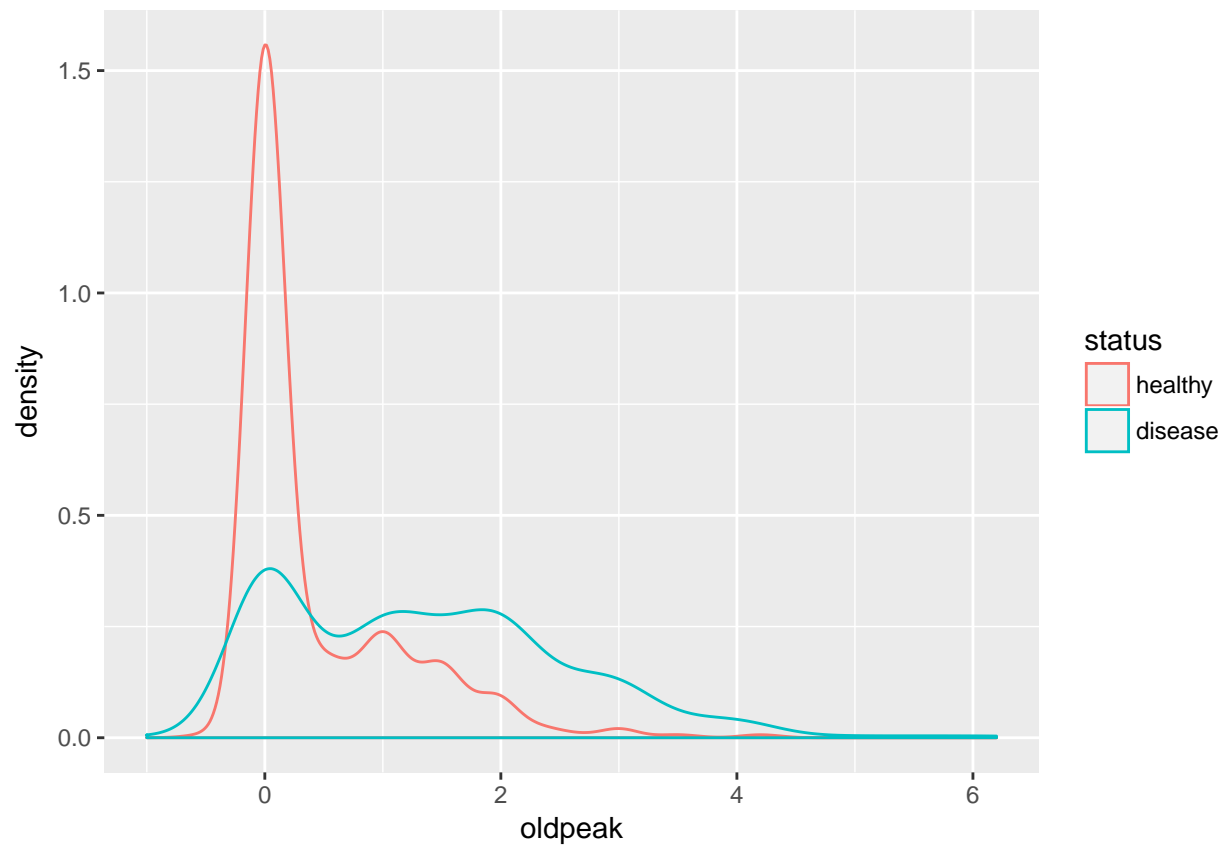


```
ggplot(combined.data, aes(x = exang, fill=status)) + geom_bar(position="dodge")
```

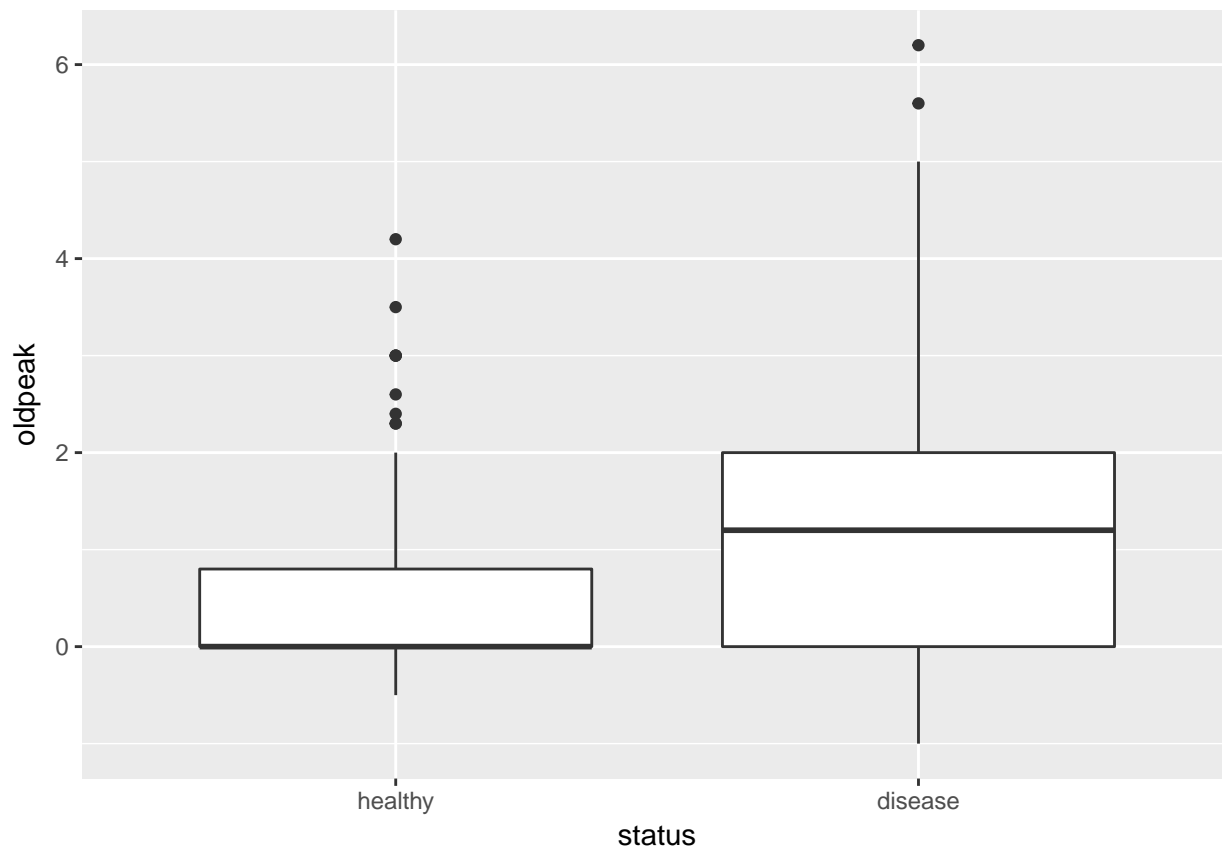




```
ggplot(combined.data, aes(x=oldpeak, color = status)) + geom_density()
```



```
ggplot(combined.data, aes(x=status, y = oldpeak)) + geom_boxplot()
```

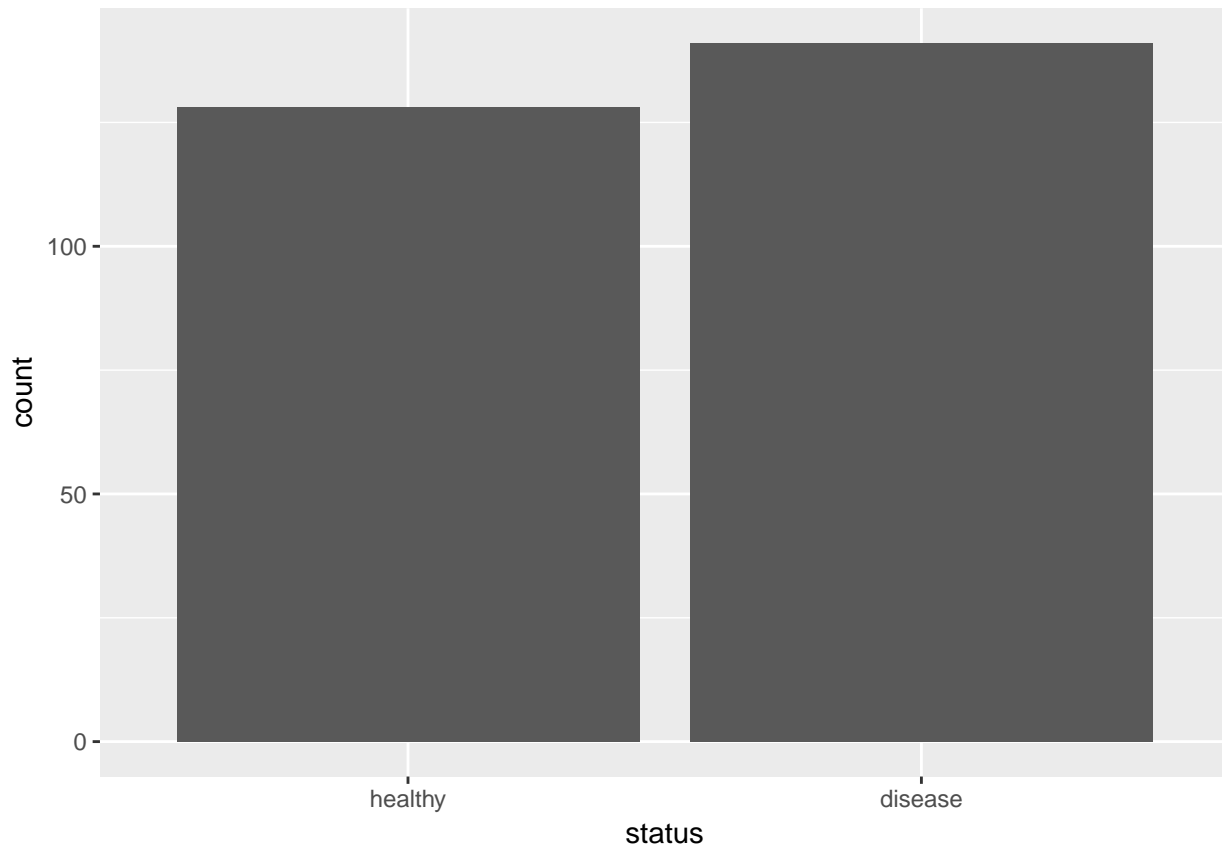


Splitting training and testing data

```
set.seed(100)
sample = sample.split(combined.data, SplitRatio = 0.70)
train = subset(combined.data, sample == T)
test = subset(combined.data, sample == F)
summary(test)
```

```
##      age      sex      cp      trestbps      chol
## Min.   :29.0  female: 67  1: 14  Min.    : 0.0  Min.    : 0.0
## 1st Qu.:46.0  male   :202  2: 53  1st Qu.:120.0 1st Qu.:203.0
## Median :54.0                3: 59  Median :130.0 Median :234.0
## Mean   :53.2                4:143  Mean   :132.3 Mean   :220.9
## 3rd Qu.:60.0                3rd Qu.:140.0 3rd Qu.:268.0
## Max.   :77.0                Max.   :200.0 Max.   :564.0
## fbs    restecg    thalach    exang    oldpeak    status
## 0:228   0:166   Min.     : 69.0  0:164   Min.    :0.0000  healthy:128
## 1: 41   1: 44   1st Qu.:120.0  1:105   1st Qu.:0.0000  disease:141
##        2: 59   Median :140.0                Median :0.4000
##        Mean   :139.4                Mean   :0.8643
##        3rd Qu.:160.0                3rd Qu.:1.5000
##        Max.   :202.0                Max.   :5.0000
```

```
ggplot(test, aes(x = status)) + geom_bar()
```



Baseline approach: Logistic regression (both with full model training, and with the selected five predictors)

```
logit.model = train(status ~ ., data=train, method = 'glm', family = 'binomial')
summary(logit.model)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -2.8479  -0.6053   0.1551   0.5699   2.6069
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.9049653  1.7734166  -1.638  0.101409
## age          0.0162828  0.0161423   1.009  0.313119
## sexmale      1.3139698  0.3380953   3.886  0.000102 ***
## cp2         -0.0368386  0.6284873  -0.059  0.953259
## cp3          0.2472175  0.5918941   0.418  0.676187
## cp4          1.5137979  0.5736822   2.639  0.008321 **
## trestbps     0.0138848  0.0075800   1.832  0.066986 .
## chol        -0.0009002  0.0015325  -0.587  0.556919
## fbs1         0.5491646  0.3725546   1.474  0.140468
## restecg1     0.0207829  0.3743793   0.056  0.955730
## restecg2     0.3275127  0.3183371   1.029  0.303562
## thalach     -0.0187948  0.0058359  -3.221  0.001279 **
## exang1       0.9493386  0.2832175   3.352  0.000802 ***
```

```
## oldpeak      0.7788933  0.1484223   5.248 1.54e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##      Null deviance: 652.59  on 470  degrees of freedom
## Residual deviance: 390.26  on 457  degrees of freedom
## AIC: 418.26
##
## Number of Fisher Scoring iterations: 5
```

```
logit.result = predict(logit.model, test)
confusionMatrix(test$status, logit.result)
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction healthy disease
##   healthy      109      19
##   disease       35     106
##
##              Accuracy : 0.7993
##              95% CI : (0.7463, 0.8454)
##   No Information Rate : 0.5353
##   P-Value [Acc > NIR] : < 2e-16
##
##              Kappa : 0.5999
## Mcnemar's Test P-Value : 0.04123
##
##              Sensitivity : 0.7569
##              Specificity : 0.8480
##              Pos Pred Value : 0.8516
##              Neg Pred Value : 0.7518
##              Prevalence : 0.5353
##              Detection Rate : 0.4052
##   Detection Prevalence : 0.4758
##              Balanced Accuracy : 0.8025
##
##              'Positive' Class : healthy
##
```

```
smalllogit.model = train(status ~ sex +cp + thalach +exang +oldpeak, data=train, method = 'glm', family
smalllogit.result = predict(smalllogit.model, test)
confusionMatrix(test$status, smalllogit.result)
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction healthy disease
##   healthy      108      20
##   disease       32     109
##
##              Accuracy : 0.8067
##              95% CI : (0.7544, 0.8521)
```

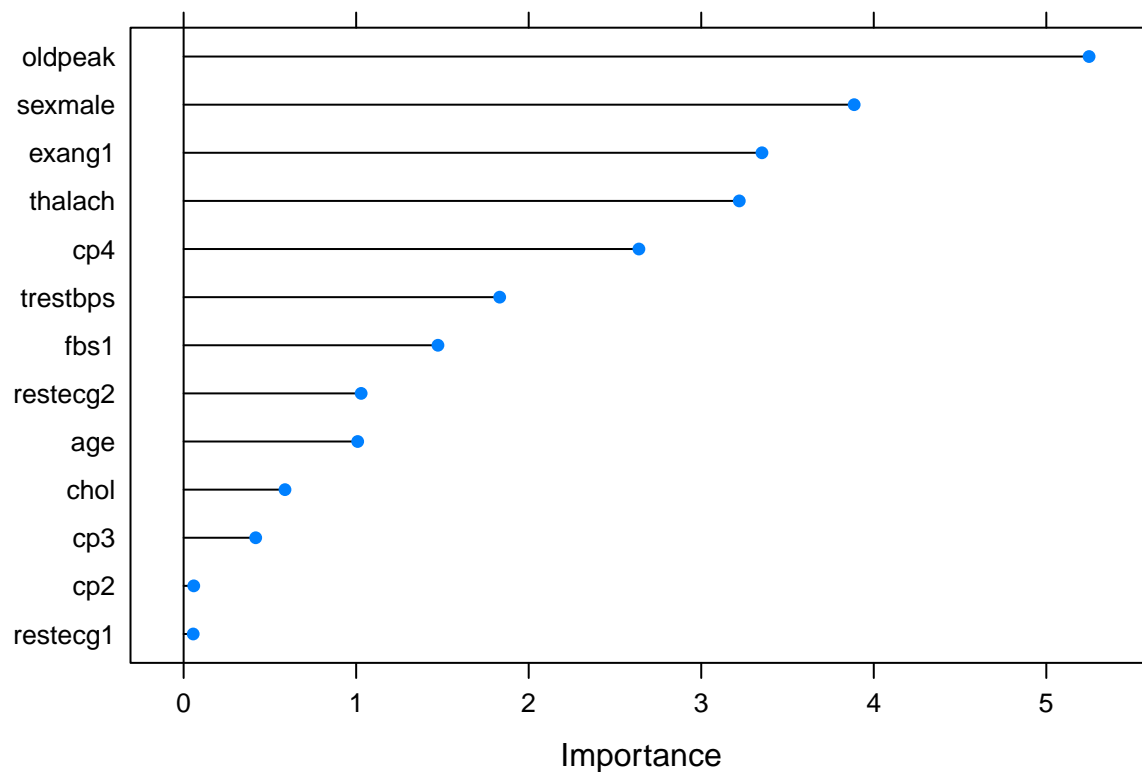
```
##      No Information Rate : 0.5204
##      P-Value [Acc > NIR] : <2e-16
##
##              Kappa : 0.6141
##  Mcnemar's Test P-Value : 0.1272
##
##      Sensitivity : 0.7714
##      Specificity : 0.8450
##      Pos Pred Value : 0.8438
##      Neg Pred Value : 0.7730
##      Prevalence : 0.5204
##      Detection Rate : 0.4015
##      Detection Prevalence : 0.4758
##      Balanced Accuracy : 0.8082
##
##      'Positive' Class : healthy
##
```

variable importance for logistic model

```
importance = varImp(logit.model, scale=FALSE)
importance
```

```
## glm variable importance
##
##      Overall
## oldpeak  5.24782
## sexmale  3.88639
## exang1    3.35198
## thalach  3.22054
## cp4       2.63874
## trestbps 1.83177
## fbs1      1.47405
## restecg2 1.02882
## age       1.00870
## chol      0.58742
## cp3       0.41767
## cp2       0.05861
## restecg1 0.05551
```

```
plot(importance)
```



Approach 2: Naive Bayes classification, with both full model, a small model with 5 predictors, and a “best” model that includes all the predictors except trestbps.

```
naiveBayes.model = naiveBayes(status ~., data = train)
naiveBayes.model
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##   healthy   disease
## 0.4861996 0.5138004
##
## Conditional probabilities:
##           age
## Y           [,1]      [,2]
## healthy 50.30131 9.527479
## disease 55.63223 8.548681
##
##           sex
## Y           female      male
## healthy 0.3580786 0.6419214
## disease 0.1033058 0.8966942
##
##           cp
## Y           1           2           3           4
```

```

## healthy 0.06550218 0.34934498 0.31441048 0.27074236
## disease 0.03305785 0.07024793 0.12396694 0.77272727
##
## trestbps
## Y      [,1]      [,2]
## healthy 129.7380 16.77721
## disease 136.0744 18.21457
##
## chol
## Y      [,1]      [,2]
## healthy 230.5939 65.9265
## disease 209.3430 109.4753
##
## fbs
## Y      0      1
## healthy 0.8908297 0.1091703
## disease 0.8140496 0.1859504
##
## restecg
## Y      0      1      2
## healthy 0.6593886 0.1179039 0.2227074
## disease 0.5289256 0.2024793 0.2685950
##
## thalach
## Y      [,1]      [,2]
## healthy 149.2314 22.97049
## disease 128.0372 23.89766
##
## exang
## Y      0      1
## healthy 0.8384279 0.1615721
## disease 0.3636364 0.6363636
##
## oldpeak
## Y      [,1]      [,2]
## healthy 0.4039301 0.6572352
## disease 1.3917355 1.2229538

```

```

naiveBayes.result = predict(naiveBayes.model, train)
confusionMatrix(train$status, naiveBayes.result)

```

```

## Confusion Matrix and Statistics
##
##      Reference
## Prediction healthy disease
## healthy      188      41
## disease       47     195
##
##      Accuracy : 0.8132
##      95% CI : (0.775, 0.8474)
##      No Information Rate : 0.5011
##      P-Value [Acc > NIR] : <2e-16
##
##      Kappa : 0.6263
##      Mcnemar's Test P-Value : 0.594

```



```
##
##          Sensitivity : 0.8000
##          Specificity : 0.8263
##          Pos Pred Value : 0.8210
##          Neg Pred Value : 0.8058
##          Prevalence : 0.4989
##          Detection Rate : 0.3992
##          Detection Prevalence : 0.4862
##          Balanced Accuracy : 0.8131
##
##          'Positive' Class : healthy
##
```

```
smallnaiveBayes.model = naiveBayes(status ~sex +cp + thalach +exang +oldpeak, data = train)
smallnaiveBayes.result = predict(smallnaiveBayes.model, test)
confusionMatrix(test$status, smallnaiveBayes.result)
```

```
## Confusion Matrix and Statistics
##
##          Reference
## Prediction healthy disease
##   healthy      108      20
##   disease       35     106
##
##          Accuracy : 0.7955
##          95% CI : (0.7423, 0.8421)
##   No Information Rate : 0.5316
##   P-Value [Acc > NIR] : < 2e-16
##
##          Kappa : 0.5923
## Mcnemar's Test P-Value : 0.05906
##
##          Sensitivity : 0.7552
##          Specificity : 0.8413
##          Pos Pred Value : 0.8438
##          Neg Pred Value : 0.7518
##          Prevalence : 0.5316
##          Detection Rate : 0.4015
##          Detection Prevalence : 0.4758
##          Balanced Accuracy : 0.7983
##
##          'Positive' Class : healthy
##
```

```
bestnaiveBayes.model = naiveBayes(status ~sex +cp + thalach +exang +oldpeak+restecg+fbs+chol+age, data = train)
bestnaiveBayes.result = predict(bestnaiveBayes.model, test)
confusionMatrix(test$status, bestnaiveBayes.result)
```

```
## Confusion Matrix and Statistics
##
##          Reference
## Prediction healthy disease
##   healthy      109      19
##   disease       24     117
##
```

```

##              Accuracy : 0.8401
##              95% CI : (0.7908, 0.8818)
##      No Information Rate : 0.5056
##      P-Value [Acc > NIR] : <2e-16
##
##              Kappa : 0.6801
##  Mcnemar's Test P-Value : 0.5419
##
##      Sensitivity : 0.8195
##      Specificity : 0.8603
##      Pos Pred Value : 0.8516
##      Neg Pred Value : 0.8298
##      Prevalence : 0.4944
##      Detection Rate : 0.4052
##      Detection Prevalence : 0.4758
##      Balanced Accuracy : 0.8399
##
##      'Positive' Class : healthy
##

```

Approach 3: Support vector machines, using a 3-repeat, 10-fold cross validation to fine tune the hyperparameters C, using Gaussian radial kernel. Includes both the results of the full model and a small model with the selected 5 predictors.

```

fitControl = trainControl(method = "repeatedcv", number = 10, repeats = 3)
set.seed(100)
svm.model = train(status ~., data = train, method = "svmRadial", trControl = fitControl)
svm.model

```

```

## Support Vector Machines with Radial Basis Function Kernel
##
## 471 samples
## 10 predictor
## 2 classes: 'healthy', 'disease'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 424, 424, 424, 424, 424, 424, ...
## Resampling results across tuning parameters:
##
##      C      Accuracy   Kappa
##      0.25  0.7890052  0.5774449
##      0.50  0.7911630  0.5825566
##      1.00  0.7926425  0.5856137
##
## Tuning parameter 'sigma' was held constant at a value of 0.05299966
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.05299966 and C = 1.
svm.result = predict(svm.model, newdata = test)
confusionMatrix(test$status, svm.result)

## Confusion Matrix and Statistics
##
##              Reference
## Prediction healthy disease

```

```
##      healthy      111      17
##      disease       33     108
##
##              Accuracy : 0.8141
##              95% CI : (0.7624, 0.8588)
##      No Information Rate : 0.5353
##      P-Value [Acc > NIR] : < 2e-16
##
##              Kappa : 0.6295
##  McNemar's Test P-Value : 0.03389
##
##      Sensitivity : 0.7708
##      Specificity : 0.8640
##      Pos Pred Value : 0.8672
##      Neg Pred Value : 0.7660
##      Prevalence : 0.5353
##      Detection Rate : 0.4126
##      Detection Prevalence : 0.4758
##      Balanced Accuracy : 0.8174
##
##      'Positive' Class : healthy
##
```

```
set.seed(100)
smallsvm.model = train(status ~sex +cp + thalach +exang +oldpeak, data = train,
                        method = "svmRadial",
                        trControl = fitControl)
smallsvm.result = predict(smallsvm.model, newdata = test)
confusionMatrix(test$status, smallsvm.result)
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction healthy disease
##      healthy      112      16
##      disease       41     100
##
##              Accuracy : 0.7881
##              95% CI : (0.7344, 0.8354)
##      No Information Rate : 0.5688
##      P-Value [Acc > NIR] : 3.319e-14
##
##              Kappa : 0.579
##  McNemar's Test P-Value : 0.001478
##
##      Sensitivity : 0.7320
##      Specificity : 0.8621
##      Pos Pred Value : 0.8750
##      Neg Pred Value : 0.7092
##      Prevalence : 0.5688
##      Detection Rate : 0.4164
##      Detection Prevalence : 0.4758
##      Balanced Accuracy : 0.7970
##
##      'Positive' Class : healthy
##
```

```
##
```

Variable importance for the full svm model.

```
importance = varImp(svm.model, scale=FALSE)
importance
```

```
## ROC curve variable importance
```

```
##
```

```
##      Importance
```

```
## cp      0.7570
```

```
## oldpeak  0.7479
```

```
## thalach  0.7430
```

```
## exang    0.7374
```

```
## age      0.6689
```

```
## sex      0.6274
```

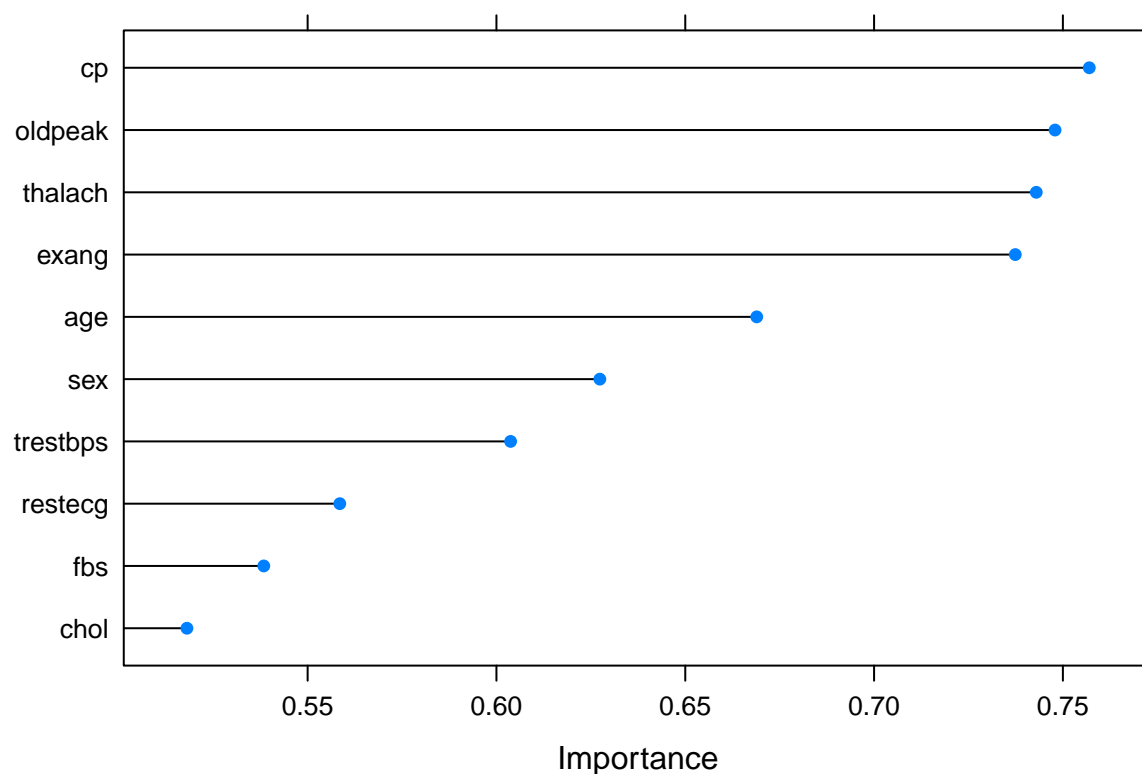
```
## trestbps 0.6037
```

```
## restecg  0.5585
```

```
## fbs      0.5384
```

```
## chol     0.5180
```

```
plot(importance)
```



Approach 4: Random forests, with 10 fold, 3 repeat cross validation to tune the parameter mtry. With results from both the full model and the small model using the 5 selected predictors

```
fitControl = trainControl(method = "repeatedcv", number = 10, repeats = 3)
set.seed(100)
rf.model = train(status~., data=train, method="rf", trControl = fitControl)
rf.model
```

```
## Random Forest
```

```

##
## 471 samples
## 10 predictor
## 2 classes: 'healthy', 'disease'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 424, 424, 424, 424, 424, 424, ...
## Resampling results across tuning parameters:
##
## mtry Accuracy Kappa
## 2 0.7855053 0.5706418
## 7 0.7685130 0.5366725
## 13 0.7550667 0.5098978
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
rf.result = predict(rf.model, test)
confusionMatrix(test$status, rf.result)

## Confusion Matrix and Statistics
##
## Reference
## Prediction healthy disease
## healthy 109 19
## disease 30 111
##
## Accuracy : 0.8178
## 95% CI : (0.7664, 0.8621)
## No Information Rate : 0.5167
## P-Value [Acc > NIR] : <2e-16
##
## Kappa : 0.6363
## McNemar's Test P-Value : 0.1531
##
## Sensitivity : 0.7842
## Specificity : 0.8538
## Pos Pred Value : 0.8516
## Neg Pred Value : 0.7872
## Prevalence : 0.5167
## Detection Rate : 0.4052
## Detection Prevalence : 0.4758
## Balanced Accuracy : 0.8190
##
## 'Positive' Class : healthy
##

predictors(rf.model)

## [1] "age" "sexmale" "cp2" "cp3" "cp4" "trestbps"
## [7] "chol" "fbs1" "restecg1" "restecg2" "thalach" "exang1"
## [13] "oldpeak"

```

Small random forests model using only the 5 selected predictors.

```

set.seed(100)
fitControl = trainControl(method = "repeatedcv", number = 10, repeats = 3)
smallrf.model = train(status~sex +cp + thalach +exang +oldpeak, data=train, method="rf")
smallrf.model

## Random Forest
##
## 471 samples
## 5 predictor
## 2 classes: 'healthy', 'disease'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 471, 471, 471, 471, 471, 471, ...
## Resampling results across tuning parameters:
##
## mtry Accuracy Kappa
## 2 0.7731039 0.5455893
## 4 0.7484221 0.4966682
## 7 0.7363129 0.4727022
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.

smallrf.result = predict(smallrf.model, test)
confusionMatrix(test$status, smallrf.result)

## Confusion Matrix and Statistics
##
## Reference
## Prediction healthy disease
## healthy 107 21
## disease 36 105
##
## Accuracy : 0.7881
## 95% CI : (0.7344, 0.8354)
## No Information Rate : 0.5316
## P-Value [Acc > NIR] : < 2e-16
##
## Kappa : 0.5775
## McNemar's Test P-Value : 0.06369
##
## Sensitivity : 0.7483
## Specificity : 0.8333
## Pos Pred Value : 0.8359
## Neg Pred Value : 0.7447
## Prevalence : 0.5316
## Detection Rate : 0.3978
## Detection Prevalence : 0.4758
## Balanced Accuracy : 0.7908
##
## 'Positive' Class : healthy
##

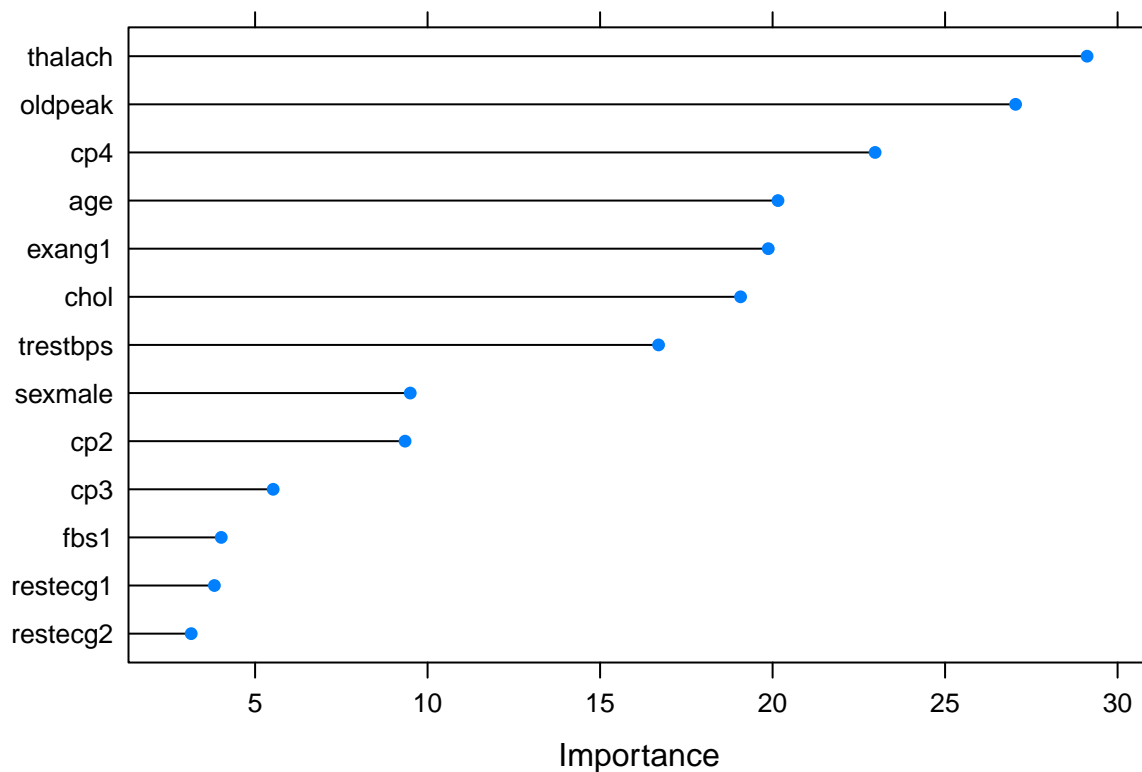
```

Variable importance for the full random forest model

```
importance = varImp(rf.model, scale=FALSE)
importance
```

```
## rf variable importance
##
## Overall
## thalach 29.115
## oldpeak 27.043
## cp4 22.971
## age 20.156
## exang1 19.875
## chol 19.073
## trestbps 16.695
## sexmale 9.497
## cp2 9.348
## cp3 5.525
## fbs1 4.020
## restecg1 3.821
## restecg2 3.149
```

```
plot(importance)
```



Variable selection via recursive variable elimination using random forests

```
set.seed(100)
control = rfeControl(functions=rfFuncs, method="cv", number=10, repeats = 3, returnResamp = "all")
results = rfe(train[, 1:10], train[, 11], rfeControl=control)
results
```

```
##
## Recursive feature selection
```

```
##
## Outer resampling method: Cross-Validated (10 fold)
##
## Resampling performance over subset size:
##
## Variables Accuracy Kappa AccuracySD KappaSD Selected
##      4  0.7284 0.4564  0.06860  0.1375
##      8  0.7538 0.5078  0.05571  0.1113
##     10  0.7580 0.5162  0.06024  0.1197      *
```

```
## The top 5 variables (out of 10):
##      cp, oldpeak, exang, sex, thalach
```

```
print(results)
```

```
##
## Recursive feature selection
##
## Outer resampling method: Cross-Validated (10 fold)
##
## Resampling performance over subset size:
##
## Variables Accuracy Kappa AccuracySD KappaSD Selected
##      4  0.7284 0.4564  0.06860  0.1375
##      8  0.7538 0.5078  0.05571  0.1113
##     10  0.7580 0.5162  0.06024  0.1197      *
```

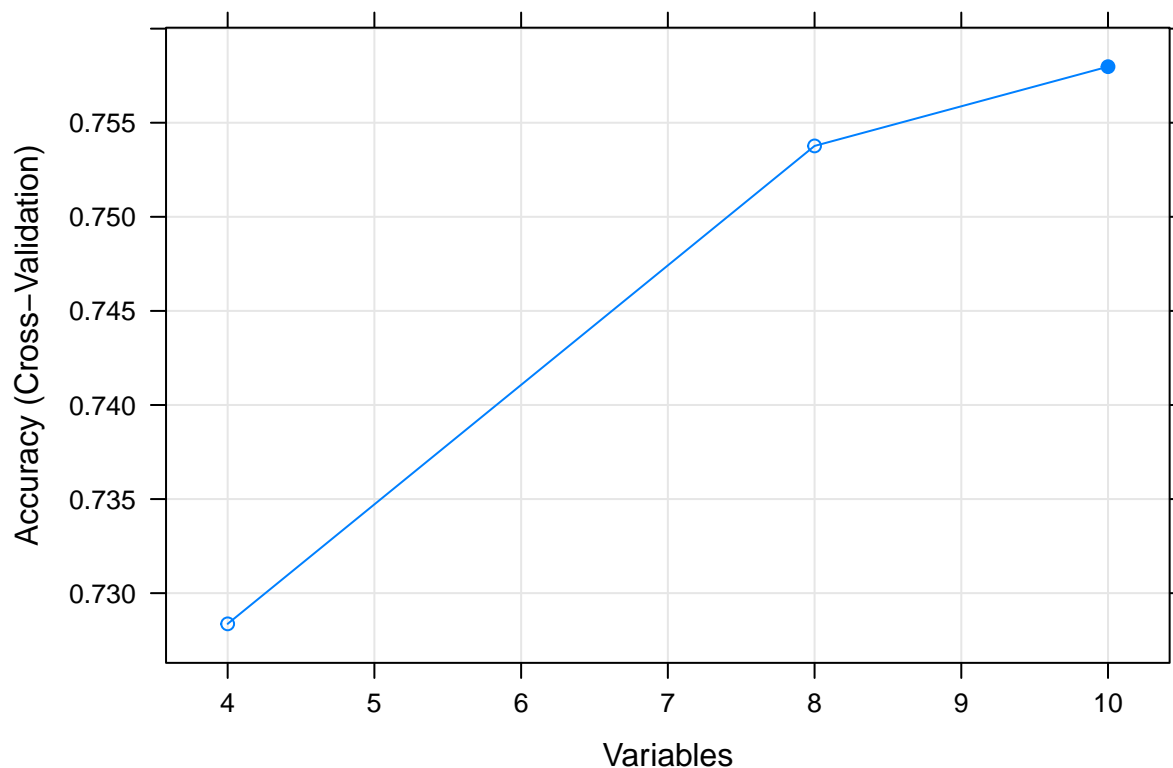
```
## The top 5 variables (out of 10):
##      cp, oldpeak, exang, sex, thalach
```

```
predictors(results)
```

```
## [1] "cp"      "oldpeak" "exang"    "sex"      "thalach"  "age"
## [7] "fbs"     "chol"    "restecg"  "trestbps"
```

```
plot(results, type = c("g", "o"))
```





Approach 5: Single layer neural network, using 10 fold, 3 repeat cross validation to tune the parameters size and decay.

```
fitControl = trainControl(method = "repeatedcv", number = 10, repeats = 3)
set.seed(100)
nnet.model = train(status ~ ., data = train, method = "nnet", trControl = fitControl, verbose = FALSE,
nnet.model
```

```
## Neural Network
##
## 471 samples
## 10 predictor
## 2 classes: 'healthy', 'disease'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 424, 424, 424, 424, 424, 424, ...
## Resampling results across tuning parameters:
##
##   size  decay  Accuracy  Kappa
##   1     0e+00  0.5762129  0.1388614
##   1     1e-04  0.5661360  0.1161913
##   1     1e-01  0.7755011  0.5522494
##   3     0e+00  0.6563483  0.3152133
##   3     1e-04  0.6554091  0.3066531
##   3     1e-01  0.7962181  0.5927021
##   5     0e+00  0.7206413  0.4429703
##   5     1e-04  0.7256090  0.4526095
##   5     1e-01  0.7826678  0.5655450
##
```

```
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were size = 3 and decay = 0.1.
```

```
nnet.result = predict(nnet.model, test)
confusionMatrix(test$status, nnet.result)
```

```
## Confusion Matrix and Statistics
```

```
##
```

```
##           Reference
```

```
## Prediction healthy disease
```

```
##   healthy      113      15
```

```
##   disease       37     104
```

```
##
```

```
##           Accuracy : 0.8067
```

```
##           95% CI : (0.7544, 0.8521)
```

```
##   No Information Rate : 0.5576
```

```
##   P-Value [Acc > NIR] : < 2.2e-16
```

```
##
```

```
##           Kappa : 0.6155
```

```
## Mcnemar's Test P-Value : 0.003589
```

```
##
```

```
##           Sensitivity : 0.7533
```

```
##           Specificity : 0.8739
```

```
##   Pos Pred Value : 0.8828
```

```
##   Neg Pred Value : 0.7376
```

```
##           Prevalence : 0.5576
```

```
##   Detection Rate : 0.4201
```

```
##   Detection Prevalence : 0.4758
```

```
##   Balanced Accuracy : 0.8136
```

```
##
```

```
##   'Positive' Class : healthy
```

```
##
```

```
set.seed(100)
```

```
smallnnet.model = train(status ~ sex +cp + thalach +exang +oldpeak, data = train, method = "nnet", trC
```

```
smallnnet.model
```

```
## Neural Network
```

```
##
```

```
## 471 samples
```

```
##   5 predictor
```

```
##   2 classes: 'healthy', 'disease'
```

```
##
```

```
## No pre-processing
```

```
## Resampling: Cross-Validated (10 fold, repeated 3 times)
```

```
## Summary of sample sizes: 424, 424, 424, 424, 424, 424, ...
```

```
## Resampling results across tuning parameters:
```

```
##
```

##	size	decay	Accuracy	Kappa
##	1	0e+00	0.5329357	0.04123052
##	1	1e-04	0.5230066	0.01914251
##	1	1e-01	0.7770981	0.55263622
##	3	0e+00	0.5857308	0.15125408
##	3	1e-04	0.6149688	0.21190317
##	3	1e-01	0.7875591	0.57553217
##	5	0e+00	0.5984794	0.17879022

```
##      5      1e-04  0.6477696  0.28080780
##      5      1e-01  0.7854925  0.57149119
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were size = 3 and decay = 0.1.
```

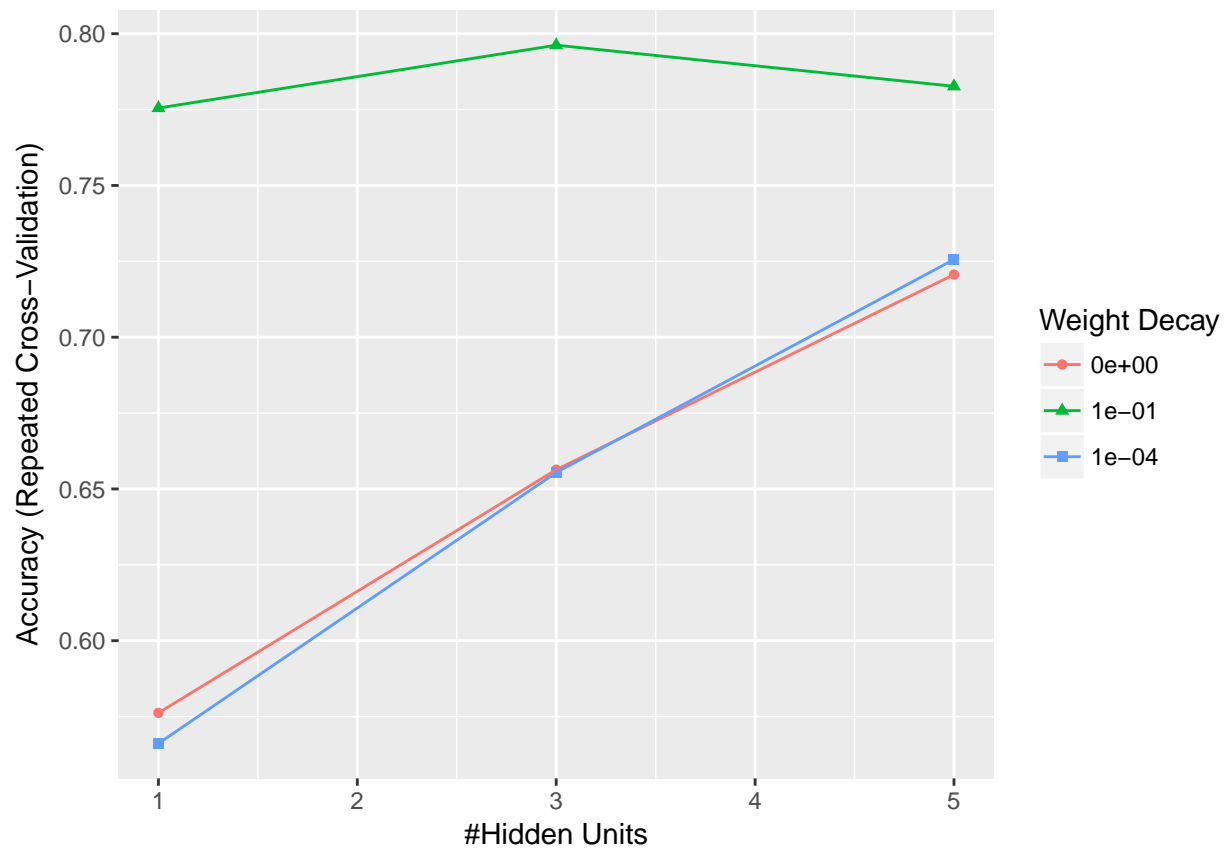
```
smallnnet.result = predict(smallnnet.model, test)
confusionMatrix(test$status, smallnnet.result)
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction healthy disease
##   healthy      111      17
##   disease       46      95
##
##              Accuracy : 0.7658
##              95% CI : (0.7106, 0.8151)
##   No Information Rate : 0.5836
##   P-Value [Acc > NIR] : 2.732e-10
##
##              Kappa : 0.5354
## Mcnemar's Test P-Value : 0.0004192
##
##              Sensitivity : 0.7070
##              Specificity : 0.8482
##              Pos Pred Value : 0.8672
##              Neg Pred Value : 0.6738
##              Prevalence : 0.5836
##              Detection Rate : 0.4126
##   Detection Prevalence : 0.4758
##              Balanced Accuracy : 0.7776
##
##              'Positive' Class : healthy
##
```

variable importance based on neural network model

```
ggplot(nnet.model)
```

```
## Warning: Ignoring unknown aesthetics: shape
```



```
importance = varImp(nnet.model, scale=FALSE)
importance
```

```
## nnet variable importance
##
##      Overall
## cp2      15.4286
## exang1   14.2669
## cp4      12.4885
## cp3      12.3490
## oldpeak  10.5689
## restecg1 10.1935
## sexmale   9.9430
## fbs1      7.8338
## restecg2  2.7411
## age       2.0853
## trestbps  1.0302
## chol      0.8389
## thalach   0.2322
```

```
plot(importance)
```

