CS5525 Final Project Code Submission

Jennifer Appiah-Kubi, Rebecca DeSipio, Ajinkya Fotedar

12/11/2021

Contents

Classification Methods **1**
Decision Trees 1
KNN 8
SVM 9
Logistic Regression 13

Classification Methods

First set the directory (path which contains the `heart.csv` data), and import any needed libraries

```
#setwd(" ") # uncomment to set working directory via code
library(tree)
library(randomForest)  # bootsrap/bagging & random forest
library(class)         # KNN
library(caret)         # SVM
library(ggplot2)
library(cowplot)
```

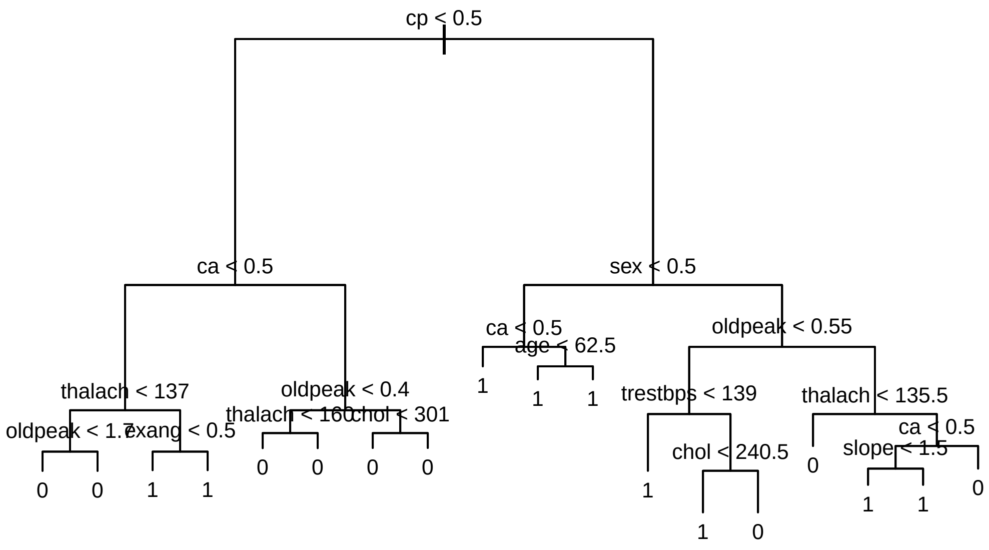
Decision Trees

```
# Read in and organize data
## -- Read data
heart <- read.csv("heart.csv")
Target <- as.factor(heart$target) # target heart rate

## -- Split into training and test sets
train <- sample(1:nrow(heart), 0.75*nrow(heart))
heart.test <- heart[-train, ]
Target.test <- Target[-train]

# ---------------------------------------------------------------------------- #
#                        Fit a Classification Tree                             #
# ---------------------------------------------------------------------------- #

# Fit a classification tree to the training data
set.seed(2441139)
tree.heart <- tree(Target~. -target, heart, subset=train)
summary(tree.heart)
```

```
##
```

```
## Classification tree:
## tree(formula = Target ~ . - target, data = heart, subset = train)
## Variables actually used in tree construction:
##  [1] "cp"       "ca"       "thalach"  "oldpeak"  "exang"    "chol"
##  [7] "sex"      "age"      "trestbps" "slope"
## Number of terminal nodes:  18
## Residual mean deviance:  0.4349 = 90.89 / 209
## Misclassification error rate: 0.1057 = 24 / 227
```

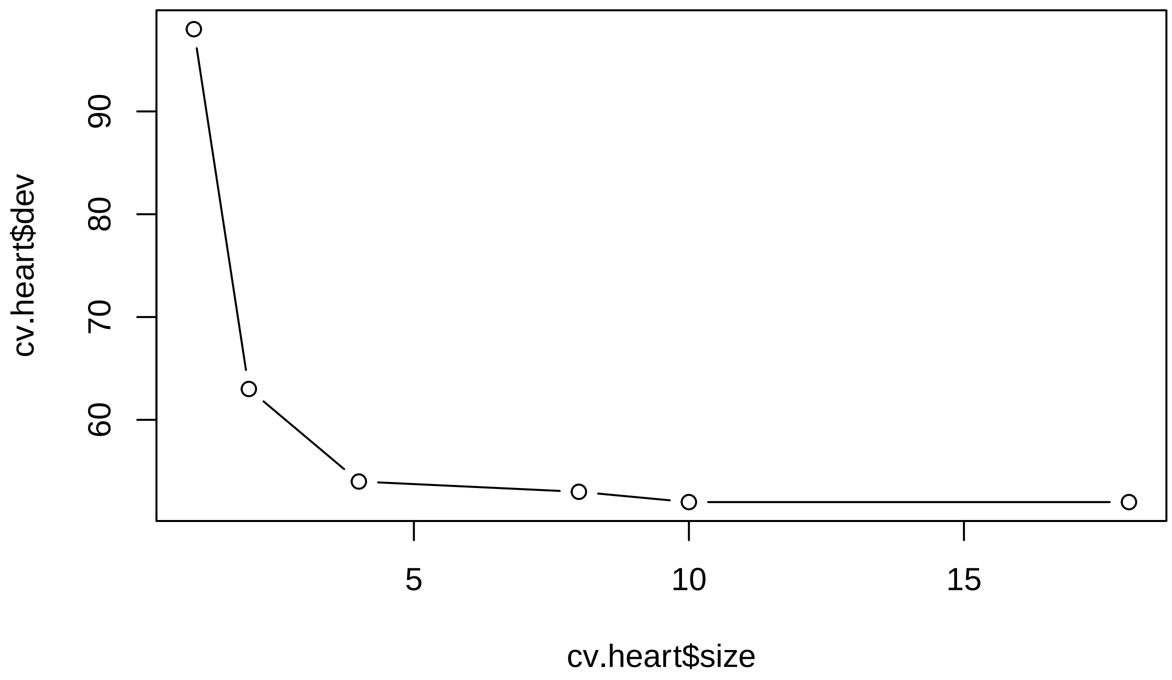
```
## -- Plot tree
plot(tree.heart)
text(tree.heart, pretty=1, cex=0.7)
```



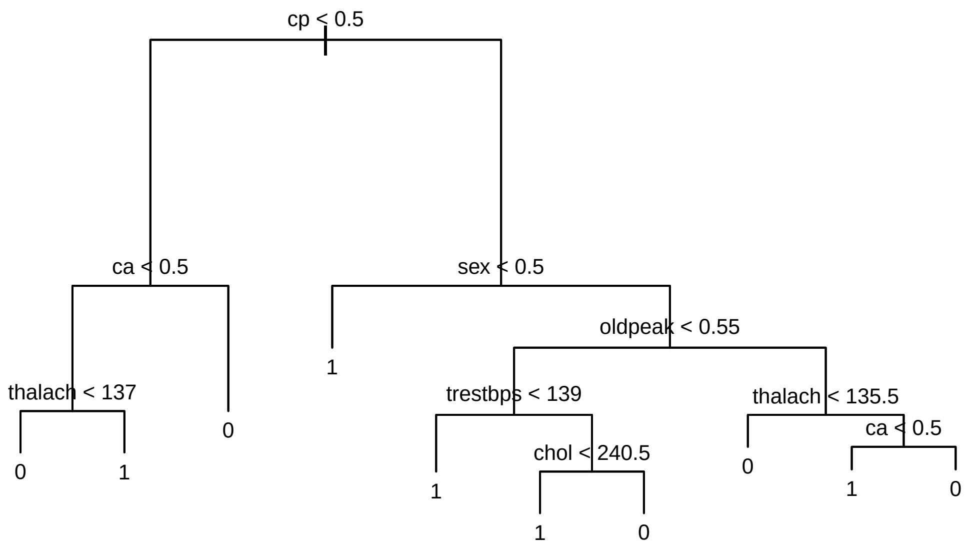
```
# Prune the classification tree
set.seed(2441139)
cv.heart <- cv.tree(tree.heart, FUN=prune.misclass)
cv.heart
```

```
## $size
## [1] 18 10  8  4  2  1
##
## $dev
## [1] 52 52 53 54 63 98
##
## $k
## [1]  -Inf  0.00  2.00  2.75  7.00 43.00
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```
plot(cv.heart$size, cv.heart$dev, type='b')
```



```r
prune.heart <- prune.misclass(tree.heart, best=10)
plot(prune.heart)
text(prune.heart, pretty=1, cex=0.65)
```



```r
# Predict using test set and pruned tree. Compare.
tree.pred <- predict(tree.heart, heart.test, type='class')   # test tree
prune.pred <- predict(prune.heart, heart.test, type='class') # pruned tree

table(prune.pred, Target.test)
```

```
##             Target.test
## prune.pred   0  1
##            0 25  9
##            1 17 25
```

```
table(tree.pred, Target.test)
```

```
##          Target.test
## tree.pred  0  1
##         0 25  9
##         1 17 25
```

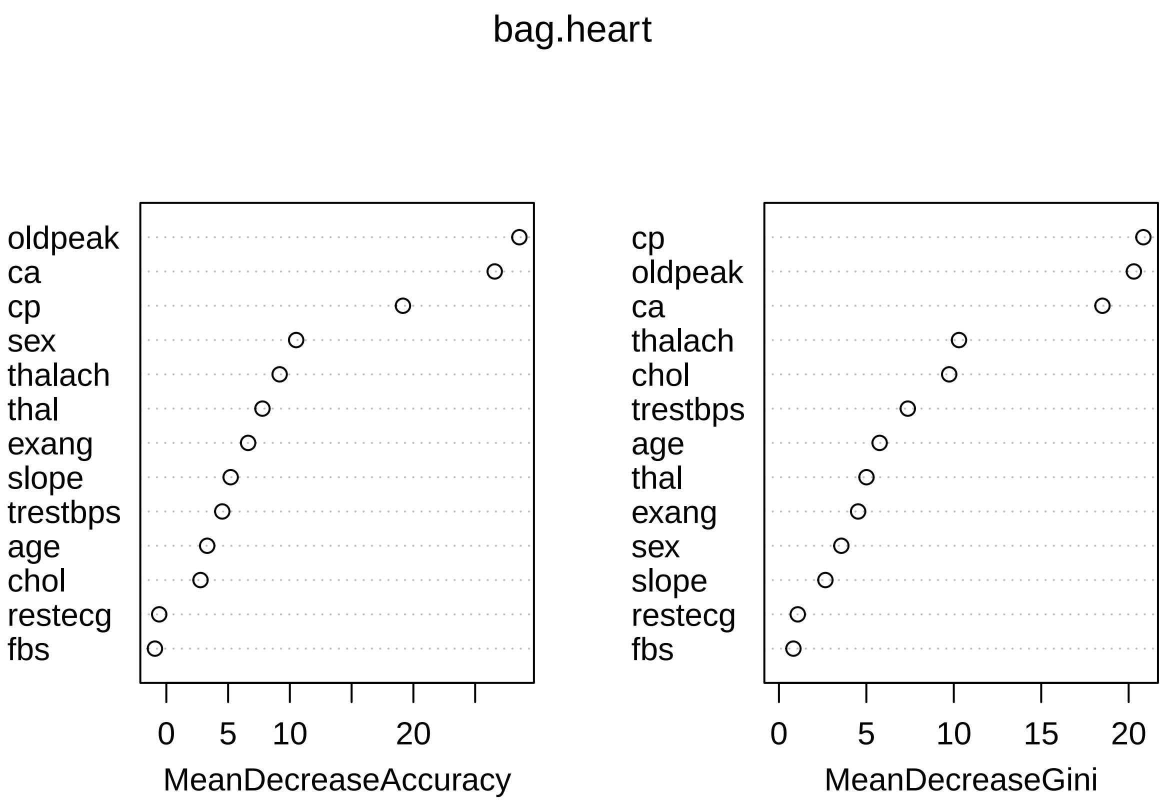
```
# ---------------------------------------------------------------------------- #
#                                  Bagging                                      #
# ---------------------------------------------------------------------------- #
set.seed(2441139)

# Perform bagging
bag.heart <- randomForest(as.factor(as.character(heart$target))~., data=heart,
                          subset=train, mtry=ncol(heart)-1,
                          importance=TRUE)
bag.heart
```

```
##
## Call:
##  randomForest(formula = as.factor(as.character(heart$target)) ~      ., data = heart, mtry = ncol(hea
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 13
##
##         OOB estimate of  error rate: 18.5%
## Confusion matrix:
##    0   1 class.error
## 0 71  25   0.2604167
## 1 17 114   0.1297710
```

```
# Predict on bagged tree
bag.pred <- predict(bag.heart, heart.test, type='class')
table(bag.pred, Target.test)
```

```
##         Target.test
## bag.pred  0  1
##        0 30  5
##        1 12 29
```

```
varImpPlot(bag.heart)
```



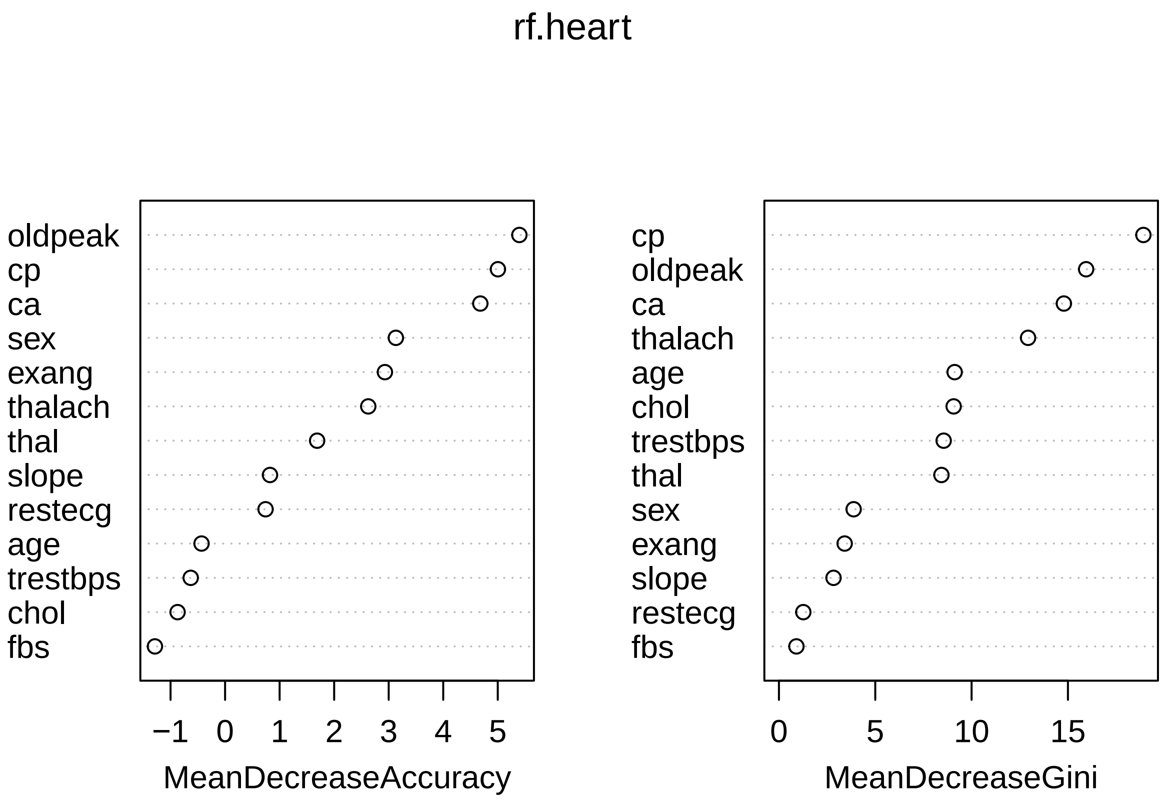
```
# ---------------------------------------------------------------------------- #
#                                Random Forest                                 #
# ---------------------------------------------------------------------------- #
set.seed(2441139)

# Perform Random Forest
rf.heart <- randomForest(as.factor(as.character(heart$target))~., data=heart,
                          subset=train, mtry=sqrt(ncol(heart)-1),
                          ntree=25, importance=TRUE)
rf.heart
```

```
##
## Call:
##  randomForest(formula = as.factor(as.character(heart$target)) ~      ., data = heart, mtry = sqrt(nc
##                Type of random forest: classification
##                      Number of trees: 25
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 20.26%
## Confusion matrix:
##    0   1 class.error
## 0 66  30   0.3125000
## 1 16 115   0.1221374
```

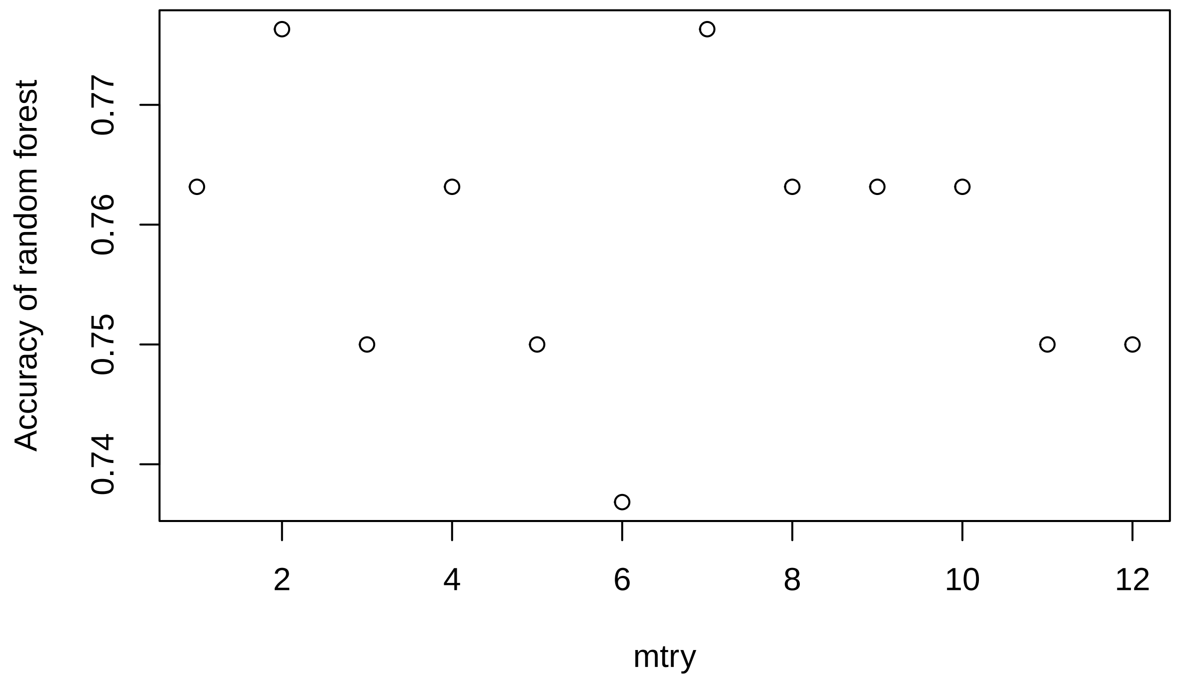
```
# Predict on the forest
rf.pred <- predict(rf.heart, heart.test, type='class')
table(rf.pred, Target.test)
```

```
##         Target.test
## rf.pred  0  1
##         0 29  3
##         1 13 31
```

```
varImpPlot(rf.heart)
```



```
# ---------------------------------------------------------------------------- #
# ---------------------------------------------------------------------------- #
#                     Determine Best Model (Random Forest)                     #
# ---------------------------------------------------------------------------- #
# ---------------------------------------------------------------------------- #
# Investigate how mtry affect the accuracy
Acc <- rep(0,ncol(heart)-2)
for (m in 1:(ncol(heart)-2)){
  set.seed(2441139)
  rf.heart <- randomForest(as.factor(as.character(heart$target))~., data=heart,
                          subset=train, mtry=m,
                          ntree=25)
  rf.pred <- predict(rf.heart, heart.test, type='class')
  t <- table(rf.pred, Target.test)
  acc <- sum(diag(t))/sum(t)
  Acc[m] <- acc
}
mbest <- which(Acc==max(Acc))
plot(1:(ncol(heart)-2), Acc, xlab='mtry', ylab='Accuracy of random forest') # include plot in final sub
```



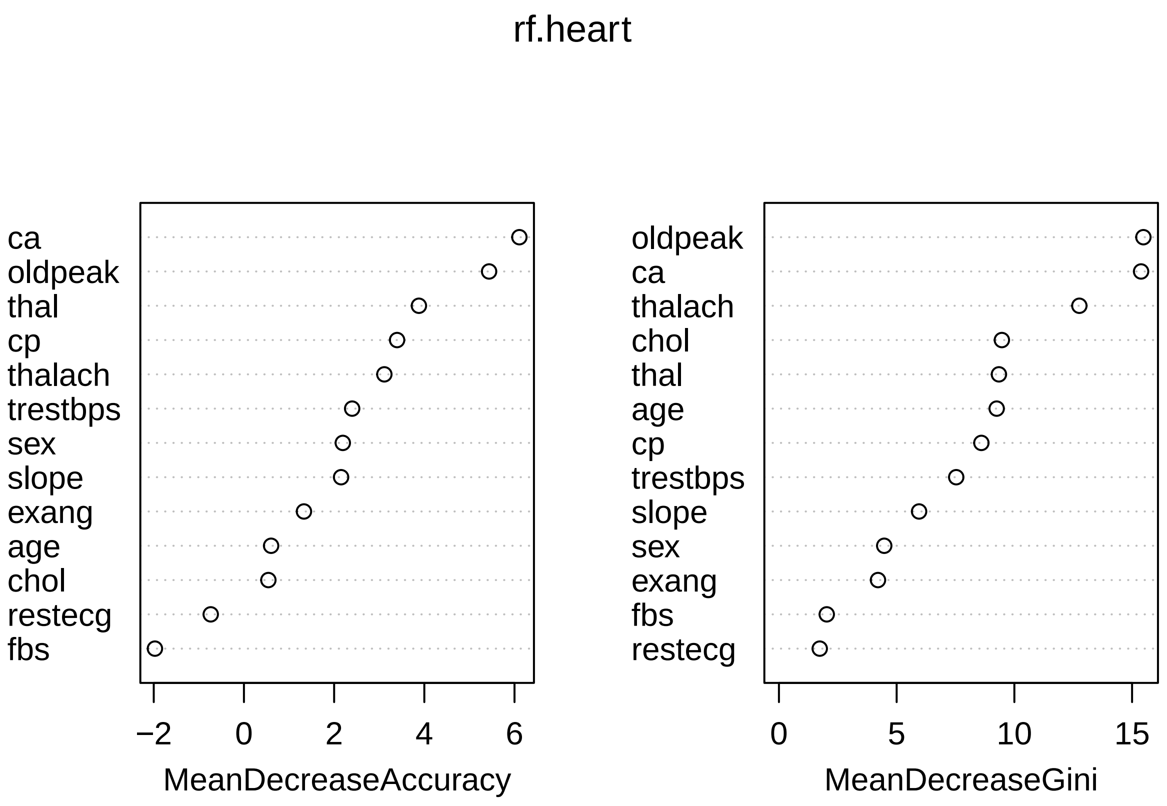
```r
# Now use the best value of m for the random forest
set.seed(2441139)
rf.heart <- randomForest(as.factor(as.character(heart$target))~., data=heart,
                         subset=train, mtry=mbest,
                         ntree=25, importance=TRUE)
rf.heart
```

```
##
## Call:
##  randomForest(formula = as.factor(as.character(heart$target)) ~      ., data = heart, mtry = mbest, 
##                Type of random forest: classification
##                      Number of trees: 25
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 17.62%
## Confusion matrix:
##    0   1 class.error
## 0 77  19   0.1979167
## 1 21 110   0.1603053
```

```r
# Predict on the forest
rf.pred <- predict(rf.heart, heart.test, type='class')
table(rf.pred, Target.test)
```

```
##          Target.test
## rf.pred   0  1
##         0 31  7
##         1 11 27
```

```r
varImpPlot(rf.heart)
```



KNN

```
# K-Nearest Neighbor
set.seed(2441139)

# Read data
heart <- read.csv("heart.csv")
Target <- as.factor(heart$target)

# Split into training and test sets
train <- sample(1:nrow(heart), 0.75*nrow(heart))
heart.test <- heart[-train, ]
Target.test <- Target[-train]

# Normalizing function
normalizer <-function(x){(x -min(x))/(max(x)-min(x))}

# Use normalized set to do cross-validation
trControl <- trainControl(method  = "cv",
                          number  = 5)
fit <- train(as.factor(target) ~ .,
             method     = "knn",
             tuneGrid   = expand.grid(k = 1:15),
             trControl  = trControl,
             metric     = "Accuracy",
             data       = as.data.frame(lapply(heart, normalizer)))

# For best value of k perform training
```

```
heart.norm <- as.data.frame(lapply(heart[,1:13], normalizer))

# Get training and test set
heart.norm.train <- heart.norm[train,]
heart.norm.test <- heart.norm[-train,]

# Get cl argument
cl <- as.factor(heart$target[train])

# Train and get confusion matrix
knn.heart <- knn(heart.norm.train, heart.norm.test, cl, k = 12, prob=TRUE)
cm <- table(knn.heart, Target.test)
(cm)
```

```
##            Target.test
## knn.heart   0  1
##           0 24 11
##           1  5 36
```

```
acc <- sum(diag(cm))/sum(cm)

# Train and get confusion matrix for un-normalized data
knn.heart <- knn(heart[train,1:13],heart[-train,1:13], cl, k = 5, prob=TRUE)
cm <- table(knn.heart, Target.test)
(cm)
```

```
##            Target.test
## knn.heart   0  1
##           0 19 14
##           1 10 33
```

```
acc <- sum(diag(cm))/sum(cm)
```

SVM

```
set.seed(2441139)

# splitting data into test and train
intrain <- createDataPartition(y = heart$target, p = 0.7, list = F)

training <- heart[intrain,]
testing <- heart[-intrain,]
training[["target"]] <- as.factor(training[["target"]])

dim(training)
```

```
## [1] 213  14
```

```
dim(testing)
```

```
## [1] 90 14
```

```
# set up repeated k-fold cross-validation
trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)

# model training with svm
```

```
svm.mod <- train(target ~ ., data = training, method = "svmLinear",
                 trControl = trctrl,
                 preProcess = c("center", "scale"),
                 tuneLength = 10)
svm.mod
```

```
## Support Vector Machines with Linear Kernel
##
## 213 samples
##  13 predictor
##   2 classes: '0', '1'
##
## Pre-processing: centered (13), scaled (13)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 192, 191, 192, 192, 191, 191, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8266955  0.6511675
##
## Tuning parameter 'C' was held constant at a value of 1
```

```
# predction using the above model
svm.pred <- predict(svm.mod, newdata = testing)
svm.pred
```

```
##  [1] 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 0 0 1 0 1 1 1 1 1 1 1 0 1 1 0 0 1 1 1 1 1
## [39] 1 1 1 1 1 1 1 0 0 1 1 1 1 0 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
## [77] 0 0 0 1 1 1 0 0 1 0 1 0 0 0
## Levels: 0 1
```

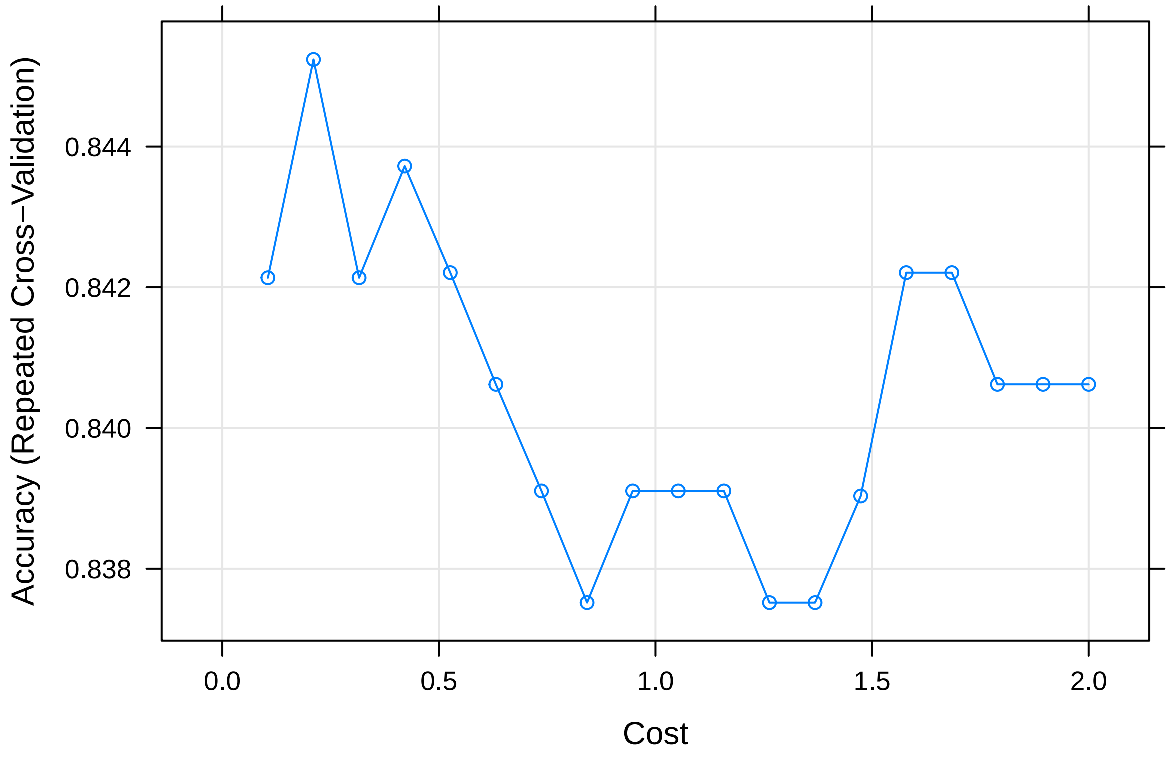
```
# accuracy of the trained model
confusionMatrix(table(svm.pred, testing$target))
```

```
## Confusion Matrix and Statistics
##
##
## svm.pred  0  1
##        0 28 11
##        1  9 42
##
##                Accuracy : 0.7778
##                  95% CI : (0.6779, 0.8587)
##     No Information Rate : 0.5889
##     P-Value [Acc > NIR] : 0.0001266
##
##                   Kappa : 0.5448
##
##  Mcnemar's Test P-Value : 0.8230633
##
##             Sensitivity : 0.7568
##             Specificity : 0.7925
##          Pos Pred Value : 0.7179
##          Neg Pred Value : 0.8235
##              Prevalence : 0.4111
##          Detection Rate : 0.3111
##    Detection Prevalence : 0.4333
##       Balanced Accuracy : 0.7746
##
##        'Positive' Class : 0
##
```

```
# costs for further tuning
grid <- expand.grid(C = seq(0, 2, length = 20))

svm.mod.tuned <- train(target ~ ., data = training, method = "svmLinear",
                      trControl = trctrl,
                      preProcess = c("center", "scale"),
                      tuneGrid = grid,
                      tuneLength = 10)
svm.mod.tuned
```

```
## Support Vector Machines with Linear Kernel
##
## 213 samples
##  13 predictor
##   2 classes: '0', '1'
##
## Pre-processing: centered (13), scaled (13)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 192, 191, 192, 192, 192, 192, ...
## Resampling results across tuning parameters:
##
##   C          Accuracy   Kappa
##   0.0000000        NaN        NaN
##   0.1052632  0.8421356  0.6805929
##   0.2105263  0.8452381  0.6870639
##   0.3157895  0.8421356  0.6810364
##   0.4210526  0.8437229  0.6842502
##   0.5263158  0.8422078  0.6814217
##   0.6315789  0.8406205  0.6783419
##   0.7368421  0.8391053  0.6752263
##   0.8421053  0.8375180  0.6721433
##   0.9473684  0.8391053  0.6753687
##   1.0526316  0.8391053  0.6753687
##   1.1578947  0.8391053  0.6753687
##   1.2631579  0.8375180  0.6721433
##   1.3684211  0.8375180  0.6721433
##   1.4736842  0.8390332  0.6751736
##   1.5789474  0.8422078  0.6815951
##   1.6842105  0.8422078  0.6815951
##   1.7894737  0.8406205  0.6785395
##   1.8947368  0.8406205  0.6785395
##   2.0000000  0.8406205  0.6785395
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was C = 0.2105263.
```

```
# accuracy plot with varying costs
plot(svm.mod.tuned)
```



```
# best cost value that maximizes model accuracy
svm.mod.tuned$bestTune
```

```
##           C
## 3 0.2105263
```

```
# prediction using tuned model
svm.pred.tuned <- predict(svm.mod.tuned, newdata = testing)
svm.pred.tuned
```

```
##  [1] 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1
## [39] 1 1 1 1 1 1 1 1 0 1 1 1 1 0 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
## [77] 0 0 0 1 1 1 0 0 1 0 1 0 0 0
## Levels: 0 1
```

```
# accuracy of the tuned model
confusionMatrix(table(svm.pred.tuned, testing$target))
```

```
## Confusion Matrix and Statistics
##
##
## svm.pred.tuned  0  1
##              0 28  5
##              1  9 48
##
##                Accuracy : 0.8444
##                  95% CI : (0.7528, 0.9123)
##     No Information Rate : 0.5889
##     P-Value [Acc > NIR] : 1.588e-07
##
##                   Kappa : 0.6734
##
```

```
##  Mcnemar's Test P-Value : 0.4227
##
##             Sensitivity : 0.7568
##             Specificity : 0.9057
##          Pos Pred Value : 0.8485
##          Neg Pred Value : 0.8421
##              Prevalence : 0.4111
##          Detection Rate : 0.3111
##    Detection Prevalence : 0.3667
##       Balanced Accuracy : 0.8312
##
##        'Positive' Class : 0
##
```

Logistic Regression

```
set.seed(2441139)

# pre-processing of data
str(heart)
```

```
## 'data.frame':    303 obs. of  14 variables:
##  $ age     : int  63 37 41 56 57 57 56 44 52 57 ...
##  $ sex     : int  1 1 0 1 0 1 0 1 1 1 ...
##  $ cp      : int  3 2 1 1 0 0 1 1 2 2 ...
##  $ trestbps: int  145 130 130 120 120 140 140 120 172 150 ...
##  $ chol    : int  233 250 204 236 354 192 294 263 199 168 ...
##  $ fbs     : int  1 0 0 0 0 0 0 0 1 0 ...
##  $ restecg : int  0 1 0 1 1 1 0 1 1 1 ...
##  $ thalach : int  150 187 172 178 163 148 153 173 162 174 ...
##  $ exang   : int  0 0 0 0 1 0 0 0 0 0 ...
##  $ oldpeak : num  2.3 3.5 1.4 0.8 0.6 0.4 1.3 0 0.5 1.6 ...
##  $ slope   : int  0 0 2 2 2 1 1 2 2 2 ...
##  $ ca      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ thal    : int  1 2 2 2 2 1 2 3 3 2 ...
##  $ target  : int  1 1 1 1 1 1 1 1 1 1 ...
```

```
heart$sex <- ifelse(test = heart$sex == 0, yes = "F", no = "M")
heart$sex <- as.factor(heart$sex)
heart$cp <- as.factor(heart$cp)
heart$fbs <- as.factor(heart$fbs)
heart$restecg <- as.factor(heart$restecg)
heart$exang <- as.factor(heart$exang)
heart$slope <- as.factor(heart$slope)
heart$ca <- as.factor(heart$ca)
heart$thal <- as.factor(heart$thal)
heart$age <- as.numeric(heart$age)
heart$trestbps <- as.numeric(heart$trestbps)
heart$chol <- as.numeric(heart$chol)
heart$thalach <- as.numeric(heart$thalach)
heart$target <- ifelse(test = heart$target == 0, yes = "Healthy", no = "Unhealthy")
heart$target <- as.factor(heart$target)

str(heart)
```

```
## 'data.frame':    303 obs. of  14 variables:
##  $ age     : num  63 37 41 56 57 57 56 44 52 57 ...
##  $ sex     : Factor w/ 2 levels "F","M": 2 2 1 2 1 2 1 2 2 2 ...
##  $ cp      : Factor w/ 4 levels "0","1","2","3": 4 3 2 2 1 1 2 2 3 3 ...
##  $ trestbps: num  145 130 130 120 120 140 140 120 172 150 ...
##  $ chol    : num  233 250 204 236 354 192 294 263 199 168 ...
##  $ fbs     : Factor w/ 2 levels "0","1": 2 1 1 1 1 1 1 1 2 1 ...
##  $ restecg : Factor w/ 3 levels "0","1","2": 1 2 1 2 2 2 1 2 2 2 ...
##  $ thalach : num  150 187 172 178 163 148 153 173 162 174 ...
##  $ exang   : Factor w/ 2 levels "0","1": 1 1 1 1 2 1 1 1 1 1 ...
##  $ oldpeak : num  2.3 3.5 1.4 0.8 0.6 0.4 1.3 0 0.5 1.6 ...
##  $ slope   : Factor w/ 3 levels "0","1","2": 1 1 3 3 3 2 2 3 3 3 ...
##  $ ca      : Factor w/ 5 levels "0","1","2","3",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ thal    : Factor w/ 4 levels "0","1","2","3": 2 3 3 3 3 2 3 4 4 3 ...
##  $ target  : Factor w/ 2 levels "Healthy","Unhealthy": 2 2 2 2 2 2 2 2 2 2 ...
```

```
# getting the number of samples by gender
xtabs(~ target + sex, data = heart)
```

```
##            sex
## target        F   M
##   Healthy    24 114
##   Unhealthy  72  93
```

```
# getting the number of samples by chest pain
xtabs(~ target + cp, data = heart)
```

```
##            cp
## target        0   1   2   3
##   Healthy   104   9  18   7
##   Unhealthy  39  41  69  16
```

```
# getting the number of samples by resting ECG
xtabs(~ target + restecg, data = heart)
```

```
##            restecg
## target       0  1  2
##   Healthy   79 56  3
##   Unhealthy 68 96  1
```

```
# simple logistic model
logistic <- glm(target ~ sex, data = heart, family = "binomial")
summary(logistic)
```

```
##
## Call:
## glm(formula = target ~ sex, family = "binomial", data = heart)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6651  -1.0923   0.7585   1.2650   1.2650
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   1.0986     0.2357   4.661 3.15e-06 ***
## sexM         -1.3022     0.2740  -4.752 2.01e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 417.64  on 302  degrees of freedom
## Residual deviance: 392.80  on 301  degrees of freedom
## AIC: 396.8
##
## Number of Fisher Scoring iterations: 4
```

```
R_sq_1 <- 1 - logistic$deviance / logistic$null.deviance
R_sq_1
```

```
## [1] 0.05947945
```

```
BIC_1 <- logistic$deviance + 2 * log(dim(heart)[1])
BIC_1
```

```
## [1] 404.2246
```

```
# complex logistic model
logistic <- glm(target ~ ., data = heart, family = "binomial")
summary(logistic)
```

```
##
## Call:
## glm(formula = target ~ ., family = "binomial", data = heart)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.9459  -0.2738   0.1012   0.4515   3.1248
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.179045   3.705420   0.048 0.961461
## age          0.027819   0.025428   1.094 0.273938
## sexM        -1.862297   0.570844  -3.262 0.001105 **
## cp1          0.864708   0.578000   1.496 0.134645
## cp2          2.003186   0.529356   3.784 0.000154 ***
## cp3          2.417107   0.719242   3.361 0.000778 ***
## trestbps    -0.026162   0.011943  -2.191 0.028481 *
## chol        -0.004291   0.004245  -1.011 0.312053
## fbs1         0.445666   0.587977   0.758 0.448472
## restecg1     0.460582   0.399615   1.153 0.249089
## restecg2    -0.714204   2.768873  -0.258 0.796453
## thalach      0.020055   0.011859   1.691 0.090820 .
## exang1      -0.779111   0.451839  -1.724 0.084652 .
## oldpeak     -0.397174   0.242346  -1.639 0.101239
## slope1      -0.775084   0.880495  -0.880 0.378707
## slope2       0.689965   0.947657   0.728 0.466568
## ca1         -2.342301   0.527416  -4.441 8.95e-06 ***
## ca2         -3.483178   0.811640  -4.292 1.77e-05 ***
## ca3         -2.247144   0.937629  -2.397 0.016547 *
## ca4          1.267961   1.720014   0.737 0.461013
## thal1        2.637558   2.684285   0.983 0.325808
## thal2        2.367747   2.596159   0.912 0.361759
## thal3        0.915115   2.600380   0.352 0.724901
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 417.64  on 302  degrees of freedom
## Residual deviance: 179.63  on 280  degrees of freedom
## AIC: 225.63
##
## Number of Fisher Scoring iterations: 6
```

```
R_sq_2 <- 1 - logistic$deviance / logistic$null.deviance
R_sq_2
```

```
## [1] 0.569889
```

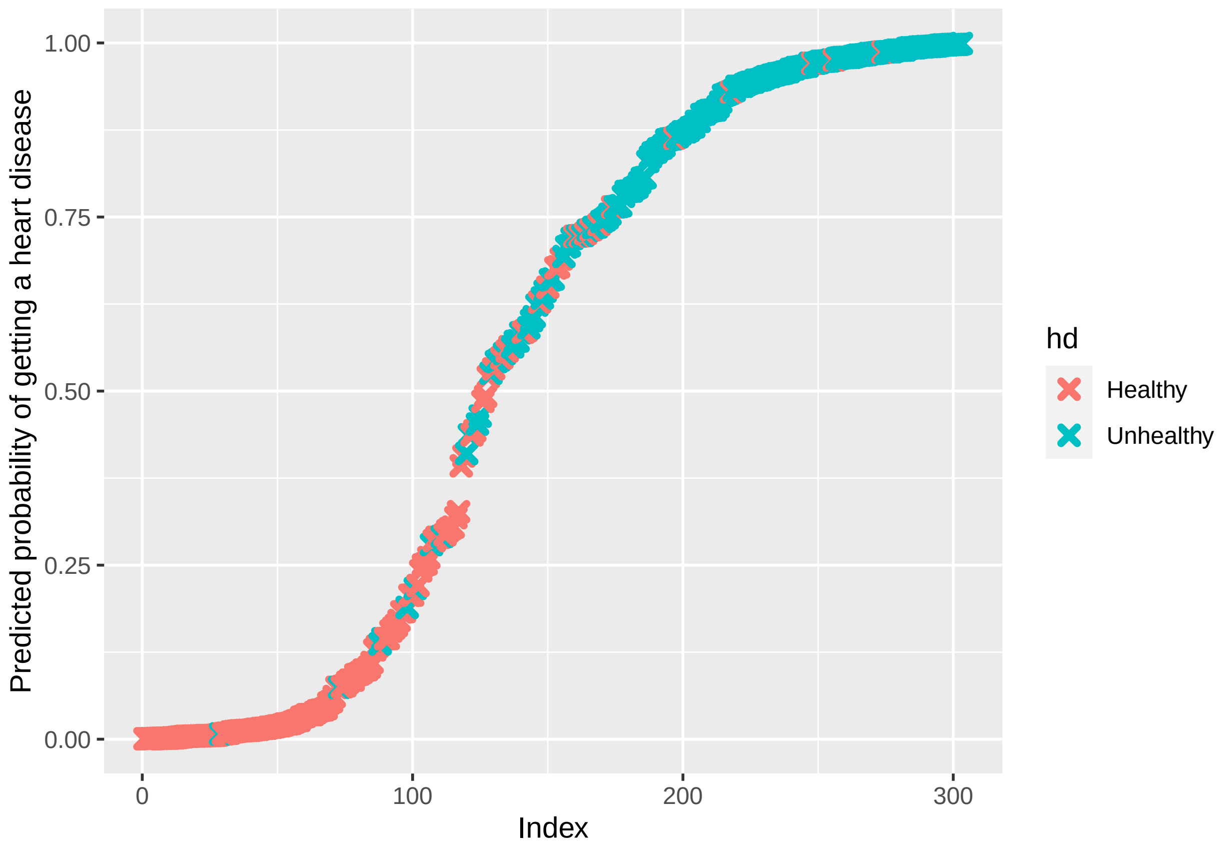
```
BIC_2 <- logistic$deviance + 14 * log(dim(heart)[1])
BIC_2
```

```
## [1] 259.623
```

```
# why age isn't of statistical significance
median(heart$age)
```

```
## [1] 55
```

```
# plotting the probability of getting a heart disease
predict.hd <- data.frame(prob.of.hd = logistic$fitted.values, hd = heart$target)
predict.hd <- predict.hd[order(predict.hd$prob.of.hd, decreasing = FALSE), ]
predict.hd$rank <- 1:nrow(predict.hd)

ggplot(data = predict.hd, aes(x = rank, y = prob.of.hd)) +
  geom_point(aes(color = hd), alpha = 1, shape = 4, stroke = 2) +
  xlab("Index") +
  ylab("Predicted probability of getting a heart disease")
```



```r
# predicting model accuracy
log.mod <- glm(target ~ ., data = training, family = "binomial")
log.pred <- predict(log.mod, newdata = testing, type = "response")
log.pred <- ifelse(log.pred > 0.5, 1, 0)
confusionMatrix(table(log.pred, testing$target))
```

```
## Confusion Matrix and Statistics
##
##
## log.pred  0  1
##        0 28 11
##        1  9 42
##
##                Accuracy : 0.7778
##                  95% CI : (0.6779, 0.8587)
##     No Information Rate : 0.5889
##     P-Value [Acc > NIR] : 0.0001266
##
##                   Kappa : 0.5448
##
##  Mcnemar's Test P-Value : 0.8230633
##
##             Sensitivity : 0.7568
##             Specificity : 0.7925
##          Pos Pred Value : 0.7179
##          Neg Pred Value : 0.8235
##              Prevalence : 0.4111
##          Detection Rate : 0.3111
##    Detection Prevalence : 0.4333
##       Balanced Accuracy : 0.7746
##
##        'Positive' Class : 0
##
```