CS5525 Final Project Code Submission

Jennifer Appiah-Kubi, Rebecca DeSipio, Ajinkya Fotedar

12/11/2021

Contents

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
```

```
## randomForest 4.6-14
```

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

```
library(class)         # KNN
```

```
## Warning: package 'class' was built under R version 4.1.2
```

```
library(caret)         # SVM
```

```
## Warning: package 'caret' was built under R version 4.1.2
```

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

```
##
## Attaching package: 'ggplot2'
```

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

## Loading required package: lattice
```

```
library(glmnet)        # logistic regression
```

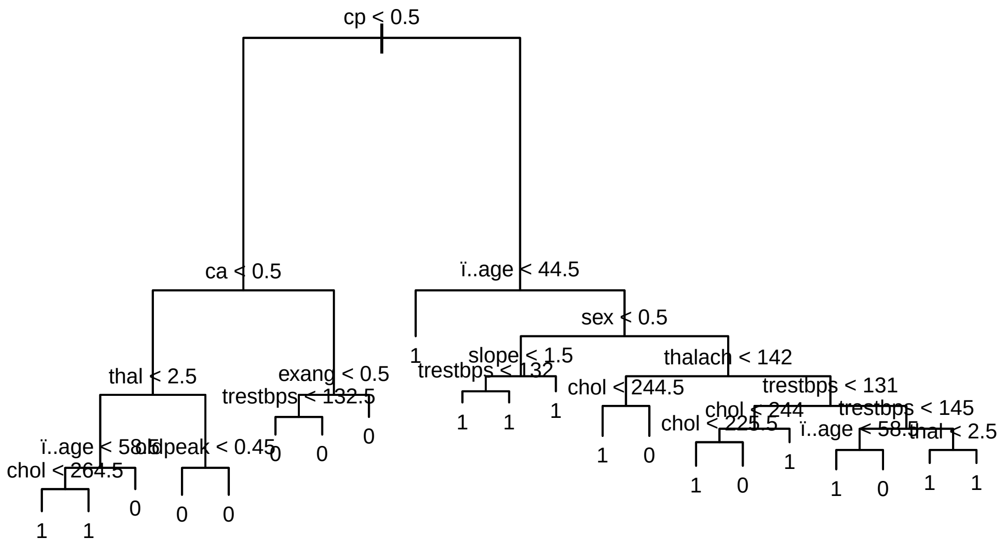
```
## Loading required package: Matrix

## Loaded glmnet 4.1-2
```

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"       "thal"     "ï..age"   "chol"     "oldpeak"
##  [7] "exang"    "trestbps" "sex"      "slope"    "thalach"
## Number of terminal nodes:  21
## Residual mean deviance:  0.379 = 78.07 / 206
## Misclassification error rate: 0.1013 = 23 / 227
```

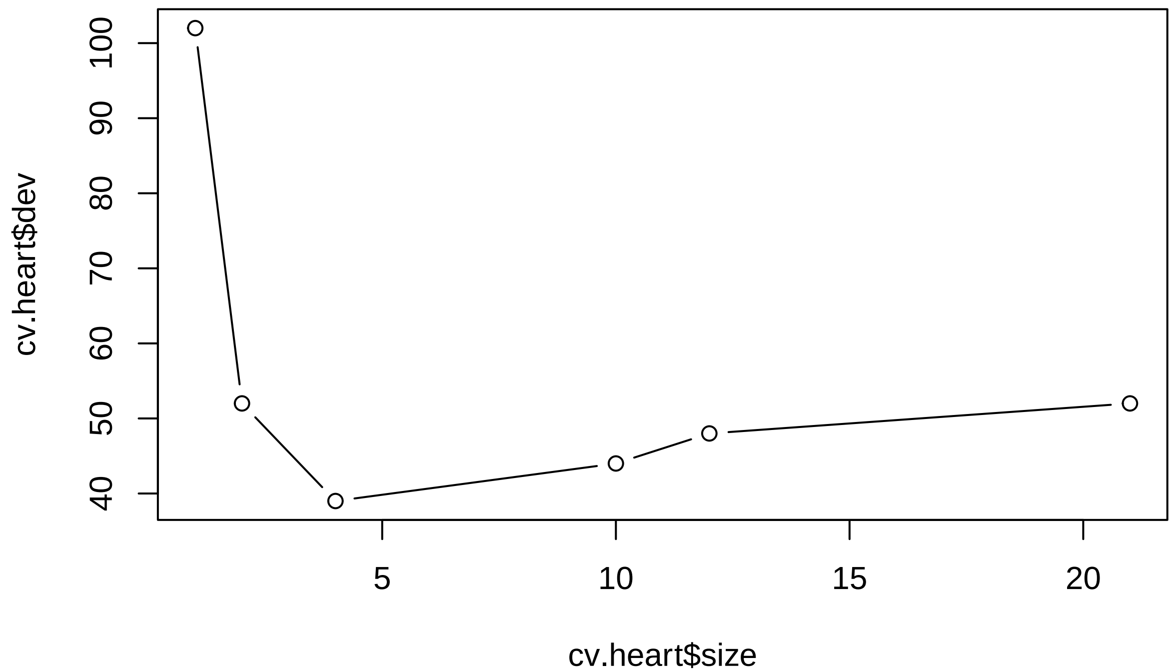
```
## -- Plot tree
plot(tree.heart)
text(tree.heart, pretty=1, cex=0.7)
```



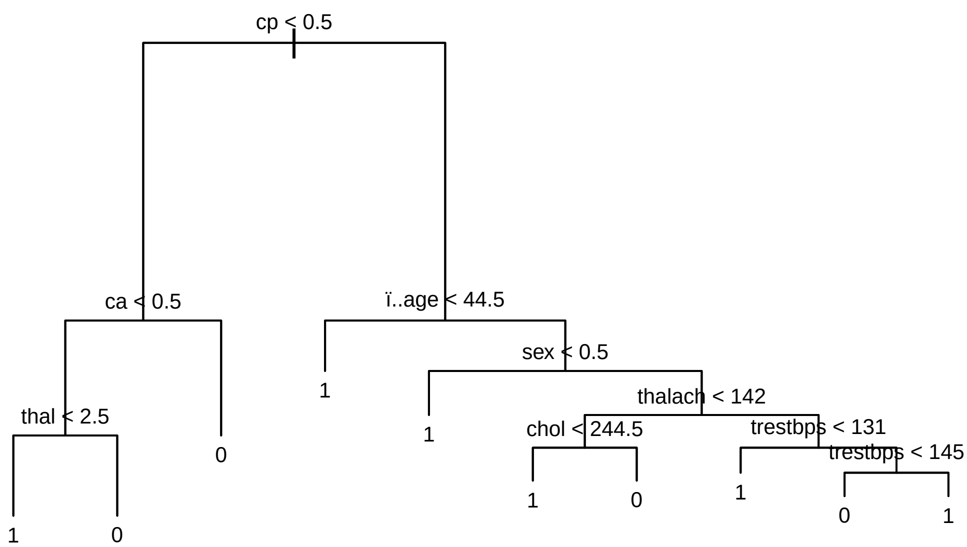

```r
# Prune the classification tree
set.seed(2441139)
cv.heart <- cv.tree(tree.heart, FUN=prune.misclass)
cv.heart
```

```
## $size
## [1] 21 12 10  4  2  1
##
## $dev
## [1]  52  48  44  39  52 102
##
## $k
## [1]      -Inf  0.000000  1.000000  1.833333  7.500000 51.000000
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

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




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




```
# 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 24  6
##          1 12 34
```

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

```
##          Target.test
## tree.pred  0  1
##         0 25 10
##         1 11 30
```

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

# Perform bagging
```

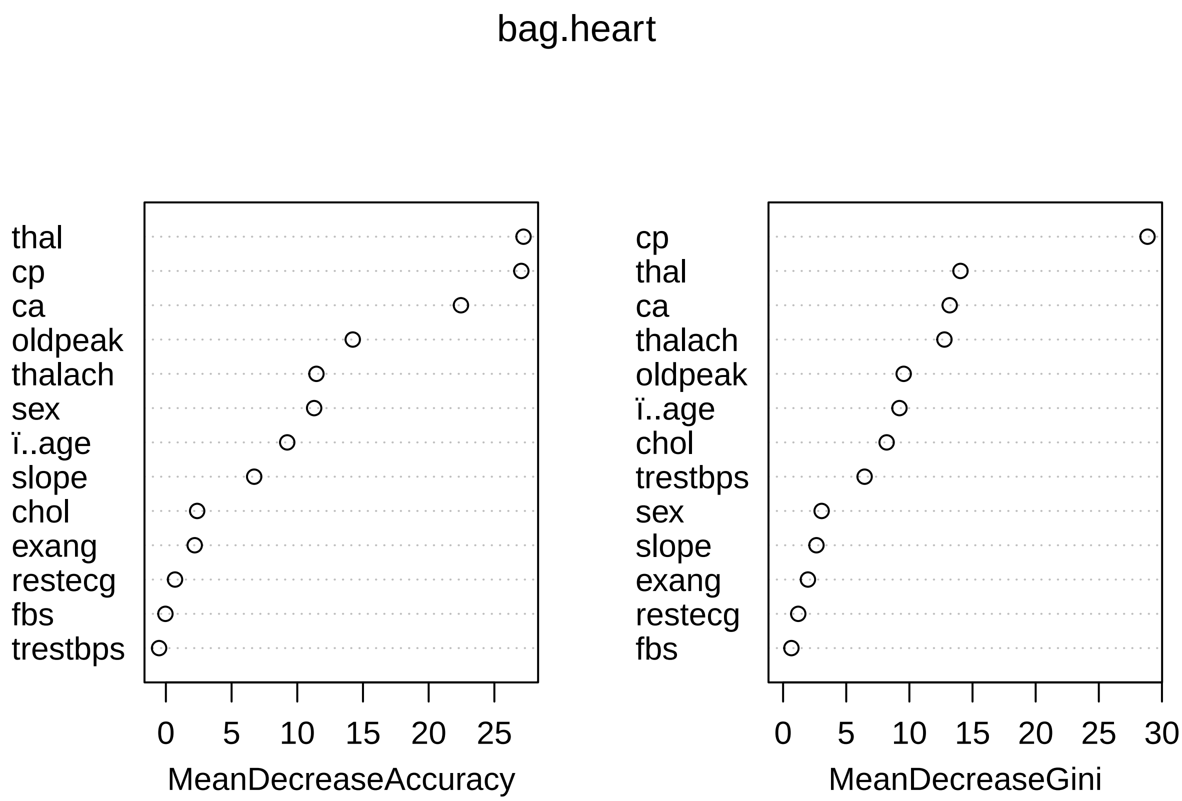
```r
bag.heart <- randomForest(as.factor(as.character(heart$target))~., data=heart,
                          subset=train, mtry=ncol(heart)-1,
                          importance=TRUE)
bag.heart
```

```
##
## 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: 19.82%
## Confusion matrix:
##    0   1 class.error
## 0 77  25    0.245098
## 1 20 105    0.160000
```

```r
# 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 25  6
##        1 11 34
```

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



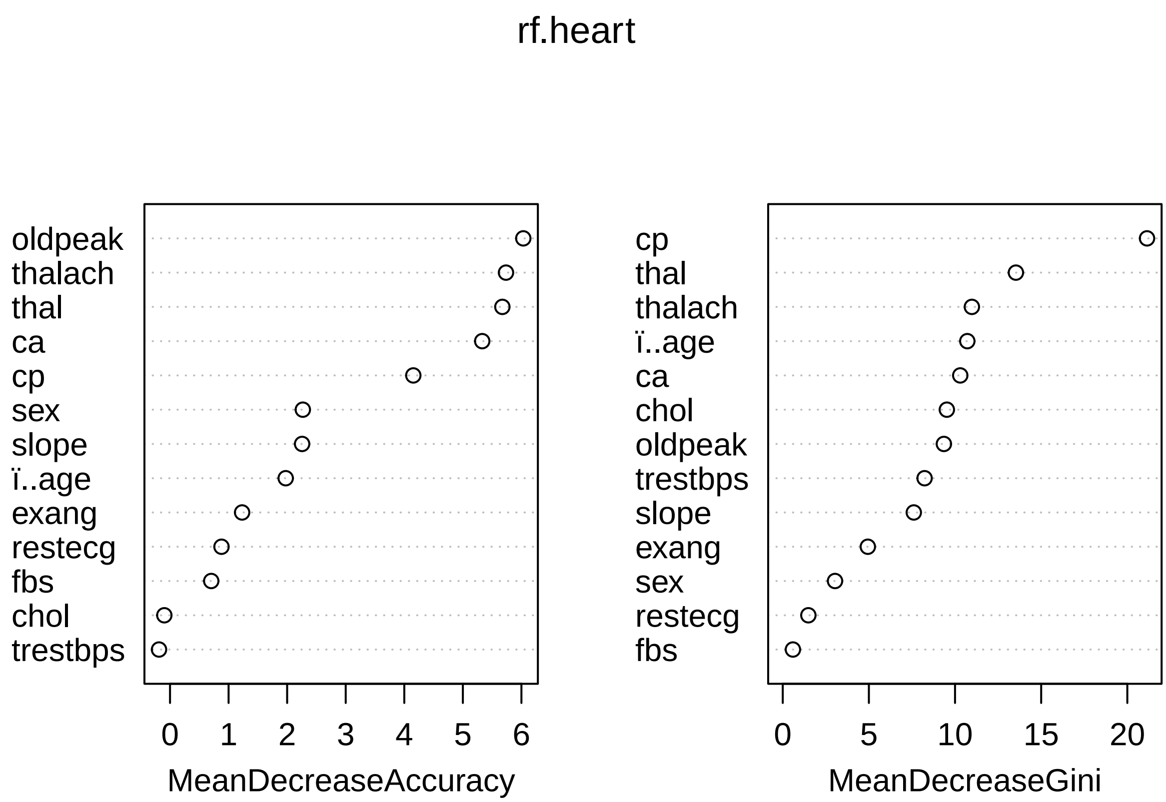

```
# ---------------------------------------------------------------------------- #
#                               Random Forest                                  #
# ---------------------------------------------------------------------------- #
set.seed(2441139)

# 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 76  26    0.254902
## 1 20 105    0.160000
```

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

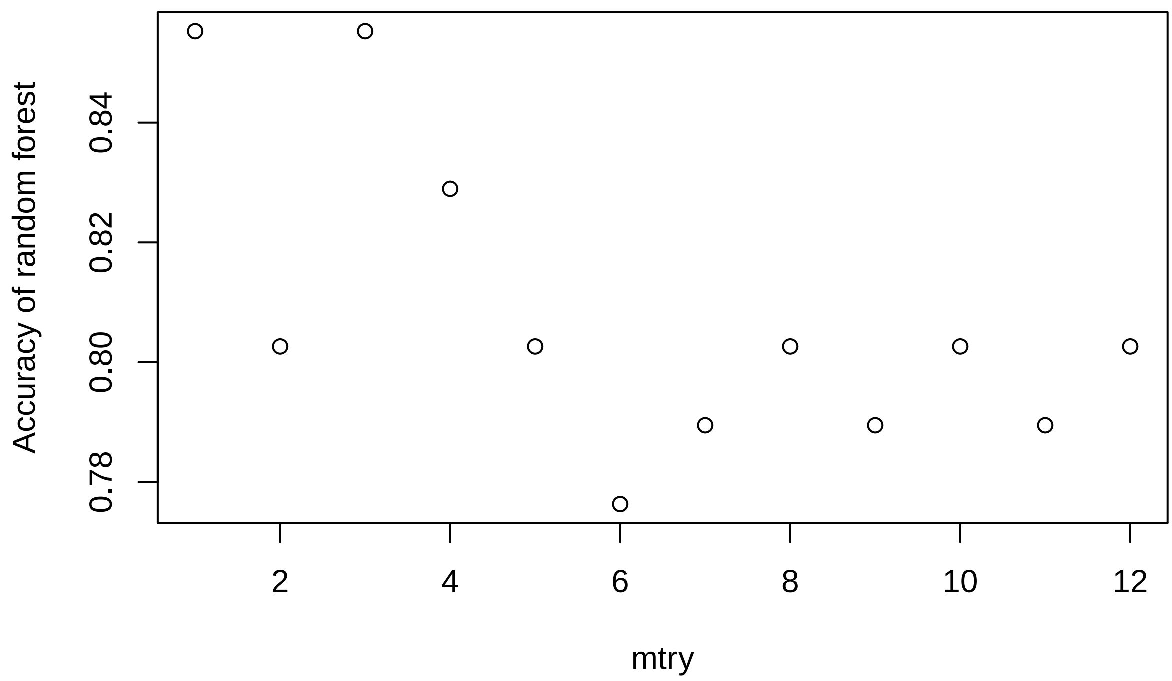
```
##          Target.test
## rf.pred   0  1
##         0 30  7
##         1  6 33
```

```
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
```




```
# 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
```

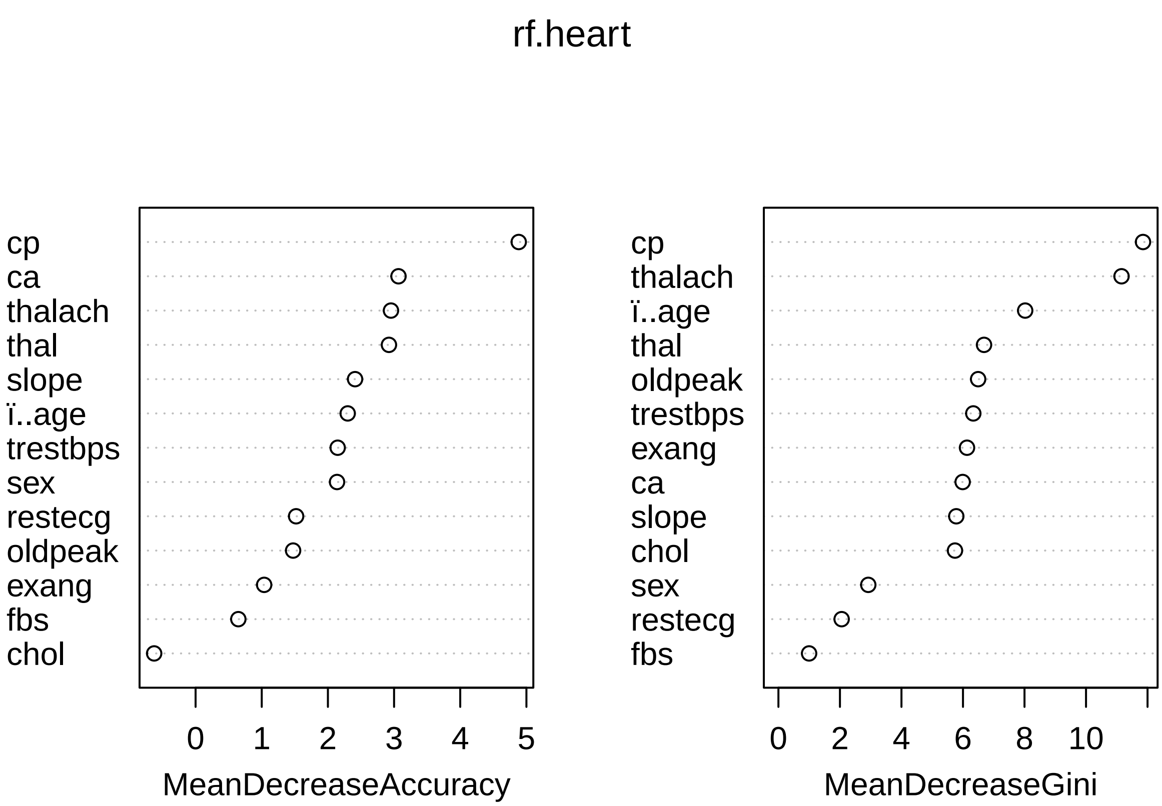
```
##
## Call:
##  randomForest(formula = as.factor(as.character(heart$target)) ~      ., data = heart, mtry = mbest, 
##                Type of random forest: classification
##                      Number of trees: 25
## No. of variables tried at each split: 1
##
##         OOB estimate of  error rate: 22.03%
## Confusion matrix:
##    0   1 class.error
## 0 76  26    0.254902
```

```
## 1 24 101     0.192000
```

```
# 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  5
##         1  7 35
```

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



KNN

```
# K-Nearest Neighbor
cl <- as.factor(heart$target[train])
knn.heart <- knn(heart[train,], heart.test, cl, k = 5, prob=TRUE)
table(knn.heart, Target.test)
```

```
##            Target.test
## knn.heart   0  1
##           0 18  7
##           1 18 33
```

SVM

```r
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
```

```r
dim(testing)
```

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

```r
# model training with svm
trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)
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
```

```r
# 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 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
```

```r
# 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
##
```

```r
# costs for further tuning with 10-fold cross-validation
grid <- expand.grid(C = c(0, 0.01, 0.05, 0.1, 0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75, 2, 5))

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

```
## Warning: model fit failed for Fold01.Rep1: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold02.Rep1: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold03.Rep1: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold04.Rep1: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters
```

```
## Warning: model fit failed for Fold05.Rep1: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold06.Rep1: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold07.Rep1: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold08.Rep1: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold09.Rep1: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold10.Rep1: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold01.Rep2: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold02.Rep2: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold03.Rep2: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold04.Rep2: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold05.Rep2: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold06.Rep2: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold07.Rep2: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold08.Rep2: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold09.Rep2: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold10.Rep2: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold01.Rep3: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters
```

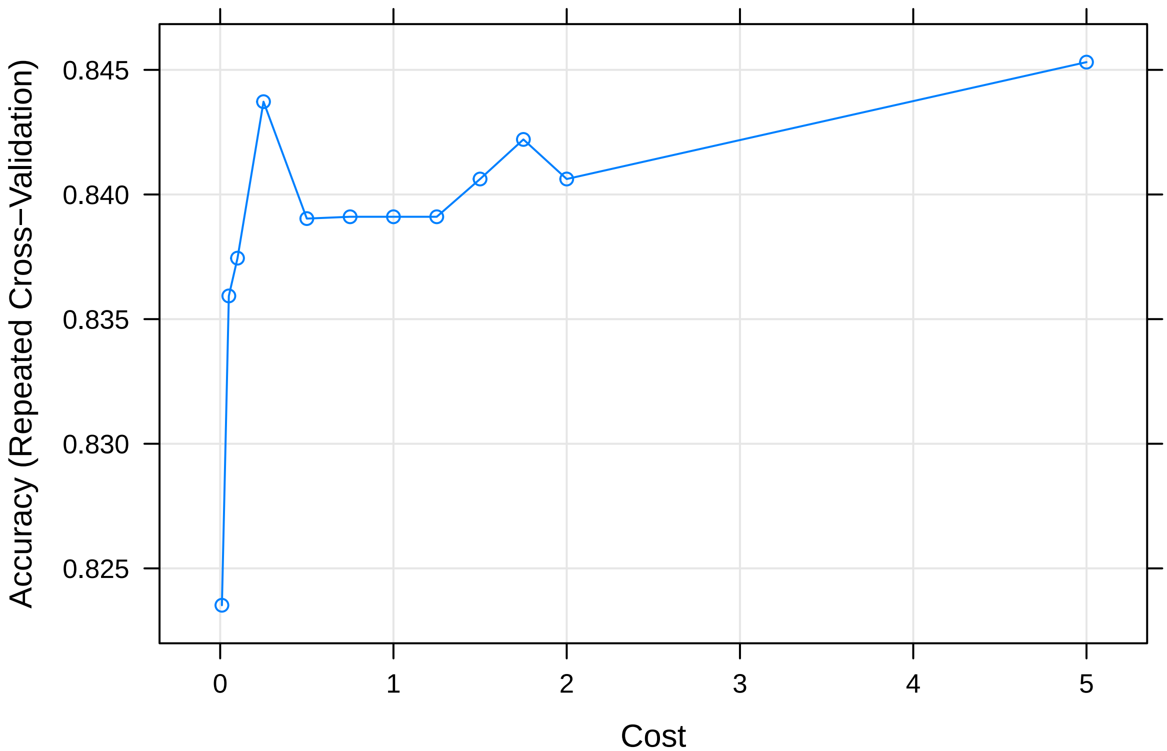
```
## Warning: model fit failed for Fold02.Rep3: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold03.Rep3: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold04.Rep3: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold05.Rep3: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold06.Rep3: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold07.Rep3: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold08.Rep3: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold09.Rep3: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold10.Rep3: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :
## There were missing values in resampled performance measures.

## Warning in train.default(x, y, weights = w, ...): missing values found in
## aggregated results
```

```
svm.mod.grid
```

```
## 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
##   0.00        NaN        NaN
##   0.01  0.8235209  0.6414705
##   0.05  0.8359307  0.6677455
##   0.10  0.8374459  0.6712576
```

```
##   0.25  0.8437229  0.6843572
##   0.50  0.8390332  0.6749413
##   0.75  0.8391053  0.6752263
##   1.00  0.8391053  0.6753687
##   1.25  0.8391053  0.6753687
##   1.50  0.8406205  0.6783406
##   1.75  0.8422078  0.6815951
##   2.00  0.8406205  0.6785395
##   5.00  0.8453102  0.6879521
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was C = 5.
```

```r
# accuracy plot of tuned model
plot(svm.mod.grid)
```




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

```
##  [1] 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 0 0 1 0 1 1 1 1 1 1 1 1 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 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.grid, testing$target))
```

```
## Confusion Matrix and Statistics
##
##
## svm.pred.grid  0  1
##             0 28 10
##             1  9 43
##
##                Accuracy : 0.7889
##                  95% CI : (0.6901, 0.8679)
##     No Information Rate : 0.5889
##     P-Value [Acc > NIR] : 4.918e-05
##
##                   Kappa : 0.5658
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.7568
##             Specificity : 0.8113
##          Pos Pred Value : 0.7368
##          Neg Pred Value : 0.8269
##              Prevalence : 0.4111
##          Detection Rate : 0.3111
##    Detection Prevalence : 0.4222
##       Balanced Accuracy : 0.7840
##
##        'Positive' Class : 0
##
```

Logistic Regression

```
set.seed(2441139)
# Organize data to get training and test data

X <- as.matrix(heart, c("age", "sex", "cp", "trestbps", "chol", "fbs",
                           "restecg", "thalach", "exang", "oldpeak", "slope",
                           "ca", "thal"))
```

```
## Warning in if (rownames.force %in% FALSE) NULL else if (rownames.force %in% :
## the condition has length > 1 and only the first element will be used
```

```
y <- heart$target

n <- nrow(X)
train_rows <- sample(1:n, n * 0.7)

X.train <- X[train_rows,]
X.test <- X[-train_rows,]
```

```r
y.train <- y[train_rows]
y.test <- y[-train_rows]

dim(X.train)
```

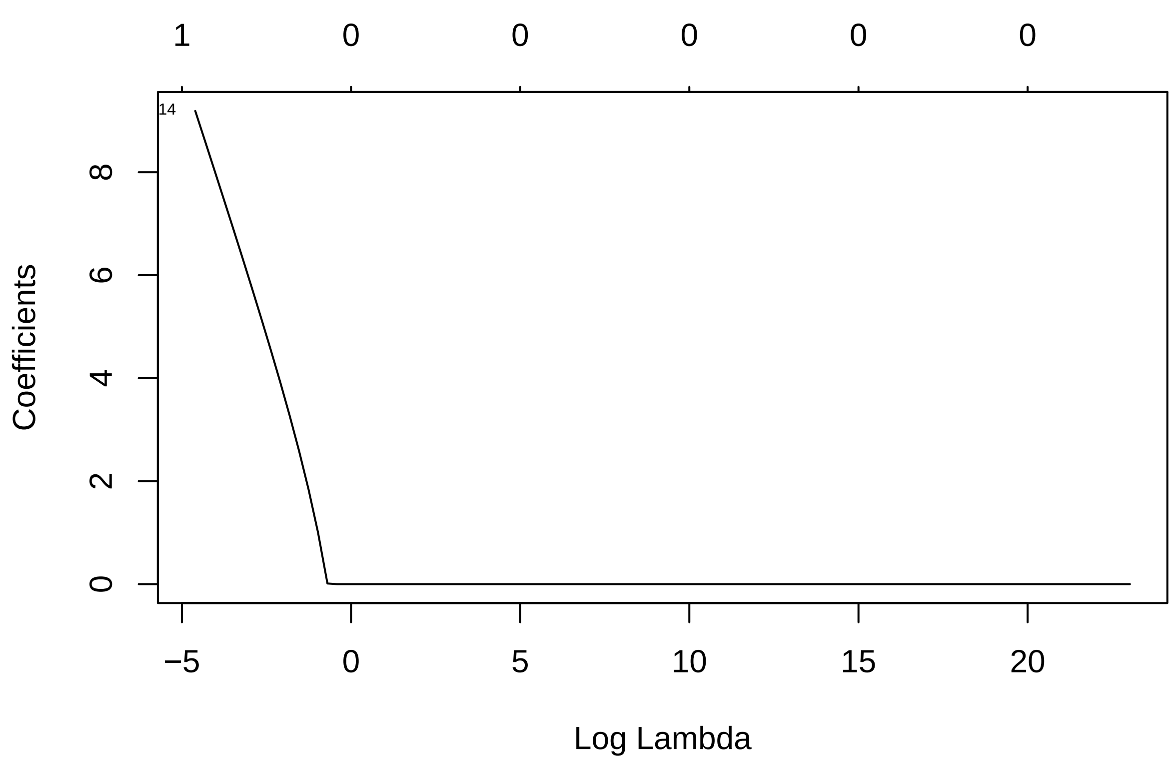
```
## [1] 212  14
```

```r
dim(X.test)
```

```
## [1] 91 14
```

```r
grid <- 10^seq(10, -2, length = 100)

# ------------------------------------------------------------------------------ #
#                                     Lasso                                      #
# ------------------------------------------------------------------------------ #

# lasso model
lasso.mod <- glmnet(X.train, as.factor(y.train), alpha = 1, lambda = grid,
                    family = "binomial")
plot(lasso.mod, xvar = "lambda", label = T)
```

```
## Warning in plotCoef(x$beta, lambda = x$lambda, df = x$df, dev = x$dev.ratio, : 1
## or less nonzero coefficients; glmnet plot is not meaningful
```



```
# cross-validation for lambda
cv.out <- cv.glmnet(X.train, as.factor(y.train), family = "binomial", alpha = 1,
                    type.measure = "class")
bestlam <- cv.out$lambda.min
bestlam
```

```
## [1] 0.4144432
```

```
# coefficients of the best model
best.lasso.mod <- glmnet(X.train, as.factor(y.train), alpha = 1, lambda = bestlam,
                         family = "binomial")
coef(best.lasso.mod)
```

```
## 15 x 1 sparse Matrix of class "dgCMatrix"
##                     s0
## (Intercept) -0.2468234
## ï..age       .
## sex          .
## cp           .
## trestbps     .
## chol         .
## fbs          .
## restecg      .
## thalach      .
## exang        .
## oldpeak      .
## slope        .
## ca           .
## thal         .
## target       0.6873644
```

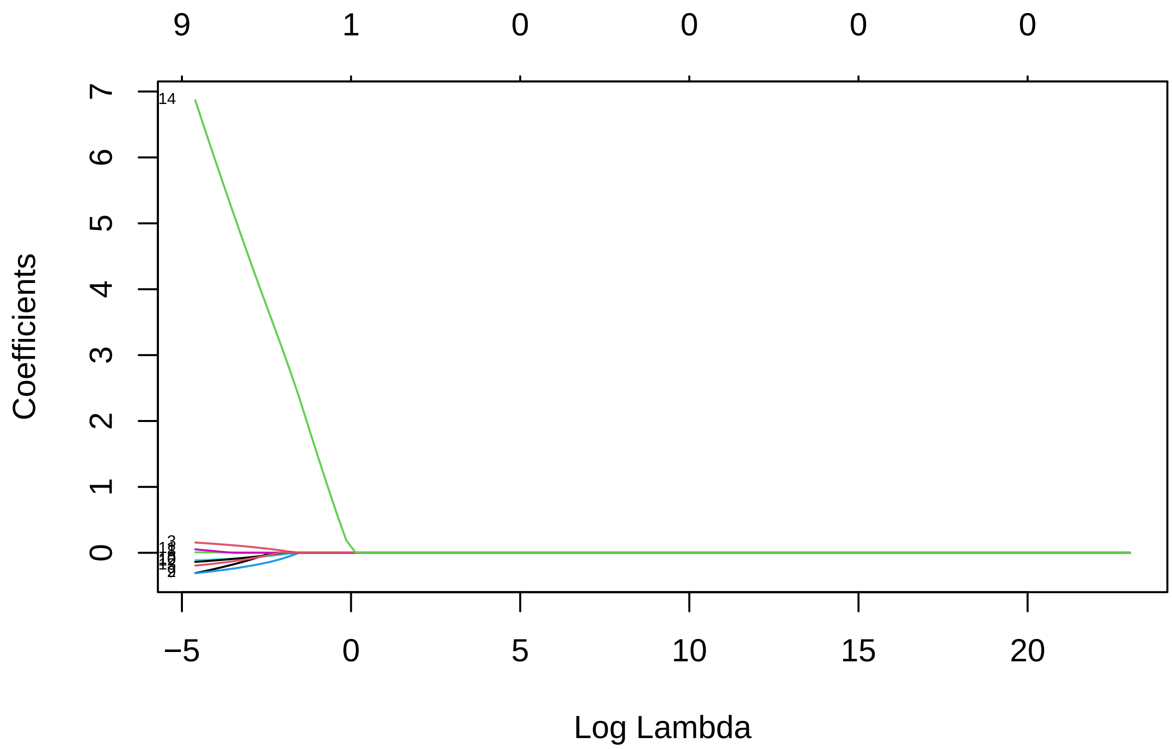
```
# test error
lasso.pred <- predict(best.lasso.mod, newx = X.test, s = bestlam)
lasso.mse <- mean((lasso.pred - y.test)^2)
lasso.mse
```

```
## [1] 0.2077333
```

```
# non-zero coefficients
lasso.coef <- predict(best.lasso.mod, type = "coefficients", s = bestlam)
lasso.coef <- lasso.coef[which(lasso.coef != 0)]
lasso.coef
```

```
## [1] -0.2468234  0.6873644
```

```
# ---------------------------------------------------------------------------- #
#                                 Elastic Net                                  #
# ---------------------------------------------------------------------------- #
# elastic net model
en.mod <- glmnet(X.train, as.factor(y.train), alpha = 0.5, lambda = grid,
                 family = "binomial")
plot(en.mod, xvar = "lambda", label = T)
```




```r
# cross-validation for lambda (with a fixed alpha)
cv.out <- cv.glmnet(X.train, y.train, alpha = 0.5)
bestlam <- cv.out$lambda.min

# coefficients of the best model
best.en.mod <- glmnet(X.train, as.factor(y.train), alpha = 0.5, lambda = bestlam,
                      family = "binomial")
coef(best.en.mod)
```

```
## 15 x 1 sparse Matrix of class "dgCMatrix"
##                       s0
## (Intercept) -3.323601000
## ï..age        .
## sex         -0.262775524
## cp           0.141658244
## trestbps      .
## chol          .
## fbs           .
## restecg       .
## thalach      0.006374926
## exang       -0.286353082
## oldpeak     -0.104291670
## slope        0.033116852
## ca          -0.122549553
## thal        -0.172448531
## target       6.221297686
```

```r
# test error
en.pred <- predict(best.en.mod, s = bestlam, newx = X.test)
en.mse <- mean((en.pred - y.test)^2)
en.mse
```

```
## [1] 8.411157
```

```r
# non-zero coefficients
en.coef <- predict(best.en.mod, type = "coefficients", s = bestlam)
en.coef <- en.coef[which(en.coef != 0)]
en.coef
```

```
##  [1] -3.323601000 -0.262775524  0.141658244  0.006374926 -0.286353082
##  [6] -0.104291670  0.033116852 -0.122549553 -0.172448531  6.221297686
```