CS5525 Project

Jennifer Appiah-Kubi, Rebecca DeSipio, Ajinkya Fotedar

11/30/2021

Contents

Introduction

- In this project, we will be trying to predict the probability of having a heart attack using 14 variables available in the `hearts.csv` data-set.
- Classification techniques employed for model fit, analysis, interpretation, and visualization:
  1. Penalized Logistic Regression
     - LASSO
     - Elastic Net
  2. Decision trees
     - Random Forest
     - Bootstrapping
  3. Support Vector Machines
  4. KNN
- Libraries used:
  1. `glmnet`
  2. `tree`
  3. `randomForest`
  4. `caret`

Data-set

```
# reading data
setwd("/Users/ajinkyafotedar/CS5525/Project/CS5525-Final-Project")
heart <- read.csv("heart.csv")

# observations
dim(heart)
```

```
## [1] 303  14
```

```
# attributes
names(heart)
```

```
##  [1] "age"      "sex"      "cp"       "trestbps" "chol"     "fbs"
##  [7] "restecg"  "thalach"  "exang"    "oldpeak"  "slope"    "ca"
## [13] "thal"     "target"
```

Attribute Information

- age
- sex
- chest pain type (4 values)
- resting blood pressure
- serum cholesterol in mg/dl
- fasting blood sugar > 120 mg/dl
- resting electrocardiograph results (values 0, 1, 2)
- maximum heart rate achieved
- exercise induced angina
- old peak = ST depression induced by exercise relative to rest
- the slope of the peak exercise ST segment
- number of major vessels (0 - 3) colored by fluoroscope
- thal: 0 = normal; 1 = fixed defect; 2 = reversible defect
- target: 0 = less chance of heart attack; 1 = more chance of heart attack

Splitting Into Train and Test

```
set.seed(123)

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

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

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

dim(X.train)
```

```
## [1] 212  13
```

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

```
## [1] 91 13
```

Classification Methods

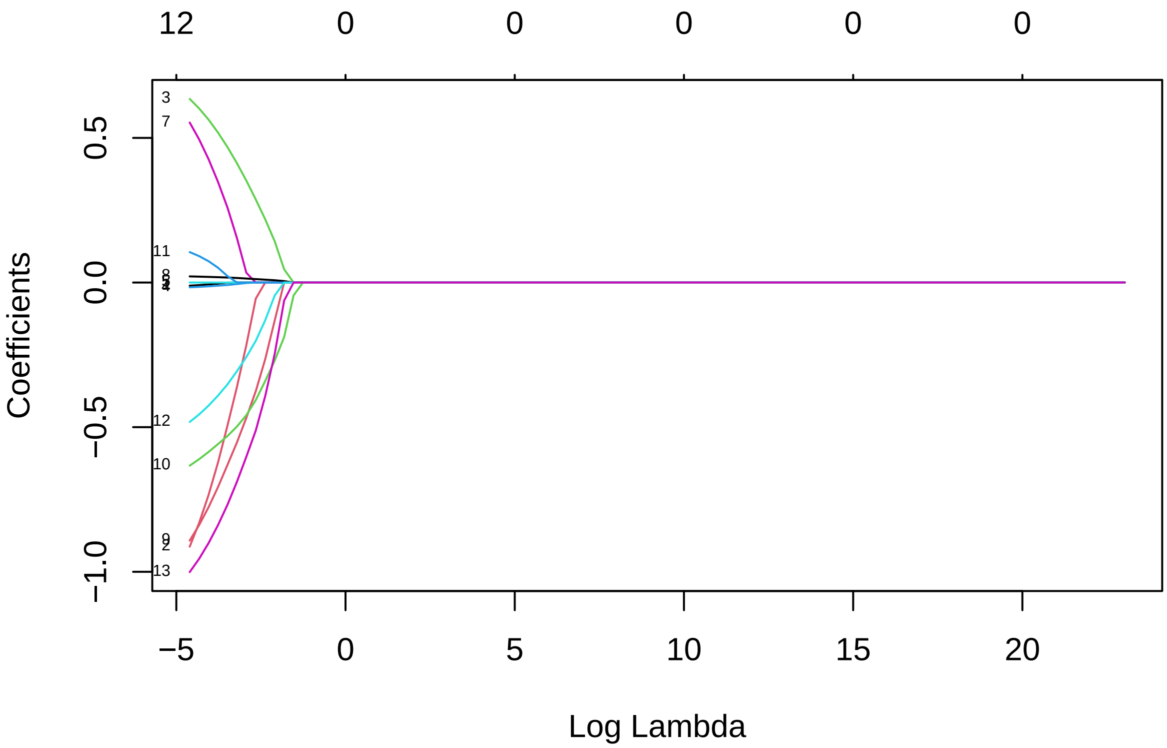
1. Penalized Logistic Regression

1.1 Lasso

```r
library(glmnet)

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

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




```r
# 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] 0.0323212
```

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

```r
coef(best.lasso.mod)
```

```
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                       s0
## (Intercept)  1.176776502
## age          .
## sex         -0.467569833
## cp           0.457343243
## trestbps    -0.007852149
## chol         .
## fbs          .
## restecg      0.239425545
## thalach      0.017066141
## exang       -0.614708498
## oldpeak     -0.524761483
## slope        0.015640883
## ca          -0.343435426
## thal        -0.752920629
```

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

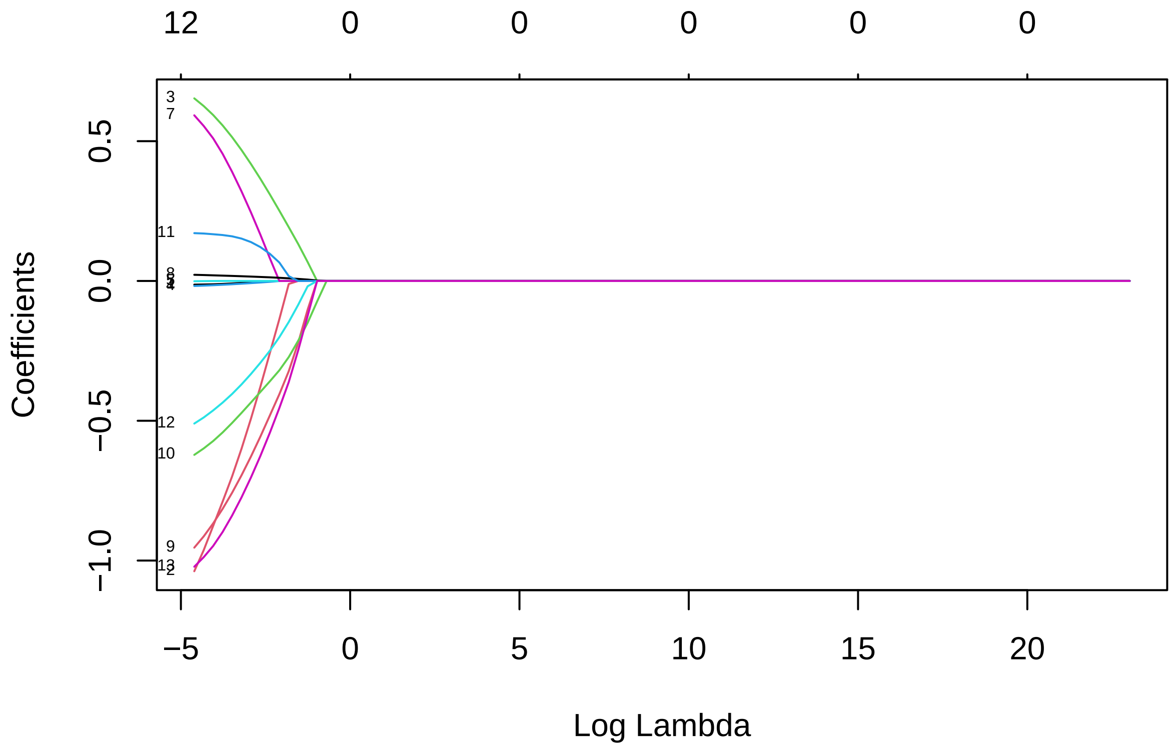
```
## [1] 2.099165
```

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

```
##  [1]  1.176776502 -0.467569833  0.457343243 -0.007852149  0.239425545
##  [6]  0.017066141 -0.614708498 -0.524761483  0.015640883 -0.343435426
## [11] -0.752920629
```

1.2 Elastic Net

```r
# 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)
```

```
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                       s0
## (Intercept)  2.49796172
## age         -0.01066195
## sex         -0.78764834
## cp           0.55563347
## trestbps    -0.01360960
## chol         .
## fbs          .
## restecg      0.45365860
## thalach      0.01909390
## exang       -0.81407451
## oldpeak     -0.54119334
## slope        0.16415637
## ca          -0.43458297
## thal        -0.89679218
```

```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] 3.515688
```

```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]  2.49796172 -0.01066195 -0.78764834  0.55563347 -0.01360960  0.45365860
##  [7]  0.01909390 -0.81407451 -0.54119334  0.16415637 -0.43458297 -0.89679218
```

2. Decision Trees

3. Support Vector Machines

```r
library(caret)

set.seed(5525)

# 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: 191, 191, 191, 191, 191, 192, ...
## Resampling results:
##
##   Accuracy  Kappa
##   0.821544  0.6317285
##
## 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 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1
## [39] 1 1 1 1 1 1 1 1 1 0 0 0 1 0 0 1 1 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 0 0 0 0 0
## [77] 0 1 0 0 0 1 1 0 0 1 0 0 0 1
## Levels: 0 1
```

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

```
## Confusion Matrix and Statistics
##
##
## svm.pred  0  1
##        0 31  4
##        1 12 43
##
##                Accuracy : 0.8222
##                  95% CI : (0.7274, 0.8948)
##     No Information Rate : 0.5222
##     P-Value [Acc > NIR] : 2.711e-09
##
##                   Kappa : 0.6409
##
##  Mcnemar's Test P-Value : 0.08012
##
##             Sensitivity : 0.7209
##             Specificity : 0.9149
##          Pos Pred Value : 0.8857
##          Neg Pred Value : 0.7818
##              Prevalence : 0.4778
##          Detection Rate : 0.3444
##    Detection Prevalence : 0.3889
##       Balanced Accuracy : 0.8179
##
##        'Positive' Class : 0
##
```

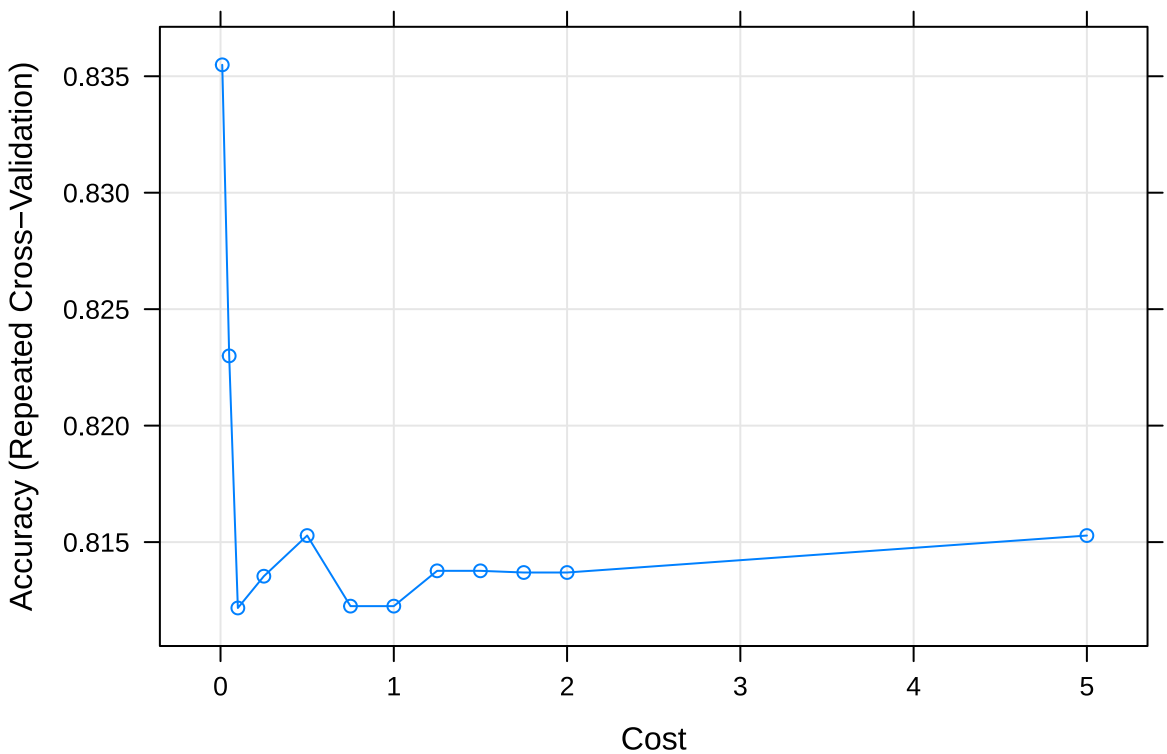
```r
# costs for further tuning with 10-fold cross-validation
grid <- expand.grid(C = c(0, 0.01, 0.05, 0.1, 0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75, 2, 5))

svm.mod.grid <- train(target ~ ., data = training, method = "svmLinear",
                      trControl = trctrl,
                      preProcess = c("center", "scale"),
                      tuneGrid = grid,
                      tuneLength = 10)
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, 192, 192, 191, 192, 191, ...
## Resampling results across tuning parameters:
##
##   C     Accuracy   Kappa
##   0.00        NaN        NaN
##   0.01  0.8354906  0.6572945
##   0.05  0.8229942  0.6321255
##   0.10  0.8121717  0.6113468
##   0.25  0.8135354  0.6158295
##   0.50  0.8152814  0.6198907
##   0.75  0.8122511  0.6134649
##   1.00  0.8122511  0.6134649
##   1.25  0.8137662  0.6167002
##   1.50  0.8137662  0.6167002
##   1.75  0.8136941  0.6167738
##   2.00  0.8136941  0.6167738
##   5.00  0.8152814  0.6199663
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was C = 0.01.
```

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

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

```
## Confusion Matrix and Statistics
##
##
## svm.pred.grid  0  1
##             0 30  2
##             1 13 45
##
##                Accuracy : 0.8333
##                  95% CI : (0.74, 0.9036)
##     No Information Rate : 0.5222
##     P-Value [Acc > NIR] : 6.187e-10
##
##                   Kappa : 0.6623
##
##  Mcnemar's Test P-Value : 0.009823
##
##             Sensitivity : 0.6977
##             Specificity : 0.9574
##          Pos Pred Value : 0.9375
##          Neg Pred Value : 0.7759
##              Prevalence : 0.4778
##          Detection Rate : 0.3333
##    Detection Prevalence : 0.3556
##       Balanced Accuracy : 0.8276
##
##        'Positive' Class : 0
##
```

4. KNN

Analysis

Conclusion