# Example_PenalizedRegression

*Lucinda Sisk*

*9/19/2019*

**Random Forest Modeling**

"Random Forest is an ensemble machine learning technique capable of performing both regression and classification tasks using multiple decision trees and a statistical technique called bagging. Bagging along with boosting are two of the most popular ensemble techniques which aim to tackle high variance and high bias. - Link here"

Helpful conceptual resorce: Towards Data Science

Random Forest in R Tutorial: Link Here

```
# Random Forest Example

library(randomForest)
```

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

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

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
require(caTools)
```

```
## Loading required package: caTools
```

```
# Download data
data <- read.csv("~/Downloads/processed.cleveland.data", header = FALSE)

# Check data dimensions
dim(data)
```

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

```
# Specify column names
names(data) <- c("age", "sex", "cp", "trestbps", "choi", "fbs",
    "restecg", "thalach", "exang", "oldpeak", "slope", "ca",
    "thai", "num")

# Check data
head(data)
```

```
##   age sex cp trestbps choi fbs restecg thalach exang oldpeak slope  ca
## 1  63   1  1      145  233   1       2     150     0     2.3     3 0.0
## 2  67   1  4      160  286   0       2     108     1     1.5     2 3.0
## 3  67   1  4      120  229   0       2     129     1     2.6     2 2.0
```

```
## 4  37   1  3       130  250   0       0      187      0      3.5      3 0.0
## 5  41   0  2       130  204   0       2      172      0      1.4      1 0.0
## 6  56   1  2       120  236   0       0      178      0      0.8      1 0.0
##   thai num
## 1  6.0   0
## 2  3.0   2
## 3  7.0   1
## 4  3.0   0
## 5  3.0   0
## 6  3.0   0
```

```
# To simplify the problem, we're only going to attempt to
# distinguish the presence of heart disease (values 1,2,3,4)
# from absence of heart disease (value 0). Therefore, we
# replace all labels greater than 1 by 1.
data$num[data$num > 1] <- 1

# Check summary data to make sure categorical variables are
# being correctly classified
summary(data)
```

```
##       age             sex               cp           trestbps
##  Min.   :29.00   Min.   :0.0000   Min.   :1.000   Min.   : 94.0
##  1st Qu.:48.00   1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:120.0
##  Median :56.00   Median :1.0000   Median :3.000   Median :130.0
##  Mean   :54.44   Mean   :0.6799   Mean   :3.158   Mean   :131.7
##  3rd Qu.:61.00   3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:140.0
##  Max.   :77.00   Max.   :1.0000   Max.   :4.000   Max.   :200.0
##       choi             fbs             restecg          thalach
##  Min.   :126.0   Min.   :0.0000   Min.   :0.0000   Min.   : 71.0
##  1st Qu.:211.0   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:133.5
##  Median :241.0   Median :0.0000   Median :1.0000   Median :153.0
##  Mean   :246.7   Mean   :0.1485   Mean   :0.9901   Mean   :149.6
##  3rd Qu.:275.0   3rd Qu.:0.0000   3rd Qu.:2.0000   3rd Qu.:166.0
##  Max.   :564.0   Max.   :1.0000   Max.   :2.0000   Max.   :202.0
##      exang           oldpeak         slope          ca        thai
##  Min.   :0.0000   Min.   :0.00   Min.   :1.000   ?  :  4   ?  :  2
##  1st Qu.:0.0000   1st Qu.:0.00   1st Qu.:1.000   0.0:176   3.0:166
##  Median :0.0000   Median :0.80   Median :2.000   1.0: 65   6.0: 18
##  Mean   :0.3267   Mean   :1.04   Mean   :1.601   2.0: 38   7.0:117
##  3rd Qu.:1.0000   3rd Qu.:1.60   3rd Qu.:2.000   3.0: 20
##  Max.   :1.0000   Max.   :6.20   Max.   :3.000
##       num
##  Min.   :0.0000
##  1st Qu.:0.0000
##  Median :0.0000
##  Mean   :0.4587
##  3rd Qu.:1.0000
##  Max.   :1.0000
```

```
# View type of each column
sapply(data, class)
```

```
##       age       sex        cp  trestbps      choi       fbs   restecg
## "numeric" "numeric" "numeric" "numeric" "numeric" "numeric" "numeric"
##   thalach     exang   oldpeak     slope        ca      thai       num
```

```
## "numeric" "numeric" "numeric" "numeric"  "factor"  "factor" "numeric"
```

```
# As we can see, sex is incorrectly treated as a number when
# in reality it can only be 1 if male and 0 if female. We can
# use the transform method to change the in built type of
# each feature.

data <- transform(data, age = as.integer(age), sex = as.factor(sex),
    cp = as.factor(cp), trestbps = as.integer(trestbps), choi = as.integer(choi),
    fbs = as.factor(fbs), restecg = as.factor(restecg), thalach = as.integer(thalach),
    exang = as.factor(exang), oldpeak = as.numeric(oldpeak),
    slope = as.factor(slope), ca = as.factor(ca), thai = as.factor(thai),
    num = as.factor(num))
sapply(data, class)
```

```
##       age       sex        cp  trestbps      choi       fbs   restecg
## "integer"  "factor"  "factor" "integer" "integer"  "factor"  "factor"
##   thalach     exang   oldpeak     slope        ca      thai       num
## "integer"  "factor" "numeric"  "factor"  "factor"  "factor"  "factor"
```

```
# Now, the categorical variables are expressed as the counts
# for each respective class. The ca and thai of certain
# samples are ? indicating missing values. R expects missing
# values to be written as NA. After replacing them, we can
# use the colSums function to view the missing value counts
# of each column.

data[data == "?"] <- NA
colSums(is.na(data))
```

```
##      age      sex       cp trestbps     choi      fbs  restecg  thalach
##        0        0        0        0        0        0        0        0
##    exang  oldpeak    slope       ca     thai      num
##        0        0        0        4        2        0
```

```
# we're just going to replace the missing values for thai
# with what is considered normal. Next, we're going to drop
# the rows where ca is missing.
data$thai[which(is.na(data$thai))] <- as.factor("3.0")
data <- data[!(data$ca %in% c(NA)), ]
colSums(is.na(data))
```

```
##      age      sex       cp trestbps     choi      fbs  restecg  thalach
##        0        0        0        0        0        0        0        0
##    exang  oldpeak    slope       ca     thai      num
##        0        0        0        0        0        0
```

```
# If we run summary again, we'll see that it still views ? as
# a potential class.
summary(data)
```

```
##       age        sex     cp          trestbps          choi       fbs
##  Min.   :29.00   0: 97   1: 23   Min.   : 94.0   Min.   :126.0   0:255
##  1st Qu.:48.00   1:202   2: 49   1st Qu.:120.0   1st Qu.:211.0   1: 44
##  Median :56.00           3: 84   Median :130.0   Median :242.0
##  Mean   :54.53           4:143   Mean   :131.7   Mean   :247.1
##  3rd Qu.:61.00                   3rd Qu.:140.0   3rd Qu.:275.5
```

```
##  Max.   :77.00                  Max.   :200.0   Max.   :564.0
##  restecg    thalach      exang      oldpeak      slope     ca
##  0:148   Min.   : 71.0   0:201   Min.   :0.000   1:140   ?  :  0
##  1:  4   1st Qu.:133.0   1: 98   1st Qu.:0.000   2:138   0.0:176
##  2:147   Median :153.0           Median :0.800   3: 21   1.0: 65
##          Mean   :149.5           Mean   :1.052           2.0: 38
##          3rd Qu.:165.5           3rd Qu.:1.600           3.0: 20
##          Max.   :202.0           Max.   :6.200
##   thai      num
##  ?  :  0   0:161
##  3.0:166   1:138
##  6.0: 18
##  7.0:115
##
##
```

```
# To get around this issue, we cast the columns to factors.
data$ca <- factor(data$ca)
data$thai <- factor(data$thai)
summary(data)
```

```
##       age        sex     cp         trestbps          choi       fbs
##  Min.   :29.00   0: 97   1: 23   Min.   : 94.0   Min.   :126.0   0:255
##  1st Qu.:48.00   1:202   2: 49   1st Qu.:120.0   1st Qu.:211.0   1: 44
##  Median :56.00           3: 84   Median :130.0   Median :242.0
##  Mean   :54.53           4:143   Mean   :131.7   Mean   :247.1
##  3rd Qu.:61.00                   3rd Qu.:140.0   3rd Qu.:275.5
##  Max.   :77.00                   Max.   :200.0   Max.   :564.0
##  restecg    thalach      exang      oldpeak      slope     ca
##  0:148   Min.   : 71.0   0:201   Min.   :0.000   1:140   0.0:176
##  1:  4   1st Qu.:133.0   1: 98   1st Qu.:0.000   2:138   1.0: 65
##  2:147   Median :153.0           Median :0.800   3: 21   2.0: 38
##          Mean   :149.5           Mean   :1.052           3.0: 20
##          3rd Qu.:165.5           3rd Qu.:1.600
##          Max.   :202.0           Max.   :6.200
##   thai      num
##  3.0:166   0:161
##  6.0: 18   1:138
##  7.0:115
##
##
##
```

```
# We're going to set a portion of our data aside for testing.
sample = sample.split(data$num, SplitRatio = 0.75)
train = subset(data, sample == TRUE)
test = subset(data, sample == FALSE)
dim(train)
```

```
## [1] 225  14
```

```
dim(test)
```

```
## [1] 74 14
```

```
# Next, we initialize an instance of the randomForest class.
# Unlike scikit-learn, we don't need to explicily call the
```

```r
# fit method to train our model.
rf <- randomForest(num ~ ., data = train)

# By default, the number of decision trees in the forest is
# 500 and the number of features used as potential candidates
# for each split is 3. The model will automatically attempt
# to classify each of the samples in the Out-Of-Bag dataset
# and display a confusion matrix with the results.

# Now, we use our model to predict whether the people in our
# testing set have heart disease.

pred = predict(rf, newdata = test[-14])

# Since this is a classification problem, we use a confusion
# matrix to evaluate the performance of our model. Recall
# that values on the diagonal correspond to true positives
# and true negatives (correct predictions) whereas the others
# correspond to false positives and false negatives.
(cm = table(test[, 14], pred))
```

```
##    pred
##      0  1
##   0 36  4
##   1 10 24
```