Random Forest Model for Heart Disease Classification

Methun George

2024-09-24

Random Forest Machine Learning Algorithm

This markdown outlines the implementation of a Random Forest algorithm inspired by Josh Starmer's machine learning example. The dataset used is the Heart Disease dataset from the UCI Machine Learning Repository.

```
#Loading the libraries

library(ggplot2)
library(cowplot)
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.4.1
```

```
## randomForest 4.7-1.2
```

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

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

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

We will be using the data from UCI repository

```
url <- "http://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/processed.cleveland.data
#read the data from the url
data <- read.csv(url, header=FALSE)

head(data)
```

```
##   V1 V2 V3  V4  V5 V6 V7  V8 V9 V10 V11 V12 V13 V14
## 1 63  1  1 145 233  1  2 150  0 2.3   3 0.0 6.0   0
## 2 67  1  4 160 286  0  2 108  1 1.5   2 3.0 3.0   2
## 3 67  1  4 120 229  0  2 129  1 2.6   2 2.0 7.0   1
## 4 37  1  3 130 250  0  0 187  0 3.5   3 0.0 3.0   0
## 5 41  0  2 130 204  0  2 172  0 1.4   1 0.0 3.0   0
## 6 56  1  2 120 236  0  0 178  0 0.8   1 0.0 3.0   0
```

The columns of the data are not named, so we checked the UCI website and assigned the names to it in the following line of code

```
colnames(data) <- c(
  "age",
  "sex",# 0 = female, 1 = male
  "cp", # chest pain
```

```
          # 1 = typical angina,
          # 2 = atypical angina,
          # 3 = non-anginal pain,
          # 4 = asymptomatic
  "trestbps", # resting blood pressure (in mm Hg)
  "chol", # serum cholestoral in mg/dl
  "fbs",  # fasting blood sugar if less than 120 mg/dl, 1 = TRUE, 0 = FALSE
  "restecg", # resting electrocardiographic results
          # 1 = normal
          # 2 = having ST-T wave abnormality
          # 3 = showing probable or definite left ventricular hypertrophy
  "thalach", # maximum heart rate achieved
  "exang",   # exercise induced angina, 1 = yes, 0 = no
  "oldpeak", # ST depression induced by exercise relative to rest
  "slope", # the slope of the peak exercise ST segment
          # 1 = upsloping
          # 2 = flat
          # 3 = downsloping
  "ca", # number of major vessels (0-3) colored by fluoroscopy
  "thal", # this is short of thalium heart scan
          # 3 = normal (no cold spots)
          # 6 = fixed defect (cold spots during rest and exercise)
          # 7 = reversible defect (when cold spots only appear during exercise)
  "hd" # (the predicted attribute) - diagnosis of heart disease
          # 0 if less than or equal to 50% diameter narrowing
          # 1 if greater than 50% diameter narrowing
  )

head(data)
```

```
##   age sex cp trestbps chol fbs restecg thalach exang oldpeak slope  ca thal hd
## 1  63   1  1      145  233   1       2     150     0     2.3     3 0.0  6.0  0
## 2  67   1  4      160  286   0       2     108     1     1.5     2 3.0  3.0  2
## 3  67   1  4      120  229   0       2     129     1     2.6     2 2.0  7.0  1
## 4  37   1  3      130  250   0       0     187     0     3.5     3 0.0  3.0  0
## 5  41   0  2      130  204   0       2     172     0     1.4     1 0.0  3.0  0
## 6  56   1  2      120  236   0       0     178     0     0.8     1 0.0  3.0  0
```

Now the column names are changed and need to check the structure of the data

```
str(data)
```

```
## 'data.frame':    303 obs. of  14 variables:
##  $ age     : num  63 67 67 37 41 56 62 57 63 53 ...
##  $ sex     : num  1 1 1 1 0 1 0 0 1 1 ...
##  $ cp      : num  1 4 4 3 2 2 4 4 4 4 ...
##  $ trestbps: num  145 160 120 130 130 120 140 120 130 140 ...
##  $ chol    : num  233 286 229 250 204 236 268 354 254 203 ...
##  $ fbs     : num  1 0 0 0 0 0 0 0 0 1 ...
##  $ restecg : num  2 2 2 0 2 0 2 0 2 2 ...
##  $ thalach : num  150 108 129 187 172 178 160 163 147 155 ...
##  $ exang   : num  0 1 1 0 0 0 0 1 0 1 ...
##  $ oldpeak : num  2.3 1.5 2.6 3.5 1.4 0.8 3.6 0.6 1.4 3.1 ...
##  $ slope   : num  3 2 2 3 1 1 3 1 2 3 ...
##  $ ca      : chr  "0.0" "3.0" "2.0" "0.0" ...
## $ thal    : chr  "6.0" "3.0" "7.0" "3.0" ...
## $ hd      : int  0 2 1 0 0 0 3 0 2 1 ...
```

Now the following code is used to structure the data into the right format (changing the num to factor etc)

```
#replace "?" with "NA"
data[data == "?"] <- NA
#substitute 0 with F and 1 with M in the column for SEX
data[data$sex == 0,]$sex <- "F"
data[data$sex == 1,]$sex <- "M"
#convert some of the columns into factors
data$sex <- as.factor(data$sex)
data$cp  <- as.factor(data$cp)
data$fbs <- as.factor(data$fbs)
data$restecg <- as.factor(data$restecg)
data$exang <- as.factor(data$exang)
data$slope <- as.factor(data$slope)

data$ca <- as.integer(data$ca)
data$ca <- as.factor(data$ca)

data$thal <- as.integer(data$thal)
data$thal <- as.factor(data$thal)

## This next line replaces 0 and 1 with "Healthy" and "Unhealthy"
data$hd <- ifelse(test=data$hd == 0, yes="Healthy", no="Unhealthy")
data$hd <- as.factor(data$hd)

str(data)
```

```
## 'data.frame':    303 obs. of  14 variables:
## $ age     : num  63 67 67 37 41 56 62 57 63 53 ...
## $ sex     : Factor w/ 2 levels "F","M": 2 2 2 2 1 2 1 1 2 2 ...
## $ cp      : Factor w/ 4 levels "1","2","3","4": 1 4 4 3 2 2 4 4 4 4 ...
## $ trestbps: num  145 160 120 130 130 120 140 120 130 140 ...
## $ chol    : num  233 286 229 250 204 236 268 354 254 203 ...
## $ fbs     : Factor w/ 2 levels "0","1": 2 1 1 1 1 1 1 1 1 2 ...
## $ restecg : Factor w/ 3 levels "0","1","2": 3 3 3 1 3 1 3 1 3 3 ...
## $ thalach : num  150 108 129 187 172 178 160 163 147 155 ...
## $ exang   : Factor w/ 2 levels "0","1": 1 2 2 1 1 1 1 2 1 2 ...
## $ oldpeak : num  2.3 1.5 2.6 3.5 1.4 0.8 3.6 0.6 1.4 3.1 ...
## $ slope   : Factor w/ 3 levels "1","2","3": 3 2 2 3 1 1 3 1 2 3 ...
## $ ca      : Factor w/ 4 levels "0","1","2","3": 1 4 3 1 1 1 3 1 2 1 ...
## $ thal    : Factor w/ 3 levels "3","6","7": 2 1 3 1 1 1 1 1 3 3 ...
## $ hd      : Factor w/ 2 levels "Healthy","Unhealthy": 1 2 2 1 1 1 2 1 2 2 ...
```

Now the seed is set to 42 for the reproducibility and impute the values for NA with rfImpute function. The prediction will be on the basis of HD (Heart Disease) in this case.

```
set.seed(42)

data.imputed <- rfImpute(hd ~ ., data = data, iter=6 )
```

```
## ntree      OOB      1      2
##   300:  17.49% 12.80% 23.02%
## ntree      OOB      1      2
##   300:  16.83% 14.02% 20.14%
## ntree      OOB      1      2
##   300:  17.82% 13.41% 23.02%
## ntree      OOB      1      2
##   300:  17.49% 14.02% 21.58%
## ntree      OOB      1      2
##   300:  17.16% 12.80% 22.30%
## ntree      OOB      1      2
##   300:  18.15% 14.63% 22.30%
```

Here the OOB stands for out-of-bag, this is the error rate. The OOB values gets smaller with improved estimations. create the model with data.imputed as the data

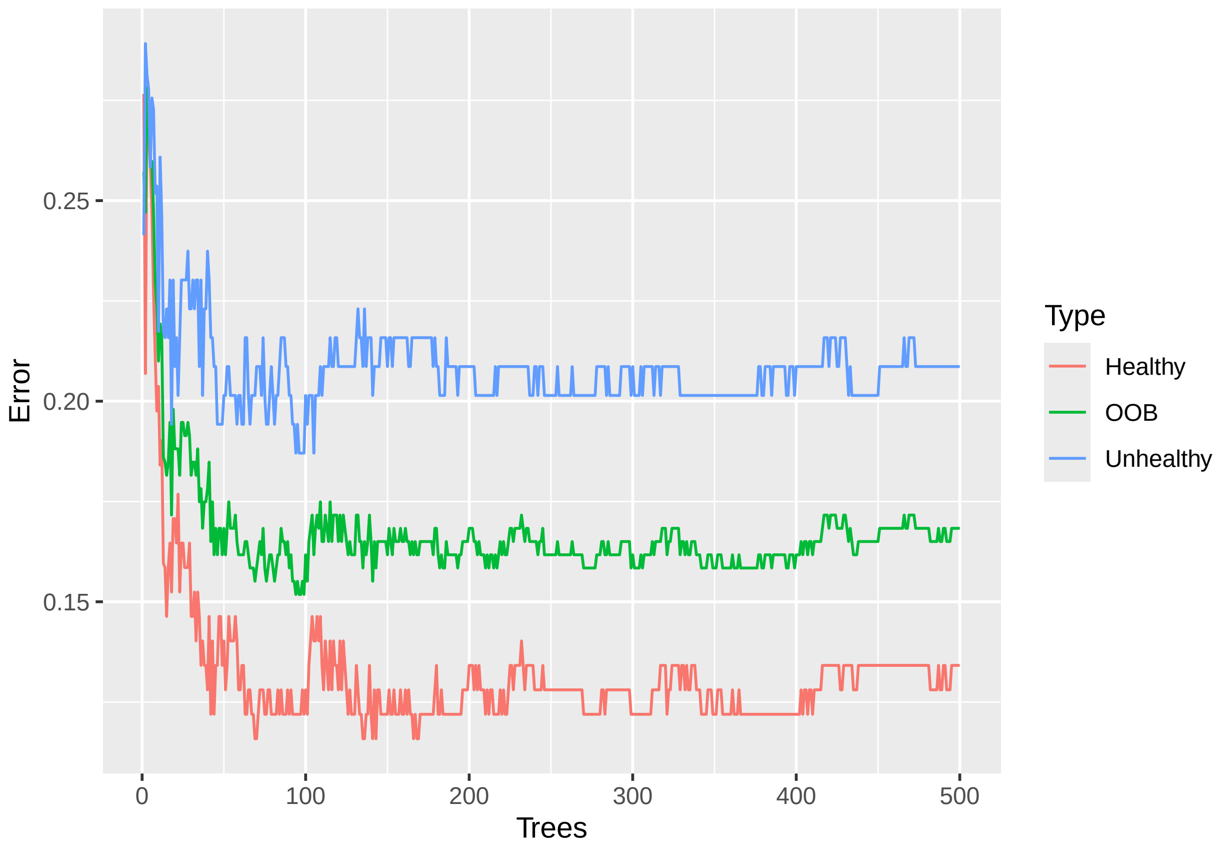
```
model <- randomForest(hd ~ ., data = data.imputed, proximity=TRUE)
model
```

```
##
## Call:
##  randomForest(formula = hd ~ ., data = data.imputed, proximity = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 16.83%
## Confusion matrix:
##           Healthy Unhealthy class.error
## Healthy       142        22   0.1341463
## Unhealthy      29       110   0.2086331
```

Above is the summary of the randomForest model created. Here, 142 healthy patinets correctly predicted to be Healthy but 29 unhealthy patients precdited to be Healthy as well. same with 22 and 110.
To see if 500 trees enough for optimal classification, we can plot the error rates using ggplot.

```
oob.error.data <- data.frame(
  Trees=rep(1:nrow(model$err.rate), times=3),
  Type=rep(c("OOB", "Healthy", "Unhealthy"), each=nrow(model$err.rate)),
  Error=c(model$err.rate[,"OOB"],
    model$err.rate[,"Healthy"],
    model$err.rate[,"Unhealthy"]))

ggplot(data=oob.error.data, aes(x=Trees, y=Error)) +
  geom_line(aes(color=Type))
```



The above graph represent the error rate respect to the no of trees.To check the influence of the number of trees in error rate, the no of trees were doubled in the following code

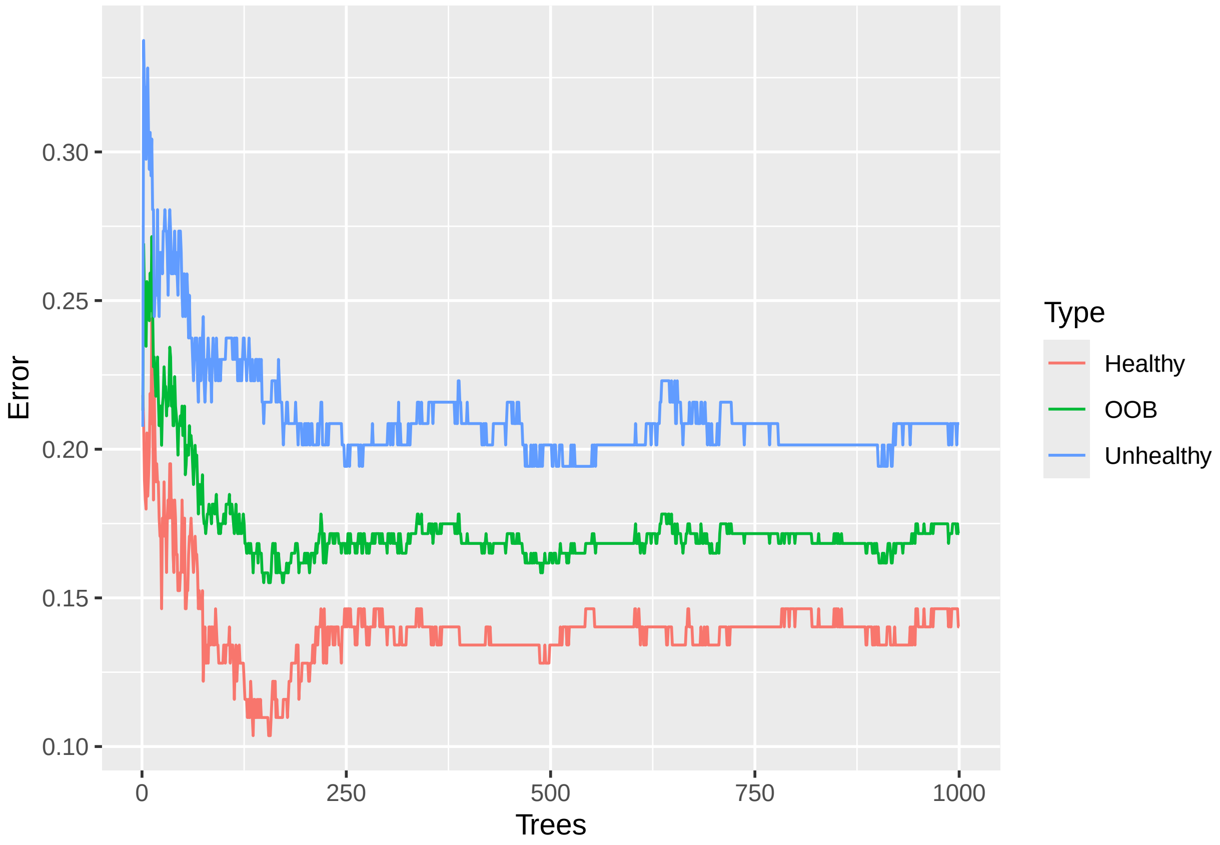
```
model_2 <- randomForest(hd ~ ., data = data.imputed, ntree=1000, proximity=TRUE)
model_2
```

```
##
## Call:
##  randomForest(formula = hd ~ ., data = data.imputed, ntree = 1000,      proximity = TRUE)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 17.16%
## Confusion matrix:
##           Healthy Unhealthy class.error
## Healthy       141        23   0.1402439
## Unhealthy      29       110   0.2086331
```

This model is compared to the old model with 500 trees and found that former was better. the graph is also created with ggplot to see the difference.

```
oob.error.data_2 <- data.frame(
  Trees=rep(1:nrow(model_2$err.rate), times=3),
  Type=rep(c("OOB", "Healthy", "Unhealthy"), each=nrow(model_2$err.rate)),
  Error=c(model_2$err.rate[,"OOB"],
    model_2$err.rate[,"Healthy"],
    model_2$err.rate[,"Unhealthy"]))
```

```r
ggplot(data=oob.error.data_2, aes(x=Trees, y=Error)) +
  geom_line(aes(color=Type))
```



Here the error rate stabilize after 500 trees.

Now we consider the optimal number of nodes in the tree (it was 3 when we look at the summary of model)

```r
#create an empty vector that holds 10 values
oob.values <- vector(length=10)
# If we want to compare this random forest to others with different values for mtry (to control how man
for (i in 1:10) {
  temp.model <- randomForest(hd ~ ., data=data.imputed, mtry=i, ntree=1000)
  oob.values[i] <- temp.model$err.rate[nrow(temp.model$err.rate),1]
}
oob.values
```

```
##  [1] 0.1749175 0.1782178 0.1716172 0.1782178 0.1683168 0.1881188 0.1815182
##  [8] 0.1980198 0.1782178 0.1848185
```

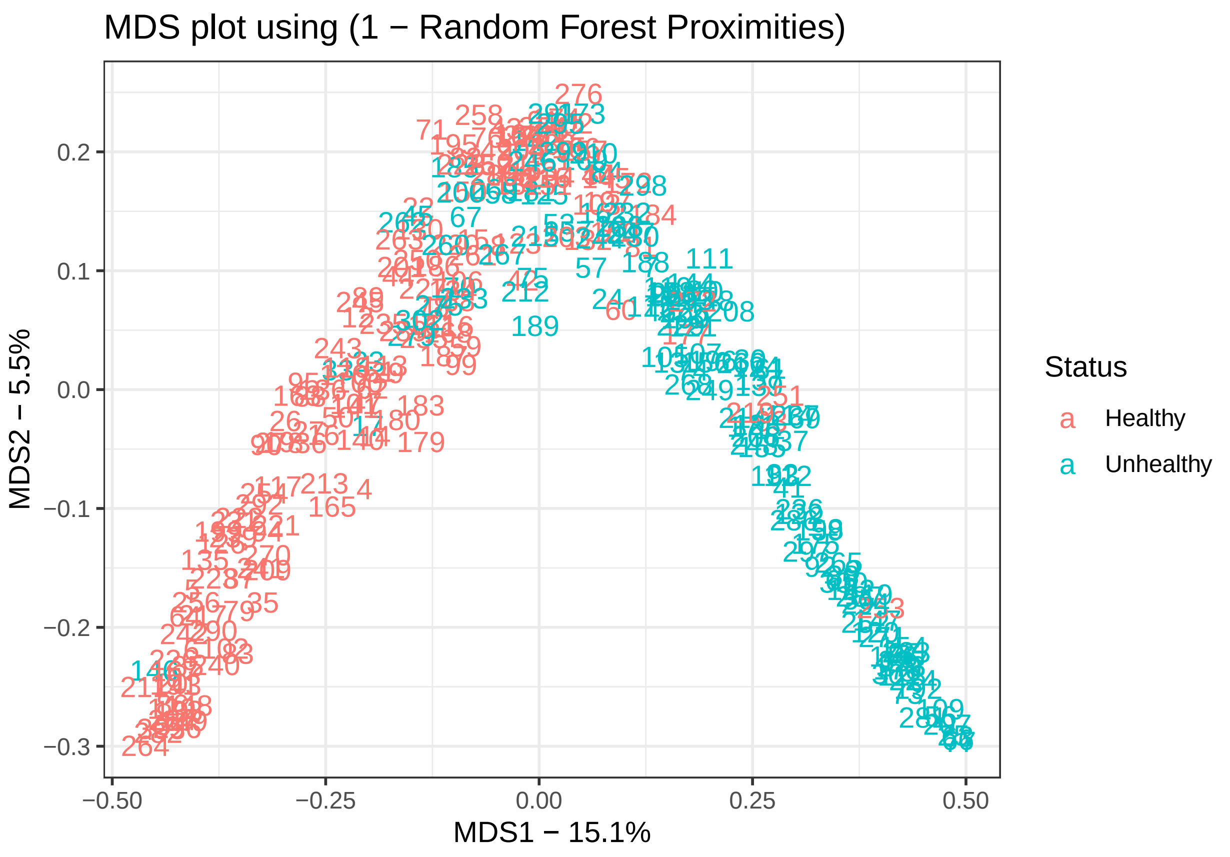
here we see the node number set to 5 has the lowest error rate. ie: 0.1683168, choose node no 5 for a better model creation.

Now we create an MDS plot using the random forest model created, this will help to understand how the samples are related to each other.

```r
#create a distance matrix by converting proximity matrix.
distance.matrix <- as.dist(1-model$proximity)
#run cmdscale on distance matrix
mds.stuff <- cmdscale(distance.matrix, eig=TRUE, x.ret=TRUE)
#calculate % of variation in distance matrix
```

```
mds.var.per <- round(mds.stuff$eig/sum(mds.stuff$eig)*100, 1)
#format the data for ggplot
mds.values <- mds.stuff$points
mds.data <- data.frame(Sample=rownames(mds.values),
  X=mds.values[,1],
  Y=mds.values[,2],
  Status=data.imputed$hd)

ggplot(data=mds.data, aes(x=X, y=Y, label=Sample)) +
  geom_text(aes(color=Status)) +
  theme_bw() +
  xlab(paste("MDS1 - ", mds.var.per[1], "%", sep="")) +
  ylab(paste("MDS2 - ", mds.var.per[2], "%", sep="")) +
  ggtitle("MDS plot using (1 - Random Forest Proximities)")
```



The healthy samples are on the left side and unhealthy samples are on the right side. The patient 251 (on the right side along with unhealthy patinets) was misdiagnosed. the X axis accounts for 15.1 % of the variation in the distance matrix and Y axis only accounts for 5.5% of the variation. This indiacate that the big difference is on the X axis.