Lab8.R

rstudio-user

2021-03-25

```
#RANDOM FOREST USING R

#installing packages
install.packages("randomForest")
```

```
## Installing package into '/home/rstudio-user/R/x86_64-pc-linux-gnu-library/4.0'
## (as 'lib' is unspecified)
```

```
library(randomForest)
```

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

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

```
install.packages("caTools")
```

```
## Installing package into '/home/rstudio-user/R/x86_64-pc-linux-gnu-library/4.0'
## (as 'lib' is unspecified)
```

```
library(caTools)

#reading dataset
data <- read.csv("processed.cleveland.data", 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
```

```
#EDA

dim(data)
```

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

```
#We can see that there are 303 rows and 14 columns.

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

#The num column contains diagnosis of heart disease. 0 means no presence of
#disease and other integer means presence. So we replace values greater than 1
```

```r
#with 1.
data$num[data$num > 1] <- 1

#Data summary
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
##  Min.   :0.0000   Min.   :0.00   Min.   :1.000   Length:303
##  1st Qu.:0.0000   1st Qu.:0.00   1st Qu.:1.000   Class :character
##  Median :0.0000   Median :0.80   Median :2.000   Mode  :character
##  Mean   :0.3267   Mean   :1.04   Mean   :1.601
##  3rd Qu.:1.0000   3rd Qu.:1.60   3rd Qu.:2.000
##  Max.   :1.0000   Max.   :6.20   Max.   :3.000
##      thai                num
##  Length:303         Min.   :0.0000
##  Class :character   1st Qu.:0.0000
##  Mode  :character   Median :0.0000
##                     Mean   :0.4587
##                     3rd Qu.:1.0000
##                     Max.   :1.0000
```

```r
#Since we are getting lost of values as 0, which means there is a problem with
#the data type of the columns.
sapply(data, class)
```

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

```r
#We can see that columns are considered wrong, like sex is considered as
#numeric, but its actually categorical. So we rectify those.
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)
)
#Now see the classes again
sapply(data, class)
```

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

```
#Now that we have rectified the column types, let's see the summary again
summary(data)
```

```
##       age        sex     cp         trestbps          choi       fbs
##  Min.   :29.00   0: 97   1: 23   Min.   : 94.0   Min.   :126.0   0:258
##  1st Qu.:48.00   1:206   2: 50   1st Qu.:120.0   1st Qu.:211.0   1: 45
##  Median :56.00           3: 86   Median :130.0   Median :241.0
##  Mean   :54.44           4:144   Mean   :131.7   Mean   :246.7
##  3rd Qu.:61.00                   3rd Qu.:140.0   3rd Qu.:275.0
##  Max.   :77.00                   Max.   :200.0   Max.   :564.0
##  restecg    thalach      exang      oldpeak     slope     ca       thai
##  0:151   Min.   : 71.0   0:204   Min.   :0.00   1:142   ?  :  4   ?  :  2
##  1:  4   1st Qu.:133.5   1: 99   1st Qu.:0.00   2:140   0.0:176   3.0:166
##  2:148   Median :153.0           Median :0.80   3: 21   1.0: 65   6.0: 18
##          Mean   :149.6           Mean   :1.04           2.0: 38   7.0:117
##          3rd Qu.:166.0           3rd Qu.:1.60           3.0: 20
##          Max.   :202.0           Max.   :6.20
##  num
##  0:164
##  1:139
##
##
##
##
```

```
#We can notice a strange "?" in category values for "ca" and "thai". Which
#implies missing values. So we replace them with NA first and then see number of
#missing values.
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
```

```
#The number showed after "?" under summary result shows the count of "?".
#The colSums() shows sum of total missing values.
```

```
#We will replace missing values for thai and drop rows with missing values in ca.
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
```

```
#Now we can see there are no null values.

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

```
#It still shows "?" as a value. So we cast it 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       thai
##  0:148   Min.   : 71.0   0:201   Min.   :0.000   1:140   0.0:176   3.0:166
##  1:  4   1st Qu.:133.0   1: 98   1st Qu.:0.000   2:138   1.0: 65   6.0: 18
##  2:147   Median :153.0           Median :0.800   3: 21   2.0: 38   7.0:115
##          Mean   :149.5           Mean   :1.052           3.0: 20
##          3rd Qu.:165.5           3rd Qu.:1.600
```

```
##          Max.   :202.0           Max.   :6.200
##  num
##  0:161
##  1:138
##
##
##
##
```

```
#Splitting data set for training and testing
sample = sample.split(data$num, SplitRatio=.75)
train = subset(data, sample==TRUE)
test = subset(data, sample==FALSE)
dim(train)
```

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

```
dim(test)
```

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

```
#Using randomForest
model <- randomForest(num ~ ., data=train)
#In this, the default number of trees is 500 and 3 features are the potential
#candidates for the split.
model
```

```
##
## Call:
##  randomForest(formula = num ~ ., data = train)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 18.67%
## Confusion matrix:
##     0  1 class.error
## 0 103 18   0.1487603
## 1  24 80   0.2307692
```

```
model1 <- randomForest(num ~ ., data=train, ntree=1000)
#Here, number of trees is specified as 200.
model1
```

```
##
## Call:
##  randomForest(formula = num ~ ., data = train, ntree = 1000)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 18.22%
## Confusion matrix:
##     0  1 class.error
## 0 104 17   0.1404959
## 1  24 80   0.2307692
```

```r
#Predicting whether people in testing set has the disease
predicted = predict(model, newdata=test[-14])
predicted
```

```
##   1   5   7   9  13  16  21  25  47  52  53  56  58  60  62  64  69  75  76  79
##   0   0   1   1   1   0   1   1   0   1   0   1   0   0   0   0   1   0   0   0
##  81  84  90  92  99 103 106 111 112 117 118 121 123 126 131 137 138 146 152 155
##   1   1   0   1   0   0   0   1   1   0   0   1   0   0   0   1   1   0   0   1
## 171 175 183 192 195 198 202 205 209 211 213 217 225 230 231 233 236 237 244 245
##   1   1   0   1   0   0   1   0   0   0   0   0   1   1   0   0   1   1   0   0
## 248 249 254 255 259 262 265 267 269 272 275 282 284 292
##   1   1   0   0   0   0   1   0   0   1   0   0   0   0
## Levels: 0 1
```

```r
#This being a classification problem, we use confusion matrix to evaluate
#the model.
mat = table(test[,14], predicted)
mat
```

```
##    predicted
##       0  1
##   0 35  5
##   1 10 24
```

```r
#With the result, we can see that 34 predictions for the people not having
#disease was correct and 26 for the people having disease were correct.
```