Artificial Neural Network (ANN) for predicting Heart Attack

Code

Hide

```
# Importing the dataset
dataset = read.csv('Heart.csv')
dataset = data.frame(dataset[-1])
dim(dataset)
```

```
[1] 303  14
```

Hide

```
str(dataset)
```

```
'data.frame':   303 obs. of  14 variables:
 $ Age      : int  63 67 67 37 41 56 62 57 63 53 ...
 $ Sex      : int  1 1 1 1 0 1 0 0 1 1 ...
 $ ChestPain: Factor w/ 4 levels "asymptomatic",..: 4 1 1 2 3 3 1 1 1 1 ...
 $ RestBP   : int  145 160 120 130 130 120 140 120 130 140 ...
 $ Chol     : int  233 286 229 250 204 236 268 354 254 203 ...
 $ Fbs      : int  1 0 0 0 0 0 0 0 0 1 ...
 $ RestECG  : int  2 2 2 0 2 0 2 0 2 2 ...
 $ MaxHR    : int  150 108 129 187 172 178 160 163 147 155 ...
 $ ExAng    : int  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    : int  3 2 2 3 1 1 3 1 2 3 ...
 $ Ca       : int  0 3 2 0 0 0 2 0 1 0 ...
 $ Thal     : Factor w/ 3 levels "fixed","normal",..: 1 2 3 2 2 2 2 2 3 3 ...
 $ AHD      : Factor w/ 2 levels "No","Yes": 1 2 2 1 1 1 2 1 2 2 ...
```

Hide

```
# Checking for missing data
d3=dataset
for(i in 1:ncol(d3))
   {
    print(colnames(d3[i]))
    print(sum(is.na(d3[i])))
}
```

```
[1] "Age"
[1] 0
[1] "Sex"
[1] 0
[1] "ChestPain"
[1] 0
[1] "RestBP"
[1] 0
[1] "Chol"
[1] 0
[1] "Fbs"
[1] 0
[1] "RestECG"
[1] 0
[1] "MaxHR"
[1] 0
[1] "ExAng"
[1] 0
[1] "Oldpeak"
[1] 0
[1] "Slope"
[1] 0
[1] "Ca"
[1] 4
[1] "Thal"
[1] 2
[1] "AHD"
[1] 0
```

Hide

```
# Some missing data is present
# Removing Missing Data from data set
dataset = na.omit(dataset)
dim(dataset)
```

```
[1] 297  14
```

Hide

```
# Checking for missing data again
d3=dataset
for(i in 1:ncol(d3))
   {
    print(colnames(d3[i]))
    print(sum(is.na(d3[i])))
}
```

```
[1] "Age"
[1] 0
[1] "Sex"
[1] 0
[1] "ChestPain"
[1] 0
[1] "RestBP"
[1] 0
[1] "Chol"
[1] 0
[1] "Fbs"
[1] 0
[1] "RestECG"
[1] 0
[1] "MaxHR"
[1] 0
[1] "ExAng"
[1] 0
[1] "Oldpeak"
[1] 0
[1] "Slope"
[1] 0
[1] "Ca"
[1] 0
[1] "Thal"
[1] 0
[1] "AHD"
[1] 0
```

Hide

```
# Encoding the categorical & numerical variables in dataset
dataset$Age = as.numeric(dataset$Age)
dataset$Sex = as.factor(dataset$Sex)
dataset$ChestPain = as.factor(dataset$ChestPain)
dataset$RestBP = as.numeric(dataset$RestBP)
dataset$Chol = as.numeric(dataset$Chol)
dataset$Fbs = as.factor(dataset$Fbs)
dataset$RestECG = as.factor(dataset$RestECG)
dataset$MaxHR = as.numeric(dataset$MaxHR)
dataset$ExAng = as.factor(dataset$ExAng)
dataset$Oldpeak = as.numeric(dataset$Oldpeak)
dataset$Slope = as.factor(dataset$Slope)
dataset$Ca = as.factor(dataset$Ca)
dataset$Thal = as.factor(dataset$Thal)
dataset$AHD = ifelse(dataset$AHD == "Yes",1,0)
dataset$AHD = as.factor(dataset$AHD)
str(dataset)
```

```
'data.frame':   297 obs. of  14 variables:
 $ Age      : num  63 67 67 37 41 56 62 57 63 53 ...
 $ Sex      : Factor w/ 2 levels "0","1": 2 2 2 2 1 2 1 1 2 2 ...
 $ ChestPain: Factor w/ 4 levels "asymptomatic",..: 4 1 1 2 3 3 1 1 1 1 ...
 $ RestBP   : 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 ...
 $ MaxHR    : 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 "fixed","normal",..: 1 2 3 2 2 2 2 2 3 3 ...
 $ AHD      : Factor w/ 2 levels "0","1": 1 2 2 1 1 1 2 1 2 2 ...
 - attr(*, "na.action")=Class 'omit'  Named int [1:6] 88 167 193 267 288 303
  .. ..- attr(*, "names")= chr [1:6] "88" "167" "193" "267" ...
```

Hide

```
# Splitting the dataset into the Training set and Test set
library(caTools)
set.seed(123)
split = sample.split(dataset$AHD, SplitRatio = 0.75)
training_set = subset(dataset, split == TRUE)
test_set = subset(dataset, split == FALSE)
# Feature Scaling
training_set[,c(1,4,5,8,10)] = scale(training_set[,c(1,4,5,8,10)])
test_set[,c(1,4,5,8,10)] = scale(test_set[,c(1,4,5,8,10)])
```

Hide

```
# Fitting ANN to the Training set
# Nodes in hidden layers selected by Avg (# Input + # Output Nodes) = (14+1)/2 =  7.5; 8
#install.packages('h2o')
library(h2o)
h2o.init(ip = "localhost")
```

```
 Connection successful!

R is connected to the H2O cluster:
    H2O cluster uptime:         23 minutes 24 milliseconds
    H2O cluster timezone:       America/New_York
    H2O data parsing timezone:  UTC
    H2O cluster version:        3.20.0.2
    H2O cluster version age:    1 month and 12 days
    H2O cluster name:           H2O_started_from_R_Rudrendu_xai688
    H2O cluster total nodes:    1
    H2O cluster total memory:   1.73 GB
    H2O cluster total cores:    4
    H2O cluster allowed cores:  4
    H2O cluster healthy:        TRUE
    H2O Connection ip:          localhost
    H2O Connection port:        54321
    H2O Connection proxy:       NA
    H2O Internal Security:      FALSE
    H2O API Extensions:         Algos, AutoML, Core V3, Core V4
    R Version:                  R version 3.4.4 (2018-03-15)
```

Hide

```
model = h2o.deeplearning(y = 14,
                         training_frame = as.h2o(training_set),
                         activation = 'Rectifier',
                         hidden = c(8,8),
                         epochs = 100,
                         train_samples_per_iteration = -2)
```

```
  |
  |                                                       |   0%
  |
  |=======================================================| 100%

  |
  |                                                       |   0%
  |
  |=======================================================| 100%
```

Hide

```
# Predicting the Test set results
y_pred = h2o.predict(model, newdata = as.h2o(test_set[-14]))
```

```
  |
  |                                                              |   0%
  |
  |==============================================================| 100%

  |
  |                                                              |   0%
  |
  |==============================================================| 100%
```

Hide

```
str(y_pred)
```

```
Class 'H2OFrame' <environment: 0x0000000002a96578>
 - attr(*, "op")= chr "predictions_b362_DeepLearning_model_R_1532814754538_14_on_file31e475b5a68
_sid_b064_15"
 - attr(*, "id")= chr "predictions_b362_DeepLearning_model_R_1532814754538_14_on_file31e475b5a68
_sid_b064_15"
 - attr(*, "eval")= logi FALSE
 - attr(*, "nrow")= int 74
 - attr(*, "ncol")= int 3
 - attr(*, "types")=List of 3
  ..$ : chr "enum"
  ..$ : chr "real"
  ..$ : chr "real"
 - attr(*, "data")='data.frame':    10 obs. of  3 variables:
  ..$ predict: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 2
  ..$ p0     : num  0.999 0.997 0.96 0.908 0.982 ...
  ..$ p1     : num  0.000569 0.003035 0.03998 0.092209 0.017695 ...
```

Hide

```
dim(test_set)
```

```
[1] 74 14
```

Hide

```
y_pred=y_pred[1]
y_pred = as.vector(y_pred)
# Making the Confusion Matrix
cm = table(test_set[, 14], y_pred)
print(cm)
```

```
   y_pred
     0  1
  0 34  6
  1  8 26
```

Hide

```
Model_Accuracy=(cm[1,1]+cm[2,2])/(cm[1,1]+cm[1,2]+cm[2,1]+cm[2,2])
print(" Model Accuracy is")
```

```
[1] " Model Accuracy is"
```

Hide

```
print(Model_Accuracy)
```

```
[1] 0.8108108
```

Hide

```
# Model Accuracy is 81% which seems quite good.
h2o.shutdown()
```

```
```