Artificial Neural Network (ANN) for improving Marketing Campaign ROI for National Bank at Portugal

Code

Hide

```
# Importing the dataset
d=read.table("bank-additional-full.csv",header=TRUE,sep=";")
dataset = data.frame(d)
dim(dataset)
```

```
[1] 41188    21
```

Hide

```
str(dataset)
```

```
'data.frame':   41188 obs. of  21 variables:
 $ age           : int  56 57 37 40 56 45 59 41 24 25 ...
 $ job           : Factor w/ 12 levels "admin.","blue-collar",..: 4 8 8 1 8 8 1 2 10 8 ...
 $ marital       : Factor w/ 4 levels "divorced","married",..: 2 2 2 2 2 2 2 2 3 3 ...
 $ education     : Factor w/ 8 levels "basic.4y","basic.6y",..: 1 4 4 2 4 3 6 8 6 4 ...
 $ default       : Factor w/ 3 levels "no","unknown",..: 1 2 1 1 1 2 1 2 1 1 ...
 $ housing       : Factor w/ 3 levels "no","unknown",..: 1 1 3 1 1 1 1 1 3 3 ...
 $ loan          : Factor w/ 3 levels "no","unknown",..: 1 1 1 1 3 1 1 1 1 1 ...
 $ contact       : Factor w/ 2 levels "cellular","telephone": 2 2 2 2 2 2 2 2 2 2 ...
 $ month         : Factor w/ 10 levels "apr","aug","dec",..: 7 7 7 7 7 7 7 7 7 7 ...
 $ day_of_week   : Factor w/ 5 levels "fri","mon","thu",..: 2 2 2 2 2 2 2 2 2 2 ...
 $ duration      : int  261 149 226 151 307 198 139 217 380 50 ...
 $ campaign      : int  1 1 1 1 1 1 1 1 1 1 ...
 $ pdays         : int  999 999 999 999 999 999 999 999 999 999 ...
 $ previous      : int  0 0 0 0 0 0 0 0 0 0 ...
 $ poutcome      : Factor w/ 3 levels "failure","nonexistent",..: 2 2 2 2 2 2 2 2 2 2 ...
 $ emp.var.rate  : num  1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 ...
 $ cons.price.idx: num  94 94 94 94 94 ...
 $ cons.conf.idx : num  -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 ...
 $ euribor3m     : num  4.86 4.86 4.86 4.86 4.86 ...
 $ nr.employed   : num  5191 5191 5191 5191 5191 ...
 $ y             : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
```

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] "job"
[1] 0
[1] "marital"
[1] 0
[1] "education"
[1] 0
[1] "default"
[1] 0
[1] "housing"
[1] 0
[1] "loan"
[1] 0
[1] "contact"
[1] 0
[1] "month"
[1] 0
[1] "day_of_week"
[1] 0
[1] "duration"
[1] 0
[1] "campaign"
[1] 0
[1] "pdays"
[1] 0
[1] "previous"
[1] 0
[1] "poutcome"
[1] 0
[1] "emp.var.rate"
[1] 0
[1] "cons.price.idx"
[1] 0
[1] "cons.conf.idx"
[1] 0
[1] "euribor3m"
[1] 0
[1] "nr.employed"
[1] 0
[1] "y"
[1] 0
```

Hide

```
# Removing Missing Data in the form of "unknown"
dataset[dataset == "unknown"] = NA
dataset = na.omit(dataset)
dim(dataset)
```

```
[1] 30488    21
```

Hide

```
# Encoding the categorical & numerical variables in dataset
dataset$age = as.numeric(dataset$age)
dataset$job = as.factor(dataset$job)
dataset$job = as.factor(dataset$job)
dataset$marital = as.factor(dataset$marital)
dataset$education = as.factor(dataset$education)
dataset$default = as.factor(dataset$default)
dataset$housing = as.factor(dataset$housing)
dataset$loan = as.factor(dataset$loan)
dataset$contact = as.factor(dataset$contact)
dataset$month = as.factor(dataset$month)
dataset$day_of_week = as.factor(dataset$day_of_week)
dataset$duration = as.numeric(dataset$duration)
dataset$campaign = as.numeric(dataset$campaign)
dataset$pdays = as.numeric(dataset$pdays)
dataset$previous = as.numeric(dataset$previous)
dataset$poutcome = as.factor(dataset$poutcome)
dataset$emp.var.rate = as.numeric(dataset$emp.var.rate)
dataset$cons.price.idx = as.numeric(dataset$cons.price.idx)
dataset$cons.conf.idx = as.numeric(dataset$cons.conf.idx)
dataset$euribor3m = as.numeric(dataset$euribor3m)
dataset$nr.employed = as.numeric(dataset$nr.employed)
dataset$y = ifelse(dataset$y == "yes",1,0)
dataset$y = as.factor(dataset$y)
str(dataset)
```

```
'data.frame':   30488 obs. of  21 variables:
 $ age           : num  56 37 40 56 59 24 25 25 29 57 ...
 $ job           : Factor w/ 12 levels "admin.","blue-collar",..: 4 8 1 8 1 10 8 8 2 4 ...
 $ marital       : Factor w/ 4 levels "divorced","married",..: 2 2 2 2 2 3 3 3 3 1 ...
 $ education     : Factor w/ 8 levels "basic.4y","basic.6y",..: 1 4 2 4 6 6 4 4 4 1 ...
 $ default       : Factor w/ 3 levels "no","unknown",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ housing       : Factor w/ 3 levels "no","unknown",..: 1 3 1 1 1 3 3 3 1 3 ...
 $ loan          : Factor w/ 3 levels "no","unknown",..: 1 1 1 3 1 1 1 1 3 1 ...
 $ contact       : Factor w/ 2 levels "cellular","telephone": 2 2 2 2 2 2 2 2 2 2 ...
 $ month         : Factor w/ 10 levels "apr","aug","dec",..: 7 7 7 7 7 7 7 7 7 7 ...
 $ day_of_week   : Factor w/ 5 levels "fri","mon","thu",..: 2 2 2 2 2 2 2 2 2 2 ...
 $ duration      : num  261 226 151 307 139 380 50 222 137 293 ...
 $ campaign      : num  1 1 1 1 1 1 1 1 1 1 ...
 $ pdays         : num  999 999 999 999 999 999 999 999 999 999 ...
 $ previous      : num  0 0 0 0 0 0 0 0 0 0 ...
 $ poutcome      : Factor w/ 3 levels "failure","nonexistent",..: 2 2 2 2 2 2 2 2 2 ...
 $ emp.var.rate  : num  1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 ...
 $ cons.price.idx: num  94 94 94 94 94 ...
 $ cons.conf.idx : num  -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 ...
 $ euribor3m     : num  4.86 4.86 4.86 4.86 4.86 ...
 $ nr.employed   : num  5191 5191 5191 5191 5191 ...
 $ y             : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
 - attr(*, "na.action")=Class 'omit'  Named int [1:10700] 2 6 8 11 16 18 20 22 27 28 ...
  .. ..- attr(*, "names")= chr [1:10700] "2" "6" "8" "11" ...
```

Hide

```
# Splitting the dataset into the Training set and Test set
library(caTools)
set.seed(123)
split = sample.split(dataset$y, SplitRatio = 0.75)
training_set = subset(dataset, split == TRUE)
test_set = subset(dataset, split == FALSE)
# Feature Scaling
training_set[,c(1,11:14,16:20)] = scale(training_set[,c(1,11:14,16:20)])
test_set[,c(1,11:14,16:20)] = scale(test_set[,c(1,11:14,16:20)])
```

Hide

```
# Fitting ANN to the Training set
# Nodes in hidden layers selected by Avg (# Input + # Output Nodes) = (21+1)/2 = 11
#install.packages('h2o')
library(h2o)
h2o.init(ip = "localhost")
```

```
 Connection successful!

R is connected to the H2O cluster:
    H2O cluster uptime:         1 minutes 27 seconds
    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 = 21,
                         training_frame = as.h2o(training_set),
                         activation = 'Rectifier',
                         hidden = c(11,11),
                         epochs = 100,
                         train_samples_per_iteration = -2)
```

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

  |
  |                                                                              |   0%
  |
  |==============                                                                |  17%
  |
  |=========================                                                     |  31%
  |
  |====================================                                          |  48%
  |
  |================================================                              |  66%
  |
  |============================================================                  |  83%
  |
  |================================================================================| 100%
```

Hide

```
# Predicting the Test set results
y_pred = h2o.predict(model, newdata = as.h2o(test_set[-21]))
```

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

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

Hide

```
str(y_pred)
```

```
Class 'H2OFrame' <environment: 0x000000001eb29e60>
 - attr(*, "op")= chr "predictions_97d9_DeepLearning_model_R_1532814754538_6_on_file31e41b193515
_sid_954f_7"
 - attr(*, "id")= chr "predictions_97d9_DeepLearning_model_R_1532814754538_6_on_file31e41b193515
_sid_954f_7"
 - attr(*, "eval")= logi FALSE
 - attr(*, "nrow")= int 7622
 - 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 2 1 1
  ..$ p0     : num  0.999 1 1 0.999 1 ...
  ..$ p1     : num  8.62e-04 3.72e-04 5.75e-05 1.44e-03 4.83e-05 ...
```

Hide

```
dim(test_set)
```

```
[1] 7622   21
```

Hide

```
y_pred=y_pred[1]
y_pred = as.vector(y_pred)
# Making the Confusion Matrix
cm = table(test_set[, 21], y_pred)
print(cm)
```

```
  y_pred
     0    1
0 5782  875
1  227  738
```

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] 0.8554185
```

Hide

```
h2o.shutdown()
```