NNsalesBagging

Dottie

12/6/2019

# Introduction

This is an application of a neural network designed to identify characterists of car buyers. The purose is to direct advertising

**ASSIGNMENT:**  
1. Find the ‘best’ sales target price for predicting sales -  
2. Determing values of age, gender, income level or commute distance *i.e.* miles/wk that predict best sales  
3. Determing the appropriate neural net - how many hidden layers, what variables to leave out?  
4. Does it help to use 80/20 (or any other) split for training/test sets?  
5. Recommend an advetising approach based on your findings 6. Based on the model you turned in in the initial NN assignment, use bagging to estimate expected effect

## first set up the environment

## table .. description of the data

The data set contains information about car buyers

|  |  |
| --- | --- |
| Variable | Description |
| age | age of the buyer |
| gender | sex of the buyer |
| miles | average number of miles driven per day |
| debt | current debt of the buyer |
| income | buyer montly income |
| sales | amount spent on a used car |

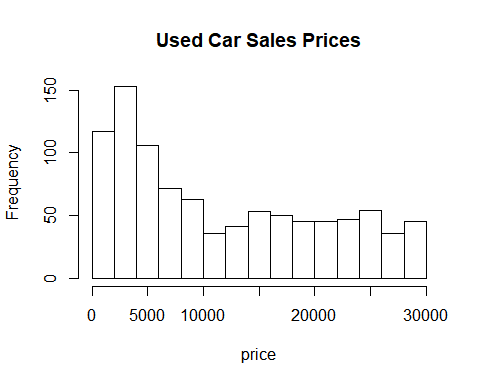
## print the first 6 entries of the cars data set

=====  
set the target price for sales  
=====

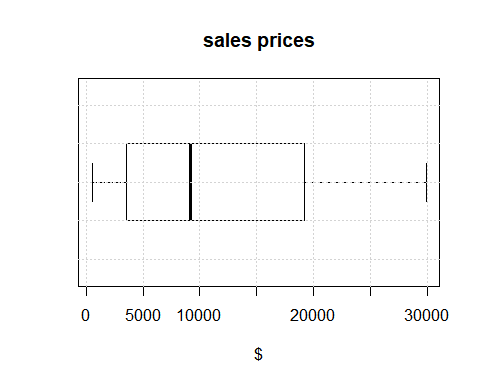
salesTargetPrice = 20000  
summary(cars$sales)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 500 3554 9130 11690 19245 29926

hist(cars$sales, xlab='price', main='Used Car Sales Prices')



boxplot(cars$sales, main='sales prices',horizontal = TRUE, xlab='$')  
grid()



# set up 'yes / no' based on sales price salesTargetPrice  
cars$salesTgt = ifelse(cars$sales>salesTargetPrice,1,0)  
head(cars)

## age gender miles debt income sales salesTgt  
## 1 28 0 23 0 4099 620 0  
## 2 26 0 27 0 2677 1792 0  
## 3 30 1 58 41576 6215 27754 1  
## 4 26 1 25 43172 7626 28256 1  
## 5 20 1 17 6979 8071 4438 0  
## 6 58 1 18 0 1262 2102 0

# Set up the Training and Test data

=====  
set the training fraction  
=====

##set up the training and test data:  
trainingFraction = 0.7  
lengthData = length(cars$age) # how many entries in the data set  
nTrain = round(trainingFraction\*lengthData,0)  
nTest = lengthData-nTrain  
# shuffle the data, take the first nTrain as the training set, the   
# remainder as the test set  
shuffle = sample(1:lengthData, replace=FALSE)   
#set up the test and train objects  
testDataSet = vector("numeric", length=nTest)  
trainDataset= vector("numeric", length=nTrain)  
trainIndex = shuffle[1:nTrain]  
trainDataSet = cars[shuffle[1:nTrain],]  
testIndex = vector("numeric")  
testIndex = shuffle[-trainIndex]  
testDataSet = cars[shuffle[-trainIndex],]  
head(trainDataSet)

## age gender miles debt income sales salesTgt  
## 359 48 0 13 963 2021 2526 0  
## 411 30 0 44 20245 10491 20638 1  
## 431 49 0 24 6446 5304 11339 0  
## 949 38 0 48 25611 11694 21989 1  
## 169 41 0 29 12317 9438 18904 0  
## 799 42 1 41 837 8509 16793 0

Set up normalized set– based on the test set

meanAge = mean(testDataSet$age)  
sdAge = sd(testDataSet$age)  
meanMiles = mean(testDataSet$miles)  
sdMiles = sd(testDataSet$miles)  
meanDebt = mean(testDataSet$debt)  
sdDebt = sd(testDataSet$debt)  
meanIncome = mean(testDataSet$income)  
sdIncome = sd(testDataSet$income)  
#normalize age, miles, debt, income, and sales  
## normalize the entire data set based on the test set..  
carsNormalized = cars  
carsNormalized$age = (cars$age-meanAge)/sdAge  
carsNormalized$miles = (cars$miles-meanMiles)/sdMiles  
carsNormalized$debt = (cars$debt - meanDebt)/sdDebt  
carsNormalized$income = (cars$income-meanIncome)/sdIncome  
carsNormalized$salesTgt = cars$salesTgt  
normalizedTrainDataSet= carsNormalized[shuffle[trainIndex],]  
normalizedTestDataSet = carsNormalized[shuffle[-trainIndex],]  
head(normalizedTrainDataSet)

## age gender miles debt income sales salesTgt  
## 390 1.09753242 0 1.23179752 -0.05218381 1.3073126 20129 1  
## 497 0.08911244 1 -0.55319444 -0.63207007 0.3764431 11413 0  
## 164 0.94239089 0 0.06767233 -0.26607267 0.1726009 6142 0  
## 534 -1.46230292 1 -0.94123617 -0.27902394 1.4254993 4827 0  
## 395 -0.68659524 0 -0.47558609 -0.73143663 -1.3721829 2832 0  
## 716 -0.22117063 0 1.54223090 -0.28552713 1.2959715 17038 0

# what factors may be in play??

linearModel = glm(sales~ age + gender + miles + debt + income, data = trainDataSet)  
summary(linearModel)

##   
## Call:  
## glm(formula = sales ~ age + gender + miles + debt + income, data = trainDataSet)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -9942.1 -2408.9 -392.5 2144.5 16090.4   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -3.951e+03 5.606e+02 -7.049 4.50e-12 \*\*\*  
## age 8.927e+01 1.220e+01 7.316 7.33e-13 \*\*\*  
## gender -2.588e+02 2.876e+02 -0.900 0.369   
## miles 1.254e+02 1.367e+01 9.177 < 2e-16 \*\*\*  
## debt 2.837e-01 1.011e-02 28.064 < 2e-16 \*\*\*  
## income 8.057e-01 5.308e-02 15.179 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for gaussian family taken to be 13919738)  
##   
## Null deviance: 5.4342e+10 on 673 degrees of freedom  
## Residual deviance: 9.2984e+09 on 668 degrees of freedom  
## AIC: 13007  
##   
## Number of Fisher Scoring iterations: 2

print(linearModel)

##   
## Call: glm(formula = sales ~ age + gender + miles + debt + income, data = trainDataSet)  
##   
## Coefficients:  
## (Intercept) age gender miles debt   
## -3951.4349 89.2748 -258.7718 125.4464 0.2837   
## income   
## 0.8057   
##   
## Degrees of Freedom: 673 Total (i.e. Null); 668 Residual  
## Null Deviance: 5.434e+10   
## Residual Deviance: 9.298e+09 AIC: 13010

leave gender out

# Traditional stat approach: Logistic Regression

logisticModel = glm(salesTgt ~ age + miles + debt + income+ gender, family=binomial, data = trainDataSet)  
summary(logisticModel)

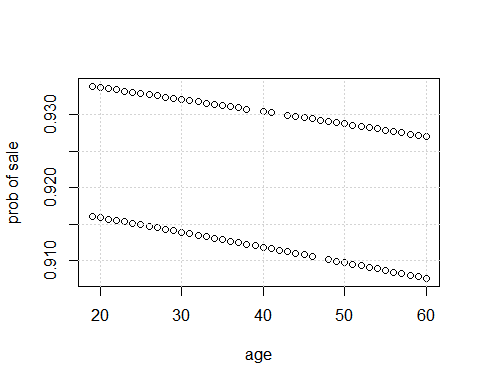
##   
## Call:  
## glm(formula = salesTgt ~ age + miles + debt + income + gender,   
## family = binomial, data = trainDataSet)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.62180 -0.21687 -0.11999 -0.05062 2.82442   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -7.881e+00 1.137e+00 -6.933 4.12e-12 \*\*\*  
## age 2.584e-03 1.837e-02 0.141 0.88815   
## miles 6.147e-02 1.914e-02 3.213 0.00132 \*\*   
## debt 1.411e-04 1.342e-05 10.514 < 2e-16 \*\*\*  
## income 2.443e-04 8.456e-05 2.888 0.00387 \*\*   
## gender 2.564e-01 3.987e-01 0.643 0.52016   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 743.41 on 673 degrees of freedom  
## Residual deviance: 192.84 on 668 degrees of freedom  
## AIC: 204.84  
##   
## Number of Fisher Scoring iterations: 7

b = as.numeric(logisticModel$coefficients)

## quick look- general probability of a sale

print probabilities of sale based on age from logistic regression

# use average values for debt, income, and miles driven/wk  
averageDebt = mean(testDataSet$debt)  
averageIncome = mean(testDataSet$income)  
averageMiles = mean(testDataSet$miles)  
# logistic regresson coefficients from previous chunk  
xx = b[1]+b[2]\*testDataSet$age + b[3]\*averageMiles +  
 b[4]\*averageDebt + b[5]\*averageIncome + b[6]\*testDataSet$gender  
pred.prob = 1./(1+exp(xx))  
plot(testDataSet$age, pred.prob, ylab='prob of sale', xlab='age')  
grid()



## Cross tabulation of logistic regression predictions

columns are ‘truth,’ rows are predictions

## [,1] [,2]  
## [1,] 0 0  
## [2,] 222 67

## [1] "columns are truth in data set, "

## [1] "rows are predicted by logistic regression "

## [1] "Overall accuracy: 23%"

# On with the NN

## Use the normalized data!

set up the nerual net

# set up the model using the training set  
salesNet = neuralnet(salesTgt~age+miles+debt+income+gender, data = normalizedTrainDataSet, hidden=c(3,2), stepmax=1.e6, rep=1, linear.output = FALSE)

plot..

plot(salesNet)

## cross tabulation for the NN

NN cross tabulation; columns are ‘truth’, rows are predictions

## [,1] [,2]  
## [1,] 172 49  
## [2,] 50 18

## [1] "columns are truth in data set, "

## [1] "rows are predicted by logistic regression "

## [1] "Overall accuracy: 0.657"

## look at a few of the fit values..

## age miles debt income  
## 1 0.6321078 -0.087544363 -0.7325389 -0.33685545  
## 2 -0.1435999 -1.096452862 -0.7757465 -1.87179035  
## 3 -0.5314537 -0.397977747 -0.7641179 -0.61381823  
## 4 -1.1520198 -0.009936017 -0.7125333 1.57472497  
## 5 -1.4623029 -0.553194439 -0.2326198 -0.48458883  
## 6 -0.1435999 0.378105714 -0.3220662 -0.18464530  
## 7 1.4853863 -0.475586093 -0.5741476 -0.61202752  
## 8 0.9423909 -1.096452862 -0.7757465 -1.32144620  
## 9 -0.8417368 0.688539098 2.0449852 0.50030038  
## 10 -1.0744491 -0.397977747 -0.7243823 -0.04467165  
## 11 0.5545370 0.610930752 -0.2828267 0.38509819  
## 12 0.5545370 0.067672329 -0.7028887 -0.41534813  
## 13 -0.3763122 -0.009936017 -0.7255397 1.37595642  
## 14 -0.3763122 -0.320369401 2.3404946 0.24094622  
## 15 -0.9968783 -1.174061208 -0.6979838 0.62952978

## [1] "Actual Predicted"

## [1] "0 0"  
## [1] "0 0"  
## [1] "0 0"  
## [1] "0 1"  
## [1] "0 0"  
## [1] "0 0"  
## [1] "0 0"  
## [1] "0 0"  
## [1] "1 0"  
## [1] "0 0"  
## [1] "0 1"  
## [1] "0 1"  
## [1] "0 0"  
## [1] "1 1"  
## [1] "0 0"

Set up an example subset of expected buyers

=====  
set up x, the input vector based on values you think are appropriate for advertising  
=====

# set test conditions  
# set up x with conditons on age, income, debt and miles you think   
# would be appropriate for advertising  
#use dply function 'filter' to pull appropriate data from test set  
x = filter(cars, age < 35 & income>3000 & debt < 15000 & miles < 30)  
# to use in the neural net, normalize the test set data  
# notice that we use mean and sd values computed from the  
# train set. The deal is, we don't know the mean and sd from   
# future observations.. until we use that data to update the model  
x$age = (x$age-meanAge)/sdAge  
x$miles = (x$miles-meanMiles)/sdMiles  
x$debt = (x$debt - meanDebt)/sdDebt  
x$income = (x$income-meanIncome)/sdIncome  
# print the values of the normalized inputs  
print(x)

## age gender miles debt income sales salesTgt  
## 1 -0.7641660 0 -0.320369401 -0.77574651 -0.648438578 620 0  
## 2 -1.3847322 1 -0.786019477 -0.39112132 0.537009884 4438 0  
## 3 -0.2987414 1 -0.320369401 -0.63273141 0.488362327 14112 0  
## 4 -0.9968783 0 -0.708411131 -0.58191834 -0.358940842 7650 0  
## 5 -1.4623029 0 -0.397977747 -0.25978993 -0.612922876 4531 0  
## 6 -0.9193075 0 -1.329277900 -0.66712117 1.039303379 5222 0  
## 7 -1.3071614 0 -0.165152709 -0.54796948 -0.614713584 4376 0  
## 8 -0.3763122 1 -0.786019477 -0.38257899 0.224829851 11530 0  
## 9 -0.5314537 1 -0.397977747 -0.77574651 -0.690818659 780 0  
## 10 -0.6865952 1 -0.708411131 -0.76659795 0.248407502 7601 0  
## 11 -1.2295906 1 -0.087544363 -0.55574024 -0.446088614 1492 0  
## 12 -1.3071614 0 -0.941236170 -0.36929705 -0.793187446 3454 0  
## 13 -0.6090245 1 -0.863627823 -0.49776264 1.531747982 15288 0  
## 14 -0.9968783 0 -0.708411131 -0.71385597 0.179763709 3852 0  
## 15 -0.8417368 1 -0.475586093 -0.64358844 -0.351479560 15573 0  
## 16 -0.8417368 0 -0.087544363 -0.41470917 -0.801245631 14074 0  
## 17 -0.5314537 0 -0.708411131 -0.69952690 -0.590240580 3453 0  
## 18 -0.6865952 0 -0.087544363 -0.47830818 -0.252095286 6376 0  
## 19 -0.9968783 1 -0.941236170 -0.77172335 0.304516341 5018 0  
## 20 -1.0744491 1 -1.018844516 -0.73534957 -0.750210463 4785 0  
## 21 -1.3071614 1 -0.165152709 -0.36378587 0.994535688 4923 0  
## 22 -1.3847322 0 -0.087544363 -0.24099680 -0.606655400 3978 0  
## 23 -0.9193075 1 -0.553194439 -0.49936088 0.441207025 9851 0  
## 24 -0.9968783 1 -0.320369401 -0.59872744 -0.875261547 3310 0  
## 25 -0.4538829 0 0.145280676 -0.34973237 -0.151815657 13699 0  
## 26 -0.9968783 1 0.067672329 -0.29070764 -0.165245965 9041 0  
## 27 -1.3071614 0 0.067672329 -0.37866606 1.155997828 1712 0  
## 28 -0.5314537 0 -0.397977747 -0.23025001 0.682654106 7165 0  
## 29 -1.3071614 1 -0.320369401 -0.25174360 -0.419526451 3014 0  
## 30 -0.6865952 0 -0.009936017 -0.66734162 -0.383413846 3224 0  
## 31 -1.2295906 0 -0.397977747 -0.63488077 -0.620384158 2950 0  
## 32 -0.6865952 0 -0.708411131 -0.39949831 -0.388487518 6748 0  
## 33 -0.9968783 0 -0.863627823 -0.48607894 -0.002888471 14039 0  
## 34 -0.8417368 0 -1.251669554 -0.71280884 1.386999114 2942 0  
## 35 -1.0744491 1 -0.320369401 -0.24215415 -0.350882657 790 0  
## 36 -1.3071614 1 -0.630802785 -0.27593768 -0.957335647 1211 0  
## 37 -0.6865952 1 -0.941236170 -0.39850630 0.673700568 16160 0  
## 38 -1.1520198 0 -0.320369401 -0.36604546 -0.455639055 3633 0  
## 39 -0.9968783 0 -0.165152709 -0.50674586 1.112423941 7960 0  
## 40 -1.0744491 0 -0.863627823 -0.30768207 -0.167335124 2626 0  
## 41 -0.6865952 1 -0.786019477 -0.77574651 -0.545771340 576 0  
## 42 -0.9968783 0 0.145280676 -0.73353088 -0.034822758 4519 0  
## 43 -1.1520198 1 -1.018844516 -0.31545283 1.120780577 3156 0  
## 44 -1.2295906 1 -0.941236170 -0.52790879 -0.540697668 1412 0  
## 45 -1.3071614 0 -0.630802785 -0.77045578 -0.472053875 4842 0  
## 46 -0.6865952 0 -1.251669554 -0.74808039 0.383904380 4250 0  
## 47 -0.5314537 0 -0.941236170 -0.75000930 -0.413258974 4620 0  
## 48 -0.6090245 0 -0.553194439 -0.72454766 -0.081381157 9937 0  
## 49 -0.7641660 1 -0.242761055 -0.48508693 0.206027421 9591 0  
## 50 -0.4538829 1 -0.009936017 -0.77574651 0.465083127 24433 1  
## 51 -1.3847322 0 -0.242761055 -0.51468196 -0.340138412 1187 0  
## 52 -0.7641660 1 -0.553194439 -0.75293023 1.082877265 5265 0  
## 53 -0.9193075 0 -1.018844516 -0.77574651 0.949171094 1097 0  
## 54 -1.0744491 1 -1.018844516 -0.39205822 1.652919200 3910 0  
## 55 -1.3071614 0 -0.242761055 -0.65229610 1.656202164 3220 0  
## 56 -0.6090245 1 -0.863627823 -0.59123223 0.960213792 6089 0  
## 57 -0.6090245 0 0.145280676 -0.59927856 0.456726491 7518 0  
## 58 -0.6865952 1 -0.941236170 -0.39856141 1.518616126 6853 0  
## 59 -1.1520198 0 -0.708411131 -0.32416050 1.672616984 3666 0  
## 60 -1.0744491 1 -1.174061208 -0.41134735 1.062284127 3004 0  
## 61 -0.7641660 0 -0.708411131 -0.77574651 -0.730811130 1347 0  
## 62 -1.3071614 0 -0.009936017 -0.37657181 -0.823331025 1041 0  
## 63 -0.8417368 0 -0.863627823 -0.77574651 1.700671404 1086 0  
## 64 -1.1520198 1 -0.553194439 -0.49974666 0.953647863 2260 0  
## 65 -0.9968783 0 -0.475586093 -0.66442069 1.290002450 16078 0  
## 66 -0.9968783 1 -0.786019477 -0.27500078 -0.521895238 7607 0  
## 67 -1.3071614 0 -1.174061208 -0.72598056 0.012630995 2440 0  
## 68 -0.3763122 1 -0.009936017 -0.72553967 1.375956417 17388 0  
## 69 -0.9193075 1 -0.475586093 -0.26662379 0.333167664 8762 0  
## 70 -0.4538829 0 -0.475586093 -0.67467148 -0.451460737 5011 0  
## 71 -1.4623029 1 -0.553194439 -0.23261981 -0.484588828 4368 0  
## 72 -1.3071614 0 -0.087544363 -0.67230168 0.886197875 1376 0  
## 73 -1.1520198 0 -0.630802785 -0.31749197 -0.434747466 3798 0  
## 74 -1.2295906 1 -1.018844516 -0.67649017 -0.751702719 3510 0  
## 75 -0.6865952 1 -0.009936017 -0.68475694 -0.845117968 5385 0  
## 76 -0.6090245 0 -0.320369401 -0.24970447 0.263628517 12608 0  
## 77 -0.8417368 0 -0.553194439 -0.67114433 -0.415049681 3068 0  
## 78 -0.9968783 0 -0.087544363 -0.75778007 0.370772524 8974 0  
## 79 -1.3847322 1 -0.009936017 -0.48128421 -0.921223043 3168 0  
## 80 -1.4623029 1 -0.863627823 -0.30900475 -0.049446870 3328 0  
## 81 -0.7641660 1 -1.174061208 -0.77574651 -0.638291235 1249 0  
## 82 -0.6090245 1 -0.087544363 -0.77574651 -0.257168957 1608 0  
## 83 -0.9968783 1 -1.018844516 -0.70746300 0.337345982 4450 0  
## 84 -0.9193075 0 -1.018844516 -0.77574651 -0.970467503 942 0  
## 85 -0.6865952 1 -0.475586093 -0.75017464 0.682355655 5029 0  
## 86 -1.2295906 1 -1.018844516 -0.46954540 -0.044970101 3437 0  
## 87 -1.0744491 1 -0.320369401 -0.58456371 -0.660376629 2756 0  
## 88 -1.3847322 0 -0.165152709 -0.53898626 -0.410274461 3635 0  
## 89 -1.3847322 0 -0.553194439 -0.28822761 -0.499511392 3002 0  
## 90 -0.9968783 0 -0.397977747 -0.30073799 -0.887199598 8506 0  
## 91 -1.0744491 1 -0.863627823 -0.76389747 -0.879738316 4765 0  
## 92 -0.6090245 0 -0.708411131 -0.30288735 1.436243574 16504 0  
## 93 -1.0744491 0 0.067672329 -0.31793286 -0.101078941 4340 0  
## 94 -0.6865952 1 -0.165152709 -0.53997827 -0.098691330 9167 0  
## 95 -0.6090245 1 -0.320369401 -0.38627148 0.166333401 5462 0  
## 96 -0.6090245 1 -0.475586093 -0.04606640 1.460418128 20103 1  
## 97 -0.9193075 0 -0.320369401 -0.76703885 -0.620085707 2707 0  
## 98 -0.9968783 1 -1.251669554 -0.67555327 -0.521596786 2838 0  
## 99 -1.2295906 0 -0.009936017 -0.57238400 -0.675299193 3240 0  
## 100 -1.1520198 0 -0.630802785 -0.69859000 -0.390278226 3039 0  
## 101 -1.0744491 1 -1.329277900 -0.30680028 -0.283134218 2620 0  
## 102 -0.5314537 0 -0.630802785 -0.67004209 -0.053625188 5327 0  
## 103 -1.3071614 1 -0.863627823 -0.74758438 0.306605500 1205 0  
## 104 -0.9193075 0 0.067672329 -0.71732801 -0.732601838 3069 0  
## 105 -0.7641660 1 -0.941236170 -0.69252771 -0.519806079 5198 0  
## 106 -1.4623029 0 -0.320369401 -0.77238469 1.274184532 2819 0  
## 107 -0.6090245 1 -0.553194439 -0.77100690 0.295264352 10401 0  
## 108 -1.4623029 0 -0.553194439 -0.22617173 1.631729159 3418 0  
## 109 -0.5314537 1 -0.397977747 -0.64160441 1.323130541 16787 0  
## 110 -0.9193075 1 0.145280676 -0.50685608 -0.012438912 8971 0  
## 111 -0.6865952 1 -0.320369401 -0.55529935 1.529957275 25182 1  
## 112 -0.4538829 0 -0.630802785 -0.73077529 0.428373620 6210 0  
## 113 -1.1520198 1 -0.708411131 -0.67450615 -0.942114632 1712 0  
## 114 -1.4623029 1 -0.941236170 -0.27902394 1.425499328 4827 0  
## 115 -0.6865952 1 -0.320369401 0.03638084 0.660568712 18848 0  
## 116 -0.8417368 0 -0.087544363 -0.69054368 -0.291789305 6146 0  
## 117 -1.0744491 0 -0.397977747 -0.72438232 -0.044671650 4456 0  
## 118 -1.1520198 1 0.067672329 -0.34344962 -0.752001170 3743 0  
## 119 -0.3763122 0 -0.397977747 -0.26590733 1.129137213 15559 0  
## 120 -1.0744491 1 -1.174061208 -0.27737059 -0.969572149 782 0  
## 121 -0.6865952 1 -0.009936017 -0.77574651 -0.206730692 1400 0  
## 122 -0.7641660 1 0.145280676 -0.54978817 -0.116598407 9508 0  
## 123 -0.9968783 0 -0.165152709 -0.29208543 0.577300806 8240 0  
## 124 -1.2295906 0 -0.087544363 -0.38445280 -0.247320065 1285 0  
## 125 -1.3071614 0 -0.087544363 -0.29092809 -0.521298335 3982 0  
## 126 -1.1520198 0 -0.087544363 -0.59536562 -0.755881037 4487 0  
## 127 -0.6865952 0 -0.009936017 -0.44435931 -0.110330930 6014 0  
## 128 -1.1520198 1 -0.165152709 -0.57761962 1.211211313 4854 0  
## 129 -1.1520198 1 -1.174061208 -0.69666109 -0.476530644 3906 0  
## 130 -1.1520198 1 -0.708411131 -0.67428570 -0.385204554 3268 0  
## 131 -1.2295906 0 -1.251669554 -0.68833921 -0.863323496 2734 0  
## 132 -0.9968783 1 -0.165152709 -0.28696004 0.228411266 7596 0  
## 133 -1.0744491 1 -0.242761055 -0.69980246 -0.722156043 4455 0  
## 134 -1.3847322 1 -0.630802785 -0.54438721 -0.751702719 4936 0  
## 135 -1.0744491 1 -1.018844516 -0.26083705 0.574614745 2184 0  
## 136 -1.3071614 0 -1.018844516 -0.52487764 -0.931071935 4731 0  
## 137 -1.2295906 1 -0.786019477 -0.70299895 -0.360134647 2010 0  
## 138 -1.3071614 1 -1.329277900 -0.36169163 -0.894362428 3487 0  
## 139 -0.8417368 1 0.067672329 -0.70470741 -0.285521828 8402 0  
## 140 -0.8417368 1 -0.397977747 -0.77574651 0.147530971 1937 0  
## 141 -0.9193075 0 -1.251669554 -0.40418282 -0.823032574 5669 0  
## 142 -0.6090245 0 -0.009936017 -0.23752476 -0.144652827 9196 0  
## 143 -0.9968783 1 -1.174061208 -0.69798377 0.629529779 3779 0  
## 144 -0.3763122 1 -0.165152709 -0.24838178 0.097391157 8396 0  
## 145 -0.6090245 1 -0.320369401 -0.73314509 -0.511747894 3267 0  
## 146 -0.6865952 0 -0.397977747 -0.77574651 -0.673806936 1358 0  
## 147 -0.4538829 0 -0.786019477 -0.73308998 0.379427611 11823 0  
## 148 -1.3847322 1 -0.009936017 -0.49208612 1.663066543 673 0  
## 149 -1.1520198 1 -1.251669554 -0.49478660 -0.611729071 3920 0  
## 150 -1.2295906 1 -1.251669554 -0.37216287 -0.461309629 4137 0  
## 151 -0.9193075 1 -0.087544363 -0.75585115 -0.855563763 5304 0  
## 152 -1.3071614 0 -1.329277900 -0.40120678 -0.599194118 4554 0  
## 153 -1.2295906 0 0.067672329 -0.75783518 -0.394158092 1176 0  
## 154 -0.7641660 0 0.145280676 -0.77574651 -0.478619803 740 0  
## 155 -0.6090245 0 -0.397977747 -0.67241190 -0.288207890 3094 0  
## 156 -1.3847322 1 -0.397977747 -0.52057892 -0.068547752 3692 0  
## 157 -0.6090245 1 -0.009936017 -0.53446709 -0.462801885 7163 0  
## 158 -1.2295906 0 -0.708411131 -0.67538794 0.067546029 4182 0  
## 159 -0.7641660 0 -0.165152709 -0.76020498 1.080191204 15823 0  
## 160 -0.5314537 1 -0.397977747 -0.76411792 -0.613818230 4257 0  
## 161 -1.3847322 1 -0.630802785 -0.42005501 0.983194540 4793 0  
## 162 -1.2295906 1 0.067672329 -0.46695515 1.493546219 4762 0  
## 163 -1.3847322 0 -0.320369401 -0.48442558 0.017406215 2802 0  
## 164 -0.2987414 1 -0.397977747 -0.35033860 -0.156292426 11428 0  
## 165 -0.9193075 1 -0.475586093 -0.54350542 1.181963088 9344 0  
## 166 -0.9193075 1 -1.174061208 -0.77574651 -0.378041723 1803 0  
## 167 -1.4623029 1 -0.630802785 -0.27389854 -0.010051302 3351 0  
## 168 -1.0744491 1 -0.009936017 -0.23862700 -0.759462452 4866 0  
## 169 -1.1520198 0 -1.329277900 -0.48200066 -0.799454923 4756 0  
## 170 -0.9968783 0 -0.165152709 -0.48839363 -0.030047537 8188 0  
## 171 -0.2987414 0 0.067672329 -0.39360135 -0.452057639 5762 0  
## 172 -1.0744491 1 -1.251669554 -0.25036581 -0.716187017 4093 0  
## 173 -1.3071614 1 -1.251669554 -0.66943586 -0.960618611 4635 0  
## 174 -0.8417368 1 -0.009936017 -0.77574651 -0.390576677 1217 0  
## 175 -1.3071614 0 -0.397977747 -0.73231842 0.304814793 1394 0  
## 176 -1.3847322 0 0.067672329 -0.52724745 0.254674979 4771 0  
## 177 -1.1520198 1 -0.009936017 -0.71253329 1.574724966 4748 0  
## 178 -0.9193075 1 0.067672329 -0.39045998 -0.141668314 7922 0  
## 179 -0.7641660 1 -0.397977747 -0.65951574 0.043371476 14063 0  
## 180 -0.3763122 1 -0.320369401 -0.45560212 -0.215982681 13722 0  
## 181 -0.8417368 0 -1.018844516 -0.73005883 -0.766625283 4272 0  
## 182 -0.6090245 1 -0.397977747 -0.27654391 0.290489132 9869 0  
## 183 -0.2987414 1 0.067672329 -0.66590871 -0.499809843 3969 0  
## 184 -0.9193075 0 -1.018844516 -0.74978886 0.387187344 4842 0  
## 185 -0.6090245 0 -0.087544363 -0.27544167 1.167935879 17821 0  
## 186 -1.0744491 0 -0.475586093 -0.63934483 -0.073919875 3702 0  
## 187 -0.2987414 1 -0.165152709 -0.39117643 0.776666258 15843 0  
## 188 -0.9968783 1 -1.329277900 -0.59911322 0.164542694 5332 0  
## 189 -1.2295906 0 -1.251669554 -0.29197521 1.061687225 1593 0  
## 190 -1.4623029 1 -0.320369401 -0.50845432 -0.836164430 4742 0

# make predicitions...   
predX = predict(salesNet, x)  
# print the results predicted by the neural net  
table(predX[,1]>0.5, x$salesTgt)

##   
## 0 1  
## FALSE 187 3

0 and 1 are truth from the train sets,  
TRUE – predicted ‘1’  
FALSE – predicted ‘0’

# Bagging set up a vector of tables,

vectorOfTables = vector(mode='list')

## run ten samples

# The automatic progress bar can be disabled by setting option dplyr.show\_progress to FALSE.  
numberBags=10  
# numberBags=25  
for (i in 1:numberBags) {  
   
 # get a bootstrap sample from the training set  
 bootSampleIndex = sample(1:nTrain, replace=TRUE)  
 bootSample = normalizedTrainDataSet[bootSampleIndex,]  
   
 # construct neural net using the bootstrap sample  
 NN = neuralnet(salesTgt~age+miles+debt+income+gender, data = bootSample, hidden=c(3,2), stepmax=1.e6, rep=1, linear.output = FALSE)  
   
 # get normalizing constants from the bootstrap sample  
 meanAge = mean(trainDataSet[bootSampleIndex,'age'])  
 sdAge = sd(trainDataSet[bootSampleIndex,'age'])  
 meanMiles=mean(trainDataSet[bootSampleIndex,'miles'])  
 sdMiles = sd(trainDataSet[bootSampleIndex,'miles'])  
 meanDebt = mean(trainDataSet[bootSampleIndex,'debt'])  
 sdDebt = sd(trainDataSet[bootSampleIndex,'debt'])  
 meanIncome = mean(trainDataSet[bootSampleIndex,'income'])  
 sdIncome = sd(trainDataSet[bootSampleIndex,'income'])  
   
 # apply the model to the test set  
 x = testDataSet  
 x$salesTgt = ifelse(x$sales>salesTargetPrice,1,0)  
 x$age = (x$age-meanAge)/sdAge  
 x$miles = (x$miles-meanMiles)/sdMiles  
 x$debt = (x$debt - meanDebt)/sdDebt  
 x$income = (x$income-meanIncome)/sdIncome  
   
   
 # save the table of predictions; we'll look at these later  
 predX = predict(NN, x)  
 vectorOfTables[[i]]=table(predX[,1]>0.5, x$salesTgt)  
}  
   
print(vectorOfTables)

## [[1]]  
##   
## 0 1  
## FALSE 210 6  
## TRUE 12 61  
##   
## [[2]]  
##   
## 0 1  
## FALSE 208 8  
## TRUE 14 59  
##   
## [[3]]  
##   
## 0 1  
## FALSE 212 9  
## TRUE 10 58  
##   
## [[4]]  
##   
## 0 1  
## FALSE 213 11  
## TRUE 9 56  
##   
## [[5]]  
##   
## 0 1  
## FALSE 213 11  
## TRUE 9 56  
##   
## [[6]]  
##   
## 0 1  
## FALSE 215 9  
## TRUE 7 58  
##   
## [[7]]  
##   
## 0 1  
## FALSE 209 7  
## TRUE 13 60  
##   
## [[8]]  
##   
## 0 1  
## FALSE 214 12  
## TRUE 8 55  
##   
## [[9]]  
##   
## 0 1  
## FALSE 213 11  
## TRUE 9 56  
##   
## [[10]]  
##   
## 0 1  
## FALSE 210 4  
## TRUE 12 63