Predicting-Automobile-Prices-using-NN-and-LReg.R

patriciamaya

2020-11-06

#Predicting Automobile Prices using Neural Networks  
  
#We want to predict an accurate manufacturer's suggested retail price (MSRP) by using data collected for a previous batch of cars.  
#This data contains 27 independent variables such as price, age, fuel type, horse power, etc, and 1 dependent variable (price of each car)  
  
library(readxl)

## Warning: package 'readxl' was built under R version 4.0.2

data <- read\_excel("~/Downloads/Car\_Data.xlsx")  
str(data)

## tibble [31 × 28] (S3: tbl\_df/tbl/data.frame)  
## $ Price : num [1:31] 21000 20000 19650 21550 22550 ...  
## $ Age : num [1:31] 26 23 26 32 33 29 31 25 25 31 ...  
## $ KM : num [1:31] 31463 43612 32191 23002 34133 ...  
## $ Fuel : chr [1:31] "Petrol" "Petrol" "Petrol" "Petrol" ...  
## $ HP : num [1:31] 195 195 195 195 195 195 195 113 113 113 ...  
## $ MC : num [1:31] 0 0 0 1 1 0 1 1 0 1 ...  
## $ Colour : chr [1:31] "silver" "red" "red" "black" ...  
## $ Auto : num [1:31] 0 0 0 0 0 0 0 0 0 0 ...  
## $ CC : num [1:31] 1800 1800 1800 1800 1800 1800 1800 1600 1600 1600 ...  
## $ Drs : num [1:31] 3 3 3 3 3 3 3 3 3 3 ...  
## $ Cyl : num [1:31] 3 3 3 3 3 3 3 3 3 3 ...  
## $ Grs : num [1:31] 6 6 6 6 6 6 5 5 5 5 ...  
## $ Wght : num [1:31] 1189 1189 1189 1189 1189 ...  
## $ G\_P : num [1:31] 10 4 4 4 4 4 4 20 4 4 ...  
## $ Mfr\_G : num [1:31] 1 1 1 1 1 1 1 0 0 1 ...  
## $ ABS : num [1:31] 1 1 1 1 1 1 1 1 1 1 ...  
## $ Abag\_1 : num [1:31] 1 1 1 1 1 1 1 1 1 1 ...  
## $ Abag\_2 : num [1:31] 1 1 1 1 1 1 1 0 1 1 ...  
## $ AC : num [1:31] 1 1 1 1 1 1 1 0 1 0 ...  
## $ Comp : num [1:31] 0 1 1 1 1 1 1 0 1 1 ...  
## $ CD : num [1:31] 1 0 0 1 1 0 1 0 1 1 ...  
## $ Clock : num [1:31] 1 1 1 1 1 1 1 1 1 1 ...  
## $ Pw : num [1:31] 1 1 1 1 1 1 1 1 1 1 ...  
## $ PStr : num [1:31] 1 1 1 1 1 1 1 1 1 1 ...  
## $ Radio : num [1:31] 0 0 0 0 0 0 0 1 0 0 ...  
## $ SpM : num [1:31] 0 1 1 1 1 1 0 0 0 1 ...  
## $ M\_Rim : num [1:31] 1 1 1 1 1 1 1 0 0 0 ...  
## $ Tow\_Bar: num [1:31] 0 0 0 0 0 0 0 1 0 0 ...

#Data Preprocessing  
data$Fuel <- as.factor(data$Fuel)  
data$MC <- as.factor(data$MC)  
data$Colour <- as.factor(data$Colour)  
data$Auto <- as.factor(data$Auto)  
data$Mfr\_G <- as.factor(data$Mfr\_G)  
data$ABS <- as.factor(data$ABS)  
data$Abag\_1 <- as.factor(data$Abag\_1)  
data$Abag\_2 <- as.factor(data$Abag\_2)  
data$AC <- as.factor(data$AC)  
data$Comp <- as.factor(data$Comp)  
data$CD <- as.factor(data$CD)  
data$Clock <- as.factor(data$Clock)  
data$Pw <- as.factor(data$Pw)  
data$PStr <- as.factor(data$PStr)  
data$Radio <- as.factor(data$Radio)  
data$SpM <- as.factor(data$SpM)  
data$M\_Rim <- as.factor(data$M\_Rim)  
data$Tow\_Bar <- as.factor(data$Tow\_Bar)  
str(data)

## tibble [31 × 28] (S3: tbl\_df/tbl/data.frame)  
## $ Price : num [1:31] 21000 20000 19650 21550 22550 ...  
## $ Age : num [1:31] 26 23 26 32 33 29 31 25 25 31 ...  
## $ KM : num [1:31] 31463 43612 32191 23002 34133 ...  
## $ Fuel : Factor w/ 1 level "Petrol": 1 1 1 1 1 1 1 1 1 1 ...  
## $ HP : num [1:31] 195 195 195 195 195 195 195 113 113 113 ...  
## $ MC : Factor w/ 2 levels "0","1": 1 1 1 2 2 1 2 2 1 2 ...  
## $ Colour : Factor w/ 6 levels "black","blue",..: 6 5 5 1 4 4 4 2 4 4 ...  
## $ Auto : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ CC : num [1:31] 1800 1800 1800 1800 1800 1800 1800 1600 1600 1600 ...  
## $ Drs : num [1:31] 3 3 3 3 3 3 3 3 3 3 ...  
## $ Cyl : num [1:31] 3 3 3 3 3 3 3 3 3 3 ...  
## $ Grs : num [1:31] 6 6 6 6 6 6 5 5 5 5 ...  
## $ Wght : num [1:31] 1189 1189 1189 1189 1189 ...  
## $ G\_P : num [1:31] 10 4 4 4 4 4 4 20 4 4 ...  
## $ Mfr\_G : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 1 1 2 ...  
## $ ABS : Factor w/ 1 level "1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ Abag\_1 : Factor w/ 1 level "1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ Abag\_2 : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 1 2 2 ...  
## $ AC : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 1 2 1 ...  
## $ Comp : Factor w/ 2 levels "0","1": 1 2 2 2 2 2 2 1 2 2 ...  
## $ CD : Factor w/ 2 levels "0","1": 2 1 1 2 2 1 2 1 2 2 ...  
## $ Clock : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...  
## $ Pw : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...  
## $ PStr : Factor w/ 1 level "1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ Radio : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 2 1 1 ...  
## $ SpM : Factor w/ 2 levels "0","1": 1 2 2 2 2 2 1 1 1 2 ...  
## $ M\_Rim : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 1 1 1 ...  
## $ Tow\_Bar: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 2 1 1 ...

#normalize numerical variables  
#x'= x - min / max -min  
num\_cols <- unlist(lapply(data, is.numeric))  
num\_cols

## Price Age KM Fuel HP MC Colour Auto CC Drs   
## TRUE TRUE TRUE FALSE TRUE FALSE FALSE FALSE TRUE TRUE   
## Cyl Grs Wght G\_P Mfr\_G ABS Abag\_1 Abag\_2 AC Comp   
## TRUE TRUE TRUE TRUE FALSE FALSE FALSE FALSE FALSE FALSE   
## CD Clock Pw PStr Radio SpM M\_Rim Tow\_Bar   
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE

dataNum <- data[,num\_cols] #only numerical variables  
#min & max of all columns  
mins<- apply(dataNum, 2, min) #1-rows, 2 -columns  
maxs<- apply(dataNum, 2, max)  
  
scaled.data<-as.data.frame(scale(dataNum, center = mins, scale= maxs - mins))  
summary(scaled.data) #we can see now min is 0 and max is 1.

## Price Age KM HP   
## Min. :0.0000 Min. :0.0000 Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.2857 1st Qu.:0.3000 1st Qu.:0.2710 1st Qu.:0.0000   
## Median :0.3367 Median :0.5000 Median :0.3880 Median :0.1368   
## Mean :0.4338 Mean :0.4935 Mean :0.4128 Mean :0.2832   
## 3rd Qu.:0.5102 3rd Qu.:0.7000 3rd Qu.:0.5605 3rd Qu.:0.1368   
## Max. :1.0000 Max. :1.0000 Max. :1.0000 Max. :1.0000   
##   
## CC Drs Cyl Grs Wght   
## Min. :0.0000 Min. : NA Min. : NA Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.0000 1st Qu.: NA 1st Qu.: NA 1st Qu.:0.0000 1st Qu.:0.2917   
## Median :0.5000 Median : NA Median : NA Median :0.0000 Median :0.4583   
## Mean :0.4355 Mean :NaN Mean :NaN Mean :0.1935 Mean :0.5081   
## 3rd Qu.:0.5000 3rd Qu.: NA 3rd Qu.: NA 3rd Qu.:0.0000 3rd Qu.:0.6667   
## Max. :1.0000 Max. : NA Max. : NA Max. :1.0000 Max. :1.0000   
## NA's :31 NA's :31   
## G\_P   
## Min. :0.00000   
## 1st Qu.:0.00000   
## Median :0.00000   
## Mean :0.05645   
## 3rd Qu.:0.00000   
## Max. :1.00000   
##

#add to scaled df the variables that were factors (categorical)  
data <- data.frame(scaled.data, data[!num\_cols] )  
str(data)

## 'data.frame': 31 obs. of 28 variables:  
## $ Price : num 0.816 0.714 0.679 0.872 0.974 ...  
## $ Age : num 0.3 0 0.3 0.9 1 0.6 0.8 0.2 0.2 0.8 ...  
## $ KM : num 0.375 0.585 0.387 0.229 0.421 ...  
## $ HP : num 1 1 1 1 1 ...  
## $ CC : num 1 1 1 1 1 1 1 0.5 0.5 0.5 ...  
## $ Drs : num NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN ...  
## $ Cyl : num NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN ...  
## $ Grs : num 1 1 1 1 1 1 0 0 0 0 ...  
## $ Wght : num 1 1 1 1 1 ...  
## $ G\_P : num 0.375 0 0 0 0 0 0 1 0 0 ...  
## $ Fuel : Factor w/ 1 level "Petrol": 1 1 1 1 1 1 1 1 1 1 ...  
## $ MC : Factor w/ 2 levels "0","1": 1 1 1 2 2 1 2 2 1 2 ...  
## $ Colour : Factor w/ 6 levels "black","blue",..: 6 5 5 1 4 4 4 2 4 4 ...  
## $ Auto : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ Mfr\_G : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 1 1 2 ...  
## $ ABS : Factor w/ 1 level "1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ Abag\_1 : Factor w/ 1 level "1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ Abag\_2 : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 1 2 2 ...  
## $ AC : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 1 2 1 ...  
## $ Comp : Factor w/ 2 levels "0","1": 1 2 2 2 2 2 2 1 2 2 ...  
## $ CD : Factor w/ 2 levels "0","1": 2 1 1 2 2 1 2 1 2 2 ...  
## $ Clock : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...  
## $ Pw : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...  
## $ PStr : Factor w/ 1 level "1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ Radio : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 2 1 1 ...  
## $ SpM : Factor w/ 2 levels "0","1": 1 2 2 2 2 2 1 1 1 2 ...  
## $ M\_Rim : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 1 1 1 ...  
## $ Tow\_Bar: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 2 1 1 ...

#we leave only categorical variables with more than 1 factor level and remove num variables that have the same value for all instances  
data<- data[,-6] #remove drs  
data<- data[,-6] #remove cyl  
data<- data[,-9] #remove fuel  
data<- data[,-13] #remove abs  
data<- data[,-13] #remove abag\_1  
data<- data[,-19] #remove PStr  
str(data)

## 'data.frame': 31 obs. of 22 variables:  
## $ Price : num 0.816 0.714 0.679 0.872 0.974 ...  
## $ Age : num 0.3 0 0.3 0.9 1 0.6 0.8 0.2 0.2 0.8 ...  
## $ KM : num 0.375 0.585 0.387 0.229 0.421 ...  
## $ HP : num 1 1 1 1 1 ...  
## $ CC : num 1 1 1 1 1 1 1 0.5 0.5 0.5 ...  
## $ Grs : num 1 1 1 1 1 1 0 0 0 0 ...  
## $ Wght : num 1 1 1 1 1 ...  
## $ G\_P : num 0.375 0 0 0 0 0 0 1 0 0 ...  
## $ MC : Factor w/ 2 levels "0","1": 1 1 1 2 2 1 2 2 1 2 ...  
## $ Colour : Factor w/ 6 levels "black","blue",..: 6 5 5 1 4 4 4 2 4 4 ...  
## $ Auto : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ Mfr\_G : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 1 1 2 ...  
## $ Abag\_2 : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 1 2 2 ...  
## $ AC : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 1 2 1 ...  
## $ Comp : Factor w/ 2 levels "0","1": 1 2 2 2 2 2 2 1 2 2 ...  
## $ CD : Factor w/ 2 levels "0","1": 2 1 1 2 2 1 2 1 2 2 ...  
## $ Clock : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...  
## $ Pw : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...  
## $ Radio : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 2 1 1 ...  
## $ SpM : Factor w/ 2 levels "0","1": 1 2 2 2 2 2 1 1 1 2 ...  
## $ M\_Rim : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 1 1 1 ...  
## $ Tow\_Bar: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 2 1 1 ...

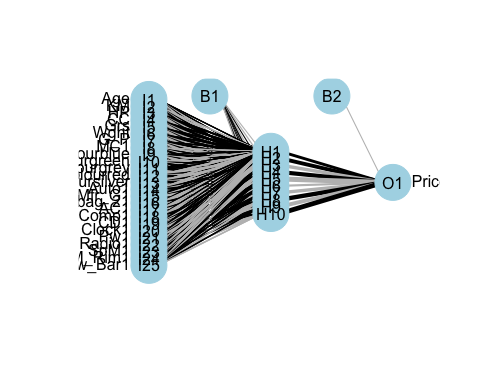
**#\*\*\*\*\*\*\*\*\*\*\*\*\* Neural Net with 10 hidden neurons**  
set.seed(154)  
indx<- sample(2, nrow(data), replace=T, prob=c(0.8,0.2))  
train <- data[indx==1, ]  
test <- data[indx==2, ]  
   
library(nnet)   
nn <- nnet(Price ~ . , data= train, linout= T , size= 10, decay=0.01)

## # weights: 271  
## initial value 2.996035   
## iter 10 value 0.485863  
## iter 20 value 0.120596  
## iter 30 value 0.066076  
## iter 40 value 0.060760  
## iter 50 value 0.059755  
## iter 60 value 0.058940  
## iter 70 value 0.058455  
## iter 80 value 0.058269  
## iter 90 value 0.058058  
## iter 100 value 0.057949  
## final value 0.057949   
## stopped after 100 iterations

library(devtools)

source\_url('https://gist.githubusercontent.com/fawda123/7471137/raw/466c1474d0a505ff044412703516c34f1a4684a5/nnet\_plot\_update.r')

plot.nnet(nn)



#TEST   
nn.preds<- predict(nn, test)  
nn.preds #predicted values on test data

## [,1]  
## 5 0.8656114  
## 7 0.7372649  
## 15 0.4151589  
## 22 0.1756761  
## 25 0.2325943  
## 26 0.2841179  
## 28 0.3165041  
## 30 0.1392859

#EVALUATE  
#MEAN SQUARED ERROR  
mse <- mean((nn.preds - test$Price)^2)   
mse

## [1] 0.01431462

**#\*\*\*\*\*\*\*\*\*\*\*\*\* 10-Fold CV with 10 hidden neurons**  
k <- 10  
nmethod <- 1  
folds <- cut(seq(1,nrow(data)),breaks=k,labels=FALSE)   
model.meansquarederror <- matrix(-1, k, nmethod, dimnames=list(paste0("Fold", 1:k), c("NNet")))  
  
for(i in 1:k)  
{   
 testindexes <- which(folds == i, arr.ind=TRUE)   
 test <- data[testindexes, ]   
 train <- data[-testindexes, ]   
   
 nnModel<- nnet(Price ~ . , data= train, linout= T, size= 10, decay=0.01)  
 predicted <- predict(nnModel, test)  
 model.meansquarederror[i] <- mean((test$Price - predicted)^2)  
}

## # weights: 271  
## initial value 13.458846   
## iter 10 value 0.574049  
## iter 20 value 0.167997  
## iter 30 value 0.092745  
## iter 40 value 0.085800  
## iter 50 value 0.084339  
## iter 60 value 0.083849  
## iter 70 value 0.083671  
## iter 80 value 0.083519  
## iter 90 value 0.083439  
## iter 100 value 0.083359  
## final value 0.083359   
## stopped after 100 iterations  
## # weights: 271  
## initial value 6.216472   
## iter 10 value 0.400509  
## iter 20 value 0.124637  
## iter 30 value 0.078163  
## iter 40 value 0.074726  
## iter 50 value 0.074036  
## iter 60 value 0.073751  
## iter 70 value 0.073616  
## iter 80 value 0.073504  
## iter 90 value 0.073409  
## iter 100 value 0.073363  
## final value 0.073363   
## stopped after 100 iterations  
## # weights: 271  
## initial value 38.442358   
## iter 10 value 0.475361  
## iter 20 value 0.140956  
## iter 30 value 0.097887  
## iter 40 value 0.089481  
## iter 50 value 0.087374  
## iter 60 value 0.086873  
## iter 70 value 0.086678  
## iter 80 value 0.086421  
## iter 90 value 0.086129  
## iter 100 value 0.085943  
## final value 0.085943   
## stopped after 100 iterations  
## # weights: 271  
## initial value 15.597217   
## iter 10 value 0.491837  
## iter 20 value 0.153532  
## iter 30 value 0.090084  
## iter 40 value 0.083377  
## iter 50 value 0.081137  
## iter 60 value 0.080187  
## iter 70 value 0.079883  
## iter 80 value 0.079657  
## iter 90 value 0.079299  
## iter 100 value 0.079056  
## final value 0.079056   
## stopped after 100 iterations  
## # weights: 271  
## initial value 11.568156   
## iter 10 value 0.417691  
## iter 20 value 0.114930  
## iter 30 value 0.086278  
## iter 40 value 0.082506  
## iter 50 value 0.081203  
## iter 60 value 0.080155  
## iter 70 value 0.079673  
## iter 80 value 0.079475  
## iter 90 value 0.079357  
## iter 100 value 0.079280  
## final value 0.079280   
## stopped after 100 iterations  
## # weights: 271  
## initial value 21.964462   
## iter 10 value 1.680096  
## iter 20 value 0.472929  
## iter 30 value 0.141074  
## iter 40 value 0.083813  
## iter 50 value 0.078038  
## iter 60 value 0.076696  
## iter 70 value 0.076228  
## iter 80 value 0.075903  
## iter 90 value 0.075422  
## iter 100 value 0.074965  
## final value 0.074965   
## stopped after 100 iterations  
## # weights: 271  
## initial value 3.642738   
## iter 10 value 0.426174  
## iter 20 value 0.124242  
## iter 30 value 0.077795  
## iter 40 value 0.071089  
## iter 50 value 0.069574  
## iter 60 value 0.068963  
## iter 70 value 0.068720  
## iter 80 value 0.068606  
## iter 90 value 0.068539  
## iter 100 value 0.068503  
## final value 0.068503   
## stopped after 100 iterations  
## # weights: 271  
## initial value 13.158712   
## iter 10 value 0.451571  
## iter 20 value 0.129317  
## iter 30 value 0.089076  
## iter 40 value 0.084959  
## iter 50 value 0.083409  
## iter 60 value 0.082615  
## iter 70 value 0.082283  
## iter 80 value 0.082116  
## iter 90 value 0.082025  
## iter 100 value 0.081913  
## final value 0.081913   
## stopped after 100 iterations  
## # weights: 271  
## initial value 12.645853   
## iter 10 value 0.485891  
## iter 20 value 0.134423  
## iter 30 value 0.097173  
## iter 40 value 0.089889  
## iter 50 value 0.088113  
## iter 60 value 0.087044  
## iter 70 value 0.086461  
## iter 80 value 0.085734  
## iter 90 value 0.085132  
## iter 100 value 0.084945  
## final value 0.084945   
## stopped after 100 iterations  
## # weights: 271  
## initial value 15.736014   
## iter 10 value 0.428039  
## iter 20 value 0.144336  
## iter 30 value 0.101542  
## iter 40 value 0.094214  
## iter 50 value 0.092471  
## iter 60 value 0.091719  
## iter 70 value 0.091272  
## iter 80 value 0.091089  
## iter 90 value 0.091015  
## iter 100 value 0.090978  
## final value 0.090978   
## stopped after 100 iterations

model.meansquarederror

## NNet  
## Fold1 0.023161749  
## Fold2 0.043680538  
## Fold3 0.020307371  
## Fold4 0.023811570  
## Fold5 0.007866919  
## Fold6 0.027924399  
## Fold7 0.036602239  
## Fold8 0.005888398  
## Fold9 0.005277719  
## Fold10 0.001916918

mean(model.meansquarederror)

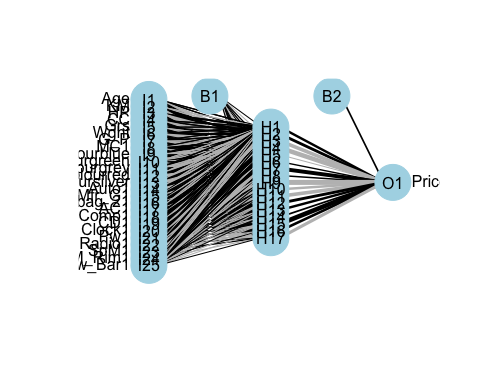
## [1] 0.01964378

**#\*\*\*\*\*\*\*\*\*\*\*\*\* Neural Net with 17 hidden neurons**  
nn2 <- nnet(Price ~ . , data= train, linout= T , size= 17, decay=0.01)

## # weights: 460  
## initial value 77.051114   
## iter 10 value 0.995366  
## iter 20 value 0.294480  
## iter 30 value 0.141737  
## iter 40 value 0.109988  
## iter 50 value 0.098667  
## iter 60 value 0.094532  
## iter 70 value 0.092805  
## iter 80 value 0.091967  
## iter 90 value 0.091502  
## iter 100 value 0.091075  
## final value 0.091075   
## stopped after 100 iterations

library(devtools)  
source\_url('https://gist.githubusercontent.com/fawda123/7471137/raw/466c1474d0a505ff044412703516c34f1a4684a5/nnet\_plot\_update.r')

plot.nnet(nn2)



#TEST   
nn.preds2<- predict(nn2, test)  
nn.preds2 #predicted values on test data

## [,1]  
## 29 0.30966331  
## 30 0.09928087  
## 31 0.03916540

#EVALUATE  
#MEAN SQUARED ERROR  
mse2 <- mean((nn.preds2 - test$Price)^2)   
mse2

## [1] 0.003882995

**#\*\*\*\*\*\*\*\*\*\*\*\*\* 10-Fold CV with 17 hidden neurons**  
k <- 10  
nmethod <- 1  
folds <- cut(seq(1,nrow(data)),breaks=k,labels=FALSE)   
model.meansquarederror <- matrix(-1, k, nmethod, dimnames=list(paste0("Fold", 1:k), c("NNet")))  
  
for(i in 1:k)  
{   
 testindexes <- which(folds == i, arr.ind=TRUE)   
 test <- data[testindexes, ]   
 train <- data[-testindexes, ]   
   
 nnModel<- nnet(Price ~ . , data= train, linout= T, size= 17, decay=0.01)  
 predicted <- predict(nnModel, test)  
 model.meansquarederror[i] <- mean((test$Price - predicted)^2)  
}

## # weights: 460  
## initial value 13.839873   
## iter 10 value 0.757658  
## iter 20 value 0.187545  
## iter 30 value 0.094094  
## iter 40 value 0.085517  
## iter 50 value 0.083944  
## iter 60 value 0.083202  
## iter 70 value 0.082985  
## iter 80 value 0.082790  
## iter 90 value 0.082637  
## iter 100 value 0.082551  
## final value 0.082551   
## stopped after 100 iterations  
## # weights: 460  
## initial value 34.389326   
## iter 10 value 0.572846  
## iter 20 value 0.185774  
## iter 30 value 0.093339  
## iter 40 value 0.078911  
## iter 50 value 0.075698  
## iter 60 value 0.074767  
## iter 70 value 0.073868  
## iter 80 value 0.073209  
## iter 90 value 0.073004  
## iter 100 value 0.072913  
## final value 0.072913   
## stopped after 100 iterations  
## # weights: 460  
## initial value 157.216064   
## iter 10 value 1.051028  
## iter 20 value 0.350676  
## iter 30 value 0.217346  
## iter 40 value 0.174619  
## iter 50 value 0.141981  
## iter 60 value 0.111726  
## iter 70 value 0.097444  
## iter 80 value 0.092850  
## iter 90 value 0.090679  
## iter 100 value 0.089063  
## final value 0.089063   
## stopped after 100 iterations  
## # weights: 460  
## initial value 13.088746   
## iter 10 value 0.735209  
## iter 20 value 0.183775  
## iter 30 value 0.091484  
## iter 40 value 0.080885  
## iter 50 value 0.079262  
## iter 60 value 0.078807  
## iter 70 value 0.078535  
## iter 80 value 0.078324  
## iter 90 value 0.078056  
## iter 100 value 0.077892  
## final value 0.077892   
## stopped after 100 iterations  
## # weights: 460  
## initial value 6.811318   
## iter 10 value 0.694724  
## iter 20 value 0.171642  
## iter 30 value 0.088748  
## iter 40 value 0.081266  
## iter 50 value 0.080034  
## iter 60 value 0.079498  
## iter 70 value 0.079155  
## iter 80 value 0.078892  
## iter 90 value 0.078687  
## iter 100 value 0.078550  
## final value 0.078550   
## stopped after 100 iterations  
## # weights: 460  
## initial value 20.244100   
## iter 10 value 0.841770  
## iter 20 value 0.226039  
## iter 30 value 0.091556  
## iter 40 value 0.078021  
## iter 50 value 0.075362  
## iter 60 value 0.074265  
## iter 70 value 0.073846  
## iter 80 value 0.073754  
## iter 90 value 0.073696  
## iter 100 value 0.073606  
## final value 0.073606   
## stopped after 100 iterations  
## # weights: 460  
## initial value 25.317101   
## iter 10 value 0.758099  
## iter 20 value 0.162044  
## iter 30 value 0.078272  
## iter 40 value 0.070130  
## iter 50 value 0.068747  
## iter 60 value 0.068202  
## iter 70 value 0.068022  
## iter 80 value 0.067880  
## iter 90 value 0.067697  
## iter 100 value 0.067590  
## final value 0.067590   
## stopped after 100 iterations  
## # weights: 460  
## initial value 21.031273   
## iter 10 value 0.766063  
## iter 20 value 0.254392  
## iter 30 value 0.096922  
## iter 40 value 0.083629  
## iter 50 value 0.081634  
## iter 60 value 0.080957  
## iter 70 value 0.080738  
## iter 80 value 0.080572  
## iter 90 value 0.080432  
## iter 100 value 0.080374  
## final value 0.080374   
## stopped after 100 iterations  
## # weights: 460  
## initial value 6.646388   
## iter 10 value 0.731647  
## iter 20 value 0.220855  
## iter 30 value 0.104485  
## iter 40 value 0.087805  
## iter 50 value 0.085421  
## iter 60 value 0.084562  
## iter 70 value 0.084114  
## iter 80 value 0.083949  
## iter 90 value 0.083802  
## iter 100 value 0.083720  
## final value 0.083720   
## stopped after 100 iterations  
## # weights: 460  
## initial value 2.275524   
## iter 10 value 0.706278  
## iter 20 value 0.185608  
## iter 30 value 0.101318  
## iter 40 value 0.092713  
## iter 50 value 0.091141  
## iter 60 value 0.090468  
## iter 70 value 0.090228  
## iter 80 value 0.090078  
## iter 90 value 0.089941  
## iter 100 value 0.089857  
## final value 0.089857   
## stopped after 100 iterations

model.meansquarederror

## NNet  
## Fold1 0.022535006  
## Fold2 0.043663383  
## Fold3 0.017654073  
## Fold4 0.022701139  
## Fold5 0.007850738  
## Fold6 0.026145582  
## Fold7 0.036080338  
## Fold8 0.005980705  
## Fold9 0.005108235  
## Fold10 0.003425833

mean(model.meansquarederror)

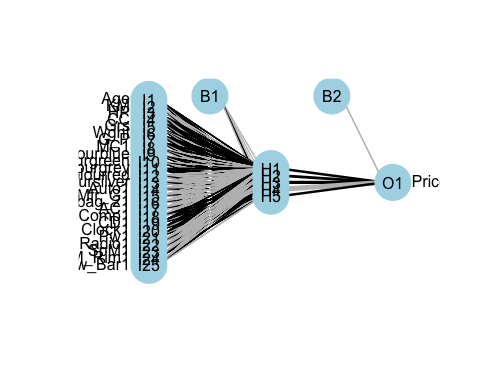
## [1] 0.0191145

**#\*\*\*\*\*\*\*\*\*\*\*\*\* Neural Net with 10 hidden neurons**  
nn3 <- nnet(Price ~ . , data= train, linout= T , size= 5, decay=0.01)

## # weights: 136  
## initial value 21.711711   
## iter 10 value 0.410980  
## iter 20 value 0.147346  
## iter 30 value 0.103705  
## iter 40 value 0.098623  
## iter 50 value 0.096664  
## iter 60 value 0.095686  
## iter 70 value 0.094906  
## iter 80 value 0.094486  
## iter 90 value 0.094105  
## iter 100 value 0.093809  
## final value 0.093809   
## stopped after 100 iterations

library(devtools)  
source\_url('https://gist.githubusercontent.com/fawda123/7471137/raw/466c1474d0a505ff044412703516c34f1a4684a5/nnet\_plot\_update.r')

plot.nnet(nn3)



#TEST   
nn.preds3<- predict(nn3, test)  
nn.preds3 #predicted values on test data

## [,1]  
## 29 0.31180098  
## 30 0.14114837  
## 31 0.08195597

#EVALUATE  
#MEAN SQUARED ERROR  
mse3 <- mean((nn.preds3 - test$Price)^2)   
mse3

## [1] 0.0009641003

**#\*\*\*\*\*\*\*\*\*\*\*\*10-Fold CV with 5 hidden neurons**  
k <- 10  
nmethod <- 1  
folds <- cut(seq(1,nrow(data)),breaks=k,labels=FALSE)   
model.meansquarederror <- matrix(-1, k, nmethod, dimnames=list(paste0("Fold", 1:k), c("NNet")))  
  
for(i in 1:k)  
{   
 testindexes <- which(folds == i, arr.ind=TRUE)   
 test <- data[testindexes, ]   
 train <- data[-testindexes, ]   
   
 nnModel<- nnet(Price ~ . , data= train, linout= T, size= 5, decay=0.01)  
 predicted <- predict(nnModel, test)  
 model.meansquarederror[i] <- mean((test$Price - predicted)^2)  
}

## # weights: 136  
## initial value 25.249575   
## iter 10 value 0.330752  
## iter 20 value 0.128456  
## iter 30 value 0.094449  
## iter 40 value 0.089451  
## iter 50 value 0.086510  
## iter 60 value 0.085595  
## iter 70 value 0.085396  
## iter 80 value 0.085328  
## iter 90 value 0.085276  
## iter 100 value 0.085207  
## final value 0.085207   
## stopped after 100 iterations  
## # weights: 136  
## initial value 3.961314   
## iter 10 value 0.280435  
## iter 20 value 0.108600  
## iter 30 value 0.079040  
## iter 40 value 0.075762  
## iter 50 value 0.075047  
## iter 60 value 0.074796  
## iter 70 value 0.074646  
## iter 80 value 0.074547  
## iter 90 value 0.074510  
## iter 100 value 0.074473  
## final value 0.074473   
## stopped after 100 iterations  
## # weights: 136  
## initial value 9.218478   
## iter 10 value 0.312561  
## iter 20 value 0.123678  
## iter 30 value 0.092948  
## iter 40 value 0.089286  
## iter 50 value 0.088381  
## iter 60 value 0.088027  
## iter 70 value 0.087747  
## iter 80 value 0.087459  
## iter 90 value 0.087226  
## iter 100 value 0.086993  
## final value 0.086993   
## stopped after 100 iterations  
## # weights: 136  
## initial value 16.907084   
## iter 10 value 0.292709  
## iter 20 value 0.117514  
## iter 30 value 0.090917  
## iter 40 value 0.084955  
## iter 50 value 0.082215  
## iter 60 value 0.081241  
## iter 70 value 0.080847  
## iter 80 value 0.080546  
## iter 90 value 0.080381  
## iter 100 value 0.080271  
## final value 0.080271   
## stopped after 100 iterations  
## # weights: 136  
## initial value 5.250299   
## iter 10 value 0.298401  
## iter 20 value 0.103878  
## iter 30 value 0.084327  
## iter 40 value 0.082037  
## iter 50 value 0.081323  
## iter 60 value 0.081171  
## iter 70 value 0.081112  
## iter 80 value 0.081060  
## iter 90 value 0.080992  
## iter 100 value 0.080938  
## final value 0.080938   
## stopped after 100 iterations  
## # weights: 136  
## initial value 4.751491   
## iter 10 value 0.457566  
## iter 20 value 0.139629  
## iter 30 value 0.081870  
## iter 40 value 0.077293  
## iter 50 value 0.076028  
## iter 60 value 0.075603  
## iter 70 value 0.075355  
## iter 80 value 0.075220  
## iter 90 value 0.075059  
## iter 100 value 0.074841  
## final value 0.074841   
## stopped after 100 iterations  
## # weights: 136  
## initial value 2.469193   
## iter 10 value 0.262399  
## iter 20 value 0.095556  
## iter 30 value 0.075737  
## iter 40 value 0.072416  
## iter 50 value 0.070746  
## iter 60 value 0.070268  
## iter 70 value 0.069958  
## iter 80 value 0.069808  
## iter 90 value 0.069679  
## iter 100 value 0.069545  
## final value 0.069545   
## stopped after 100 iterations  
## # weights: 136  
## initial value 5.617113   
## iter 10 value 0.264567  
## iter 20 value 0.104486  
## iter 30 value 0.087750  
## iter 40 value 0.085591  
## iter 50 value 0.084566  
## iter 60 value 0.084171  
## iter 70 value 0.083912  
## iter 80 value 0.083538  
## iter 90 value 0.083406  
## iter 100 value 0.083362  
## final value 0.083362   
## stopped after 100 iterations  
## # weights: 136  
## initial value 23.563796   
## iter 10 value 0.284763  
## iter 20 value 0.119996  
## iter 30 value 0.099330  
## iter 40 value 0.089865  
## iter 50 value 0.086816  
## iter 60 value 0.086238  
## iter 70 value 0.086130  
## iter 80 value 0.086081  
## iter 90 value 0.086058  
## iter 100 value 0.086048  
## final value 0.086048   
## stopped after 100 iterations  
## # weights: 136  
## initial value 42.985507   
## iter 10 value 2.164800  
## iter 20 value 0.243599  
## iter 30 value 0.119342  
## iter 40 value 0.099411  
## iter 50 value 0.096718  
## iter 60 value 0.095949  
## iter 70 value 0.095695  
## iter 80 value 0.095528  
## iter 90 value 0.095386  
## iter 100 value 0.095223  
## final value 0.095223   
## stopped after 100 iterations

model.meansquarederror

## NNet  
## Fold1 0.0235517303  
## Fold2 0.0431945467  
## Fold3 0.0196130912  
## Fold4 0.0237192556  
## Fold5 0.0079235616  
## Fold6 0.0283641162  
## Fold7 0.0373013344  
## Fold8 0.0057985268  
## Fold9 0.0054779648  
## Fold10 0.0005862518

mean(model.meansquarederror)

## [1] 0.01955304

#\*\*\*\*\*\*\*\*\*\*\*\*\*  
  
**#\*\*\*\*\*\*\*\*\*\* LINEAR REGRESSION**   
set.seed(3)  
indx<- sample(2, nrow(data), replace=T, prob=c(0.8,0.2))  
train <- data[indx==1, ]  
test <- data[indx==2, ]  
  
lmModel <- lm(Price ~ . , data=train)  
summary(lmModel)

##   
## Call:  
## lm(formula = Price ~ ., data = train)  
##   
## Residuals:  
## 1 3 4 5 6 7 8   
## 6.939e-18 -6.452e-18 -4.916e-02 3.693e-02 1.223e-02 -1.488e-17 6.781e-18   
## 9 10 11 12 13 14 17   
## -2.184e-17 3.997e-02 -3.997e-02 5.572e-18 1.552e-02 8.941e-03 6.994e-17   
## 18 20 21 22 23 24 25   
## -3.997e-02 1.552e-02 -8.305e-18 2.446e-02 6.467e-02 -7.690e-02 -1.494e-17   
## 26 27 29 30 31   
## 1.220e-02 -2.442e-02 5.194e-18 3.484e-03 -3.484e-03   
##   
## Coefficients: (3 not defined because of singularities)  
## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) 0.003361 0.243574 0.014 0.990  
## Age 0.010261 0.173732 0.059 0.957  
## KM -0.186776 0.226134 -0.826 0.469  
## HP 0.333139 0.651612 0.511 0.644  
## CC 0.160959 0.464012 0.347 0.752  
## Grs 0.031875 0.188652 0.169 0.877  
## Wght 0.248436 0.504347 0.493 0.656  
## G\_P 0.202124 0.339409 0.596 0.593  
## MC1 0.023039 0.083575 0.276 0.801  
## Colourblue 0.025264 0.113759 0.222 0.839  
## Colourgreen -0.023167 0.140968 -0.164 0.880  
## Colourgrey 0.050826 0.078681 0.646 0.564  
## Colourred -0.135386 0.160260 -0.845 0.460  
## Coloursilver -0.220539 0.168042 -1.312 0.281  
## Auto1 -0.036882 0.262313 -0.141 0.897  
## Mfr\_G1 0.038699 0.136976 0.283 0.796  
## Abag\_21 0.188468 0.331128 0.569 0.609  
## AC1 -0.050802 0.188658 -0.269 0.805  
## Comp1 NA NA NA NA  
## CD1 0.048818 0.069688 0.701 0.534  
## Clock1 0.069966 0.191941 0.365 0.740  
## Pw1 NA NA NA NA  
## Radio1 NA NA NA NA  
## SpM1 -0.095945 0.155232 -0.618 0.580  
## M\_Rim1 -0.044906 0.096624 -0.465 0.674  
## Tow\_Bar1 0.014112 0.125582 0.112 0.918  
##   
## Residual standard error: 0.08317 on 3 degrees of freedom  
## Multiple R-squared: 0.9889, Adjusted R-squared: 0.9072   
## F-statistic: 12.11 on 22 and 3 DF, p-value: 0.03119

predictions <- predict(lmModel, test)

## Warning in predict.lm(lmModel, test): prediction from a rank-deficient fit may  
## be misleading

predictions

## 2 15 16 19 28   
## 0.6386574 0.4717386 0.4027741 0.3397393 0.3255323

LMmse <- mean((predictions - test$Price)^2)   
LMmse

## [1] 0.01054387

**Results:**

|  |  |
| --- | --- |
| Model | MSE |
| Hidden Neurons=10, decay=0.01 | 0.01431462 |
| Hidden Neurons=17, decay=0.01 | 0.01433148 |
| Hidden Neurons=5, decay=0.01 | 0.01399435 |
| 10-fold CV, with Hidden Neurons=10, decay=0.01 | 0.01961636 |
| 10-fold CV, with Hidden Neurons=17, decay=0.01 | 0.01933614 |
| 10-fold CV, with Hidden Neurons=5, decay=0.01 | 0.01956156 |
| Linear Regression | 0.01054387 |