Neural Networks forest fire

##Forest fire with the neural networks

##neural networks on the forest fires

data <- read.csv(file.choose())

View(data)

str(data)##two independent variable are factor that is days and months which does not have direct influence on the forest fire

attach(data)

## observing the variables in data set##

##we have removed the variable which are not useful for model##

##EDA

DataExplorer::plot\_bar(data)

DataExplorer::plot\_histogram(data)

DataExplorer::plot\_correlation(data)

DataExplorer::plot\_missing(data)

## No missing variables

data1 <- data[,-c(1,2)]

attach(data1)

##normalization ##

normalize<-function(x){

return ( (x-min(x))/(max(x)-min(x)))

}

datanorm <- as.data.frame(lapply(data1[,-9],FUN=normalize))

datanorm <- cbind(datanorm,data$area)

colnames(datanorm)[9] <- "area"

##splitting the data for the model##

traindata <- datanorm[1:350,]

testdata <- datanorm[350:517,]

##model##

library(neuralnet)

model <- neuralnet(area~.,data = traindata,hidden = 4,linear.output = FALSE)

plot(model)

str(model)

##evaluating the model##

set.seed(123)

pre <- compute(model,testdata[1:8])

pre1 <- pre$net.result

cor(pre1,testdata$area)##0.06 indicating the model accuracy is very low

plot(pre1,testdata$area)

## i have change tunning parameter and checked the data but there is no improvement in the correlation

model1 <- neuralnet(area~.,data = traindata,hidden = 7,linear.output = FALSE)

plot(model1)

##evaluating the model##

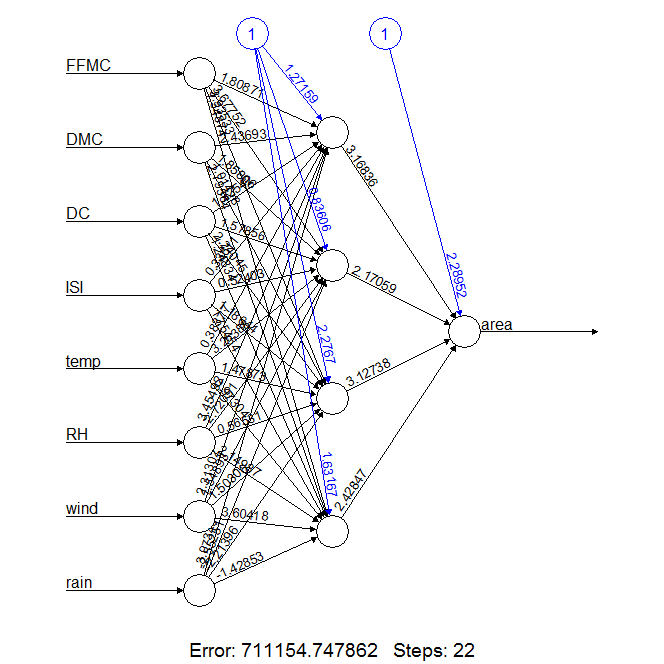
set.seed(123)

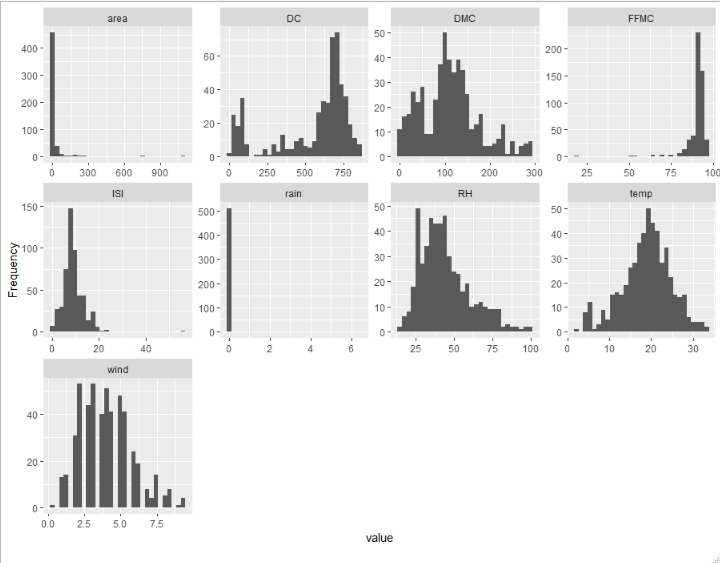
pre2 <- compute(model1,testdata[1:8])

pre3 <- pre2$net.result

cor(pre3,testdata$area)

plot(pre1,testdata$area)





50 startups

##importing the data#3

data <- read.csv(file.choose())

View(data)

str(data)

attach(data)

##state variable is not influencing the profit so we can remove state variable##

data1 <- data[,-4]

##normalize the data##

datanor <- scale(data1[,-4])

View(datanor)

normalize<-function(x){

return ( (x-min(x))/(max(x)-min(x)))

}

nor <- as.data.frame(lapply(data[,-c(4,5)],FUN = normalize))

View(nor)

nor <- cbind(nor,data$Profit)

colnames(nor)[4] <- "profit"

##spliting the data for model building##

traindata <- nor[1:35,]

testdata <- nor[35:50,]

##model buliding##

library(neuralnet)

model <- neuralnet(traindata$profit~.,data = traindata[,-4],linear.output = FALSE)

model

plot(model)

##prdection of the test data##

pre <- compute(model,testdata[1:3])

str(pre)

pre <- pre$net.result

##evaluation of the model##

cor(pre,testdata$profit)##.3856361

plot(pre,testdata$profit)

##model bulding with the tunnig parameter

model1 <- neuralnet(traindata$profit~.,data = traindata[,-4],hidden=7,linear.output = FALSE)

model1

plot(model1)

##prdection of the test data##

pre1 <- compute(model1,testdata[1:3])

str(pre)

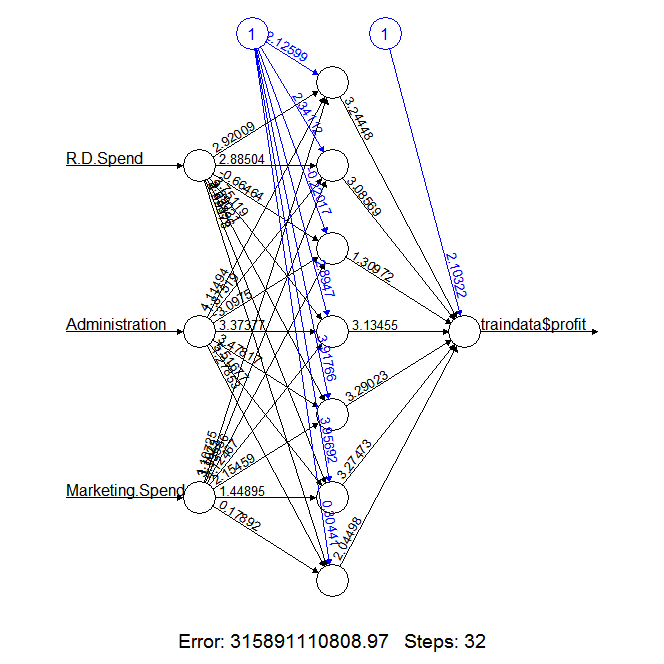
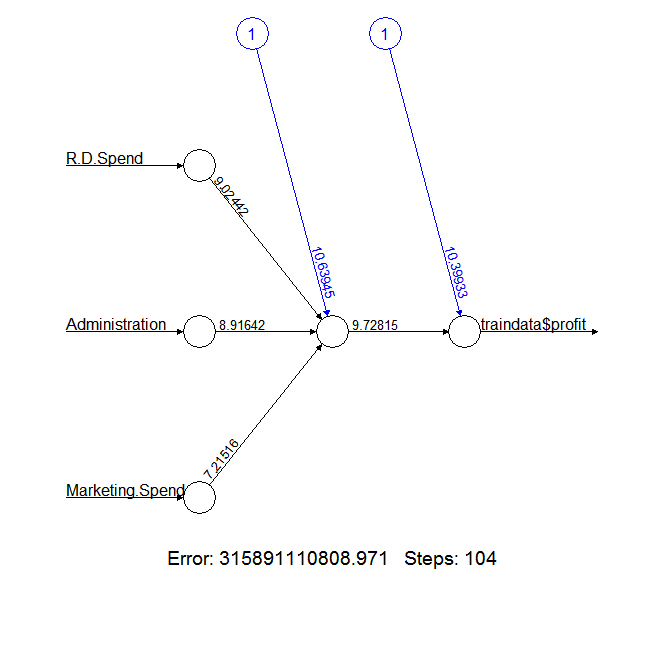
pre1 <- pre1$net.result

##evaluation of the model##

cor(pre1,testdata$profit)##.57

plot(pre1,testdata$profit)

## the accuracy of the model is 57 which is very i have tryed trial and error for more accuracy this the best value i got



Concrete

concrete <- read.csv(file.choose())

View(concrete)

str(concrete)

#attach(concrete)

#normal\_concrete<-scale(concrete)

## or

normalize<-function(x){

return ( (x-min(x))/(max(x)-min(x)))

}

concrete\_norm<-as.data.frame(lapply(concrete[,-9],FUN=normalize))

#summary(concrete\_norm$strength)

#summary(normal\_concrete)

summary(concrete$strength)

concrete\_norm <- cbind(concrete\_norm,concrete$strength)

colnames(concrete\_norm)[9] <- "strength"

concrete\_train<-concrete\_norm[1:773,]

concrete\_test<-concrete\_norm[774:1030,]

# Using multilayered feed forward nueral network

# package nueralnet

# install.packages("neuralnet")

# install.packages("nnet")

library(neuralnet) # regression

library(nnet) # classification

# Building model

formula\_nn <- paste("strength",paste(colnames(concrete[-9]),collapse ="+"),sep="~")

#concrete\_model <- neuralnet(strength~cement+slag+ash+water+superplastic+coarseagg+fineagg+age,data = concrete\_train)

concrete\_model <- neuralnet(formula = formula\_nn,data = concrete\_train)

str(concrete\_model)

plot(concrete\_model)

# SSE sum of squared errors . least SSE best model

# Evaluating model performance

# compute function to generate ouput for the model prepared

set.seed(12323)

model\_results <- compute(concrete\_model,concrete\_test[1:8])

str(model\_results)

predicted\_strength <- model\_results$net.result

# predicted\_strength

# model\_results$neurons

cor(predicted\_strength,concrete\_test$strength)##0.72

plot(predicted\_strength,concrete\_test$strength)

model\_5<-neuralnet(strength~cement+slag+ash+water+superplastic+coarseagg+fineagg+age,data= concrete\_norm,hidden = 5)

plot(model\_5)

model\_5\_res<-compute(model\_5,concrete\_test[1:8])

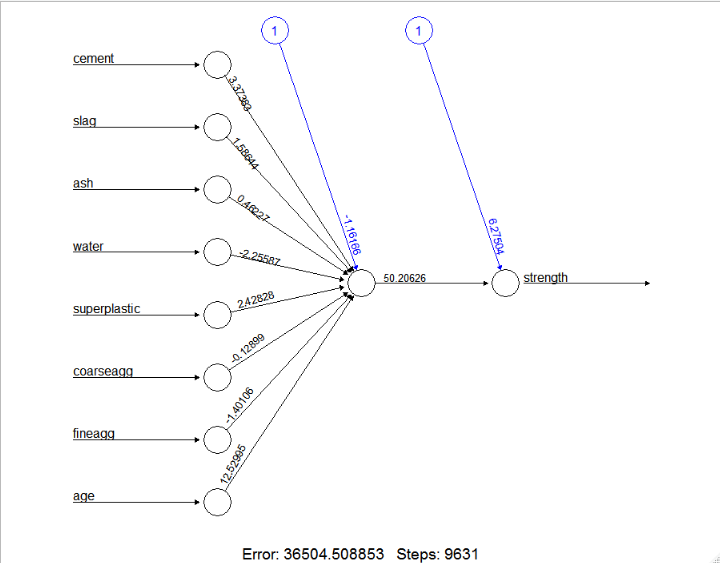
pred\_strn\_5<-model\_5\_res$net.result

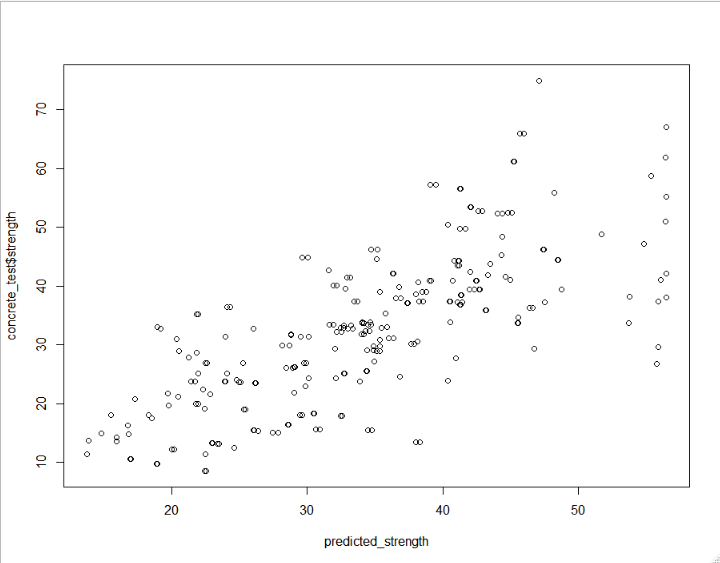
cor(pred\_strn\_5,concrete\_test$strength)

plot(pred\_strn\_5,concrete\_test$strength)

# SSE has reduced and training steps had been increased as the number of nuerons

# under hidden layer are increased





The plots of the other model cannot not be pasted because my system is hanging while performing the operation on it so I have executed the same in the my friend laptop writing the asnwer