file <- file.choose()  
library(readxl)

## Warning: package 'readxl' was built under R version 4.2.2

data <- read\_excel(file, sheet =1)

#checking null/NA values   
table(is.na(data))

##   
## FALSE   
## 868

## Removing NA values  
which(is.na(data))

## integer(0)

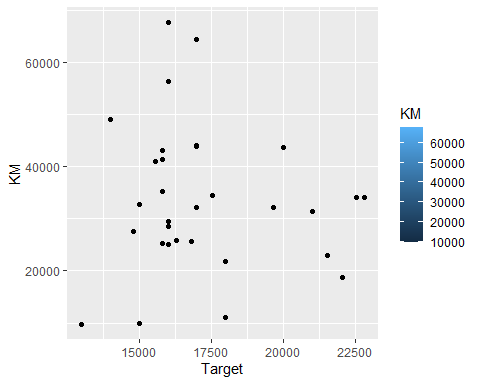
#renaming the price to target  
colnames(data)[1] <- "Target"  
#balancing the data sets  
summary(data$Target)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 13000 15800 16300 17251 18000 22800

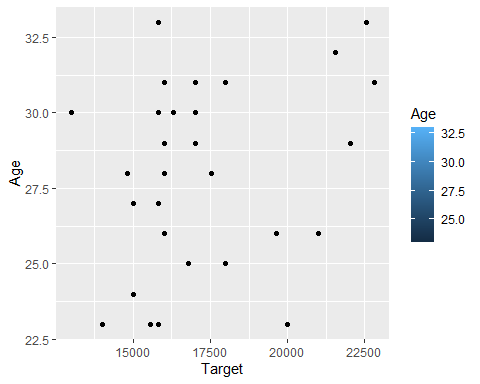
library(ROSE)  
  
FData <- data  
FullData <- data  
sampledata <- data

#plotting graphs for analysis and better visualization #Price vs KM

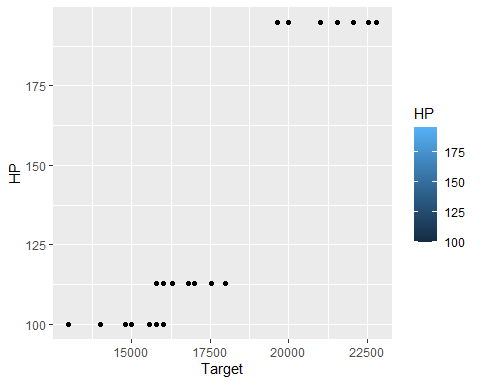
library(ggplot2)  
  
p1<-ggplot(data=data, aes(x=Target,   
 y=`KM`, fill=`KM`)) +  
 geom\_point( stat="identity")  
  
p1

 #Price vs Age

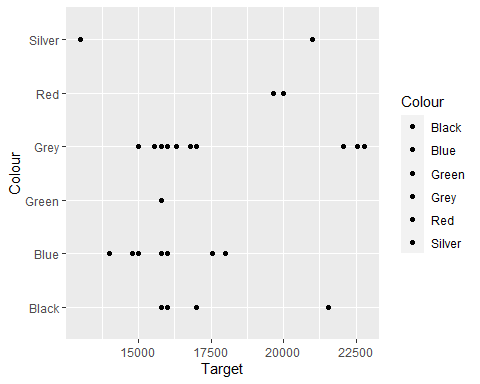
p2<-ggplot(data=data, aes(x=Target,   
 y=`Age`, fill=`Age`)) +  
 geom\_point( stat="identity")  
  
p2

 #Price vs Horsepower

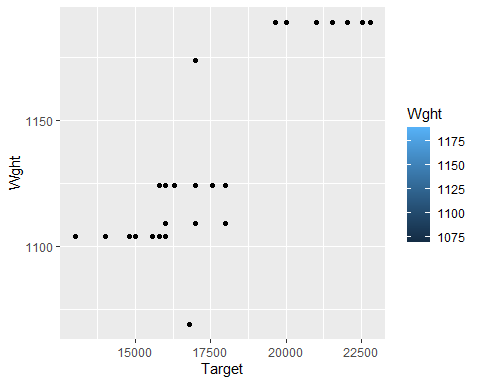
p3<-ggplot(data=data, aes(x=Target,   
 y=`HP`, fill=`HP`)) +  
 geom\_point( stat="identity")  
p3

 #Price vs COLOR

p4<-ggplot(data=data, aes(x=Target,   
 y=`Colour`, fill=`Colour`)) +  
 geom\_point( stat="identity")  
p4

 #Price vs Wght

p5<-ggplot(data=data, aes(x=Target,   
 y=`Wght`, fill=`Wght`)) +  
 geom\_point( stat="identity")  
p5

 # Plotting Correlation graphs of sample order data

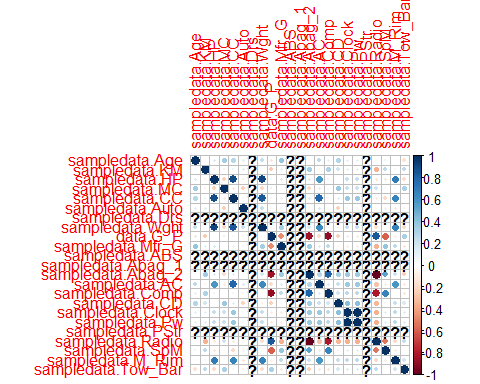
df1 <- data.frame(sampledata$Age, sampledata$KM,sampledata$HP, sampledata$MC, sampledata$CC, sampledata$Auto, sampledata$Drs, sampledata$Wght, data$G\_P, sampledata$Mfr\_G, sampledata$ABS, sampledata$Abag\_1, sampledata$Abag\_2, sampledata$AC, sampledata$Comp, sampledata$CD, sampledata$Clock, sampledata$Pw, sampledata$PStr, sampledata$Radio, sampledata$SpM, sampledata$M\_Rim, sampledata$Tow\_Bar)  
df1.cor = cor(df1 , use = "pairwise.complete.obs")

## Warning in cor(df1, use = "pairwise.complete.obs"): the standard deviation is  
## zero

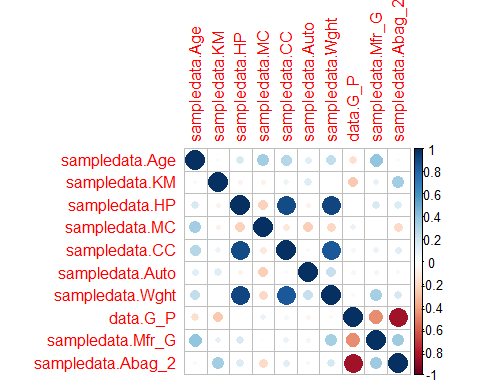
library(corrplot)

## corrplot 0.92 loaded

corrplot(df1.cor)



df3 <- data.frame(sampledata$Age, sampledata$KM,sampledata$HP, sampledata$MC, sampledata$CC, sampledata$Auto, sampledata$Wght, data$G\_P, sampledata$Mfr\_G, sampledata$Abag\_2)  
df3.cor = cor(df3 , use = "pairwise.complete.obs")  
corrplot(df3.cor)



#based on the analysis- we ommit the one's that are not correlated.  
  
library(apaTables)  
apa.cor.table(df3,"APA correlation Table1.doc")

##   
##   
## Means, standard deviations, and correlations with confidence intervals  
##   
##   
## Variable M SD 1 2 3   
## 1. sampledata.Age 27.94 3.02   
##   
## 2. sampledata.KM 33627.68 13967.30 .04   
## [-.32, .39]   
##   
## 3. sampledata.HP 126.90 37.83 .16 -.06   
## [-.20, .49] [-.41, .30]   
##   
## 4. sampledata.MC 0.65 0.49 .35 -.07 -.22   
## [-.01, .62] [-.41, .29] [-.54, .14]  
##   
## 5. sampledata.CC 1574.19 152.68 .29 .07 .89\*\*   
## [-.08, .58] [-.29, .42] [.79, .95]   
##   
## 6. sampledata.Auto 0.03 0.18 .13 .14 -.07   
## [-.24, .46] [-.23, .47] [-.41, .29]  
##   
## 7. sampledata.Wght 1129.97 36.09 .25 -.01 .92\*\*   
## [-.12, .55] [-.36, .34] [.84, .96]   
##   
## 8. data.G\_P 4.90 3.18 -.16 -.26 .01   
## [-.49, .21] [-.56, .10] [-.35, .36]  
##   
## 9. sampledata.Mfr\_G 0.90 0.30 .40\* .10 .16   
## [.05, .66] [-.26, .44] [-.21, .49]  
##   
## 10. sampledata.Abag\_2 0.94 0.25 .04 .34 .14   
## [-.32, .39] [-.01, .62] [-.22, .47]  
##   
## 4 5 6 7 8 9   
##   
##   
##   
##   
##   
##   
##   
##   
##   
##   
##   
## -.13   
## [-.46, .24]   
##   
## -.25 .03   
## [-.55, .12] [-.33, .38]   
##   
## -.21 .85\*\* .23   
## [-.52, .16] [.70, .92] [-.14, .54]   
##   
## .08 .05 -.05 -.04   
## [-.28, .43] [-.31, .40] [-.40, .31] [-.39, .32]   
##   
## -.01 .09 .06 .33 -.46\*\*   
## [-.37, .34] [-.27, .43] [-.30, .41] [-.03, .61] [-.70, -.13]   
##   
## -.19 .13 .05 .17 -.85\*\* .36\*   
## [-.51, .17] [-.24, .46] [-.31, .40] [-.19, .50] [-.92, -.71] [.00, .63]  
##   
##   
## Note. M and SD are used to represent mean and standard deviation, respectively.  
## Values in square brackets indicate the 95% confidence interval.  
## The confidence interval is a plausible range of population correlations   
## that could have caused the sample correlation (Cumming, 2014).  
## \* indicates p < .05. \*\* indicates p < .01.  
##

#checking the model for all the data

library(dplyr)  
fullnormalization <- function(x)  
{  
 (x - min(x))/ (max(x)-min(x))  
}  
FullData <- FullData %>% mutate\_if(is.numeric, fullnormalization)  
summary(FullData)

## Target Age KM Fuel   
## Min. :0.0000 Min. :0.0000 Min. :0.0000 Length:31   
## 1st Qu.:0.2857 1st Qu.:0.3000 1st Qu.:0.2710 Class :character   
## Median :0.3367 Median :0.5000 Median :0.3880 Mode :character   
## Mean :0.4338 Mean :0.4935 Mean :0.4123   
## 3rd Qu.:0.5102 3rd Qu.:0.7000 3rd Qu.:0.5605   
## Max. :1.0000 Max. :1.0000 Max. :1.0000   
##   
## HP MC Colour Auto   
## Min. :0.0000 Min. :0.0000 Length:31 Min. :0.00000   
## 1st Qu.:0.0000 1st Qu.:0.0000 Class :character 1st Qu.:0.00000   
## Median :0.1368 Median :1.0000 Mode :character Median :0.00000   
## Mean :0.2832 Mean :0.6452 Mean :0.03226   
## 3rd Qu.:0.1368 3rd Qu.:1.0000 3rd Qu.:0.00000   
## Max. :1.0000 Max. :1.0000 Max. :1.00000   
##   
## 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 Mfr\_G ABS Abag\_1   
## Min. :0.00000 Min. :0.0000 Min. : NA Min. : NA   
## 1st Qu.:0.00000 1st Qu.:1.0000 1st Qu.: NA 1st Qu.: NA   
## Median :0.00000 Median :1.0000 Median : NA Median : NA   
## Mean :0.05645 Mean :0.9032 Mean :NaN Mean :NaN   
## 3rd Qu.:0.00000 3rd Qu.:1.0000 3rd Qu.: NA 3rd Qu.: NA   
## Max. :1.00000 Max. :1.0000 Max. : NA Max. : NA   
## NA's :31 NA's :31   
## Abag\_2 AC Comp CD   
## Min. :0.0000 Min. :0.0000 Min. :0.0000 Min. :0.0000   
## 1st Qu.:1.0000 1st Qu.:0.0000 1st Qu.:1.0000 1st Qu.:0.0000   
## Median :1.0000 Median :1.0000 Median :1.0000 Median :1.0000   
## Mean :0.9355 Mean :0.5484 Mean :0.9032 Mean :0.6129   
## 3rd Qu.:1.0000 3rd Qu.:1.0000 3rd Qu.:1.0000 3rd Qu.:1.0000   
## Max. :1.0000 Max. :1.0000 Max. :1.0000 Max. :1.0000   
##   
## Clock Pw PStr Radio   
## Min. :0.0000 Min. :0.0000 Min. : NA Min. :0.00000   
## 1st Qu.:1.0000 1st Qu.:1.0000 1st Qu.: NA 1st Qu.:0.00000   
## Median :1.0000 Median :1.0000 Median : NA Median :0.00000   
## Mean :0.9032 Mean :0.9032 Mean :NaN Mean :0.06452   
## 3rd Qu.:1.0000 3rd Qu.:1.0000 3rd Qu.: NA 3rd Qu.:0.00000   
## Max. :1.0000 Max. :1.0000 Max. : NA Max. :1.00000   
## NA's :31   
## SpM M\_Rim Tow\_Bar   
## Min. :0.0000 Min. :0.0000 Min. :0.0000   
## 1st Qu.:1.0000 1st Qu.:0.0000 1st Qu.:0.0000   
## Median :1.0000 Median :0.0000 Median :0.0000   
## Mean :0.8065 Mean :0.3871 Mean :0.1613   
## 3rd Qu.:1.0000 3rd Qu.:1.0000 3rd Qu.:0.0000   
## Max. :1.0000 Max. :1.0000 Max. :1.0000   
##

#creating a dummy variable to tackle colour  
  
GreynBlue<-ifelse(FullData$Colour == 'Grey' | FullData$Colour == 'Blue',1,0)  
FullData <- subset (FullData, select = -c(4,7,10,11,16,17,24))  
FullData <- FullData %>% mutate(  
 Target,  
 Age,  
 KM,  
 HP,  
 CC,  
 Grs=as.factor(Grs),  
 Wght,  
 G\_P,  
 Auto=as.factor(Auto),  
 MC=as.factor(MC),  
 Mfr\_G=as.factor(Mfr\_G),  
 AC=as.factor(AC),  
 Abag\_2=as.factor(Abag\_2),  
 AC=as.factor(AC),  
 CD=as.factor(CD),  
 Comp=as.factor(Comp),  
 Clock=as.factor(Clock),  
 Pw=as.factor(Pw),  
 Radio=as.factor(Radio),  
 SpM=as.factor(SpM),  
 M\_Rim=as.factor(M\_Rim),  
 Tow\_Bar=as.factor(Tow\_Bar),  
 GreynBlue=as.factor(GreynBlue)  
)  
#removed pstr as all are 1  
# divide into test and training data set  
set.seed(124)  
indfull <- sample(2, nrow(FullData), replace = T, prob = c(0.7, 0.3))  
trainfull <- FullData[indfull == 1, ]  
testfull <- FullData[indfull == 2, ]  
str(FullData)

## tibble [31 × 22] (S3: tbl\_df/tbl/data.frame)  
## $ Target : num [1:31] 0.816 0.714 0.679 0.872 0.974 ...  
## $ Age : num [1:31] 0.3 0 0.3 0.9 1 0.6 0.8 0.2 0.2 0.8 ...  
## $ KM : num [1:31] 0.375 0.585 0.387 0.229 0.421 ...  
## $ HP : num [1:31] 1 1 1 1 1 ...  
## $ MC : Factor w/ 2 levels "0","1": 1 1 1 2 2 1 2 2 1 2 ...  
## $ Auto : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ CC : num [1:31] 1 1 1 1 1 1 1 0.5 0.5 0.5 ...  
## $ Grs : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 1 1 1 1 ...  
## $ Wght : num [1:31] 1 1 1 1 1 ...  
## $ G\_P : num [1:31] 0.375 0 0 0 0 0 0 1 0 0 ...  
## $ 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 ...  
## $ GreynBlue: Factor w/ 2 levels "0","1": 1 1 1 1 2 2 2 2 2 2 ...

library(nnet)  
nnModelfull<- nnet(Target~ ., data = trainfull, linout = T, size = 10, decay = 0.01, maxit = 1000)

## # weights: 231  
## initial value 11.391039   
## iter 10 value 0.425949  
## iter 20 value 0.133393  
## iter 30 value 0.092372  
## iter 40 value 0.086703

……..  
## iter 460 value 0.083099  
## iter 460 value 0.083099  
## iter 460 value 0.083099  
## final value 0.083099   
## converged

summary(nnModelfull)

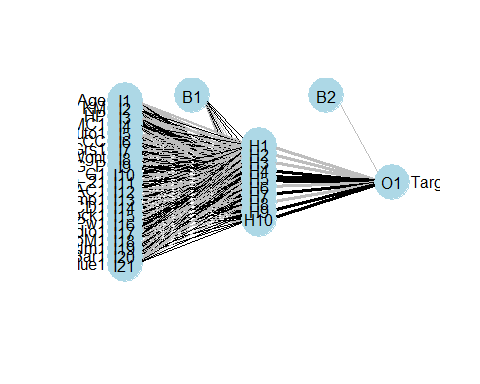
## a 21-10-1 network with 231 weights  
## options were - linear output units decay=0.01  
## b->h1 i1->h1 i2->h1 i3->h1 i4->h1 i5->h1 i6->h1 i7->h1 i8->h1 i9->h1   
## 0.05 -0.18 0.09 -0.22 0.01 0.00 -0.10 0.00 -0.11 -0.09   
## i10->h1 i11->h1 i12->h1 i13->h1 i14->h1 i15->h1 i16->h1 i17->h1 i18->h1 i19->h1   
## 0.10 -0.04 0.11 -0.09 -0.03 -0.05 -0.05 0.09 0.11 -0.01   
## i20->h1 i21->h1   
## -0.04 -0.10   
## b->h2 i1->h2 i2->h2 i3->h2 i4->h2 i5->h2 i6->h2 i7->h2 i8->h2 i9->h2   
## 0.05 -0.18 0.09 -0.22 0.01 0.00 -0.10 0.00 -0.11 -0.09   
## i10->h2 i11->h2 i12->h2 i13->h2 i14->h2 i15->h2 i16->h2 i17->h2 i18->h2 i19->h2   
## 0.10 -0.04 0.11 -0.09 -0.03 -0.05 -0.05 0.09 0.11 -0.01   
## i20->h2 i21->h2   
## -0.04 -0.10   
## b->h3 i1->h3 i2->h3 i3->h3 i4->h3 i5->h3 i6->h3 i7->h3 i8->h3 i9->h3   
## 0.05 -0.18 0.09 -0.22 0.01 0.00 -0.10 0.00 -0.11 -0.09   
## i10->h3 i11->h3 i12->h3 i13->h3 i14->h3 i15->h3 i16->h3 i17->h3 i18->h3 i19->h3   
## 0.10 -0.04 0.11 -0.09 -0.03 -0.05 -0.05 0.09 0.11 -0.01   
## i20->h3 i21->h3   
## -0.04 -0.10   
## b->h4 i1->h4 i2->h4 i3->h4 i4->h4 i5->h4 i6->h4 i7->h4 i8->h4 i9->h4   
## -0.03 0.16 -0.07 0.19 0.00 0.00 0.09 0.00 0.10 0.08   
## i10->h4 i11->h4 i12->h4 i13->h4 i14->h4 i15->h4 i16->h4 i17->h4 i18->h4 i19->h4   
## -0.08 0.04 -0.09 0.08 0.03 0.05 0.05 -0.07 -0.09 0.02   
## i20->h4 i21->h4   
## 0.03 0.09   
## b->h5 i1->h5 i2->h5 i3->h5 i4->h5 i5->h5 i6->h5 i7->h5 i8->h5 i9->h5   
## -0.06 0.38 -0.26 0.54 -0.20 0.00 0.28 0.18 0.30 0.21   
## i10->h5 i11->h5 i12->h5 i13->h5 i14->h5 i15->h5 i16->h5 i17->h5 i18->h5 i19->h5   
## -0.18 0.12 -0.17 0.19 -0.06 0.15 0.15 -0.18 -0.13 0.10   
## i20->h5 i21->h5   
## 0.08 -0.24   
## b->h6 i1->h6 i2->h6 i3->h6 i4->h6 i5->h6 i6->h6 i7->h6 i8->h6 i9->h6   
## 0.05 -0.18 0.09 -0.22 0.01 0.00 -0.10 0.00 -0.11 -0.09   
## i10->h6 i11->h6 i12->h6 i13->h6 i14->h6 i15->h6 i16->h6 i17->h6 i18->h6 i19->h6   
## 0.10 -0.04 0.11 -0.09 -0.03 -0.05 -0.05 0.09 0.11 -0.01   
## i20->h6 i21->h6   
## -0.04 -0.10   
## b->h7 i1->h7 i2->h7 i3->h7 i4->h7 i5->h7 i6->h7 i7->h7 i8->h7 i9->h7   
## -0.03 0.16 -0.07 0.19 0.00 0.00 0.09 0.00 0.10 0.08   
## i10->h7 i11->h7 i12->h7 i13->h7 i14->h7 i15->h7 i16->h7 i17->h7 i18->h7 i19->h7   
## -0.08 0.04 -0.09 0.08 0.03 0.05 0.05 -0.07 -0.09 0.02   
## i20->h7 i21->h7   
## 0.03 0.09   
## b->h8 i1->h8 i2->h8 i3->h8 i4->h8 i5->h8 i6->h8 i7->h8 i8->h8 i9->h8   
## 0.05 -0.19 0.09 -0.22 0.01 0.00 -0.10 0.00 -0.11 -0.09   
## i10->h8 i11->h8 i12->h8 i13->h8 i14->h8 i15->h8 i16->h8 i17->h8 i18->h8 i19->h8   
## 0.10 -0.04 0.11 -0.09 -0.03 -0.05 -0.05 0.09 0.11 -0.01   
## i20->h8 i21->h8   
## -0.04 -0.10   
## b->h9 i1->h9 i2->h9 i3->h9 i4->h9 i5->h9 i6->h9 i7->h9 i8->h9 i9->h9   
## -0.03 0.16 -0.07 0.19 0.00 0.00 0.09 0.00 0.10 0.08   
## i10->h9 i11->h9 i12->h9 i13->h9 i14->h9 i15->h9 i16->h9 i17->h9 i18->h9 i19->h9   
## -0.08 0.04 -0.09 0.08 0.03 0.05 0.05 -0.07 -0.09 0.02   
## i20->h9 i21->h9   
## 0.03 0.09   
## b->h10 i1->h10 i2->h10 i3->h10 i4->h10 i5->h10 i6->h10 i7->h10   
## -0.03 0.16 -0.07 0.19 0.00 0.00 0.09 0.00   
## i8->h10 i9->h10 i10->h10 i11->h10 i12->h10 i13->h10 i14->h10 i15->h10   
## 0.10 0.08 -0.08 0.04 -0.09 0.08 0.03 0.05   
## i16->h10 i17->h10 i18->h10 i19->h10 i20->h10 i21->h10   
## 0.05 -0.07 -0.09 0.02 0.03 0.09   
## b->o h1->o h2->o h3->o h4->o h5->o h6->o h7->o h8->o h9->o h10->o   
## -0.01 -0.47 -0.47 -0.47 0.40 0.99 -0.47 0.40 -0.47 0.40 0.40

nnModelfull

## a 21-10-1 network with 231 weights  
## inputs: Age KM HP MC1 Auto1 CC Grs1 Wght G\_P Mfr\_G1 Abag\_21 AC1 Comp1 CD1 Clock1 Pw1 Radio1 SpM1 M\_Rim1 Tow\_Bar1 GreynBlue1   
## output(s): Target   
## options were - linear output units decay=0.01

library(NeuralNetTools)

plotnet(nnModelfull)



nn.predsfull <- as.factor(predict(nnModelfull, testfull))  
nn.predsfull

## [1] 0.484072144750884 0.471060642832115 0.447890760999887   
## [4] 0.332599586829393 0.297394758030447 0.34001100018864   
## [7] 0.347288279037332 0.311538880101122 0.40202492114654   
## [10] -0.0205091688767445  
## 10 Levels: -0.0205091688767445 0.297394758030447 ... 0.484072144750884

#hidden neurons

set.seed(156)  
indxfull <- sample(2, nrow(trainfull), replace = T, prob = c(0.5, 0.5))  
train2full <- trainfull[indxfull == 1,]  
validationfull <- trainfull[indxfull == 2, ]  
  
errfull <- vector("numeric", 300)  
dfull <- seq(0.001, .1, length.out = 30)  
sizesfull <- seq(1,10, length.out = 10)  
k =1  
errfull\_df <- data.frame(  
 size=integer(),  
 decay=numeric(),  
 error = numeric()  
)  
  
for (s in sizesfull) {  
 for (i in dfull)  
 {  
 print(paste("K:", k, "i:", i,"size: ", s))  
 mymodelfull <-  
 nnet(  
 Target ~ .,  
 data = train2full,  
 decay = i,  
 size = s,  
 maxit = 1000,  
 trace = F  
 )  
 pred.classfull <- predict(mymodelfull, newdata = validationfull)  
 errfull\_tmp <-  
 round((sqrt(mean((pred.classfull - validationfull$Target) ^ 2  
 )) / mean(validationfull$Target)) \* 100, 2)  
 errfull\_df[nrow(errfull\_df)+1,] <- c(s,i,errfull\_tmp)  
 print(errfull\_tmp)  
 errfull[k]<-errfull\_tmp  
 k <- k + 1  
 }  
}

## [1] "K: 1 i: 0.001 size: 1"  
## [1] 36.04  
## [1] "K: 2 i: 0.00441379310344828 size: 1"  
## [1] 30.95  
## [1] "K: 3 i: 0.00782758620689655 size: 1"  
## [1] 29.63  
## [1] "K: 4 i: 0.0112413793103448 size: 1"  
## [1] 30.81  
## [1] "K: 5 i: 0.0146551724137931 size: 1"  
## [1] 29.19  
## [1] "K: 6 i: 0.0180689655172414 size: 1"  
## [1] 31.58  
## [1] "K: 7 i: 0.0214827586206897 size: 1"  
## [1] 32.2  
## [1] "K: 8 i: 0.0248965517241379 size: 1"  
## [1] 30.4  
## [1] "K: 9 i: 0.0283103448275862 size: 1"  
## [1] 31.03  
## [1] "K: 10 i: 0.0317241379310345 size: 1"  
## [1] 34.47  
and so on….

One can see the output in the rmd file

min(errfull)

## [1] 29.19

#check for the index of the minimum value and note the decay and size value  
View(errfull\_df)

#error for training data  
errtrainfull <- vector("numeric", 300)  
dtrainfull <- seq(0.001, .1, length.out = 30)  
sizestrainfull <- seq(1,10, length.out = 10)  
ktrainfull =1  
errtrainfull\_df <- data.frame(  
 size=integer(),  
 decay=numeric(),  
 error = numeric()  
)  
  
for (strainfull in sizestrainfull) {  
 for (itrainfull in dtrainfull)  
 {  
 print(paste("K:", ktrainfull, "i:", itrainfull,"size: ", strainfull))  
 mymodelfull\_train <-  
 nnet(  
 Target ~ .,  
 data = trainfull,  
 decay = itrainfull,  
 size = strainfull,  
 maxit = 1000,  
 trace = F  
 )  
 pred.classtrainfull <- predict(mymodelfull\_train, newdata = trainfull)  
 errtrainfull\_tmp <-  
 round((sqrt(mean((pred.classtrainfull - trainfull$Target) ^ 2  
 )) / mean(trainfull$Target)) \* 100, 2)  
 errtrainfull\_df[nrow(errtrainfull\_df)+1,] <- c(strainfull,itrainfull,errtrainfull\_tmp)  
 print(errtrainfull\_tmp)  
 errtrainfull[ktrainfull]<-errtrainfull\_tmp  
 ktrainfull <- ktrainfull + 1  
 }  
}

## [1] "K: 1 i: 0.001 size: 1"  
## [1] 9.85  
## [1] "K: 2 i: 0.00441379310344828 size: 1"  
## [1] 13.98  
## [1] "K: 3 i: 0.00782758620689655 size: 1"  
## [1] 16.31  
## [1] "K: 4 i: 0.0112413793103448 size: 1"  
## [1] 18.18  
## [1] "K: 5 i: 0.0146551724137931 size: 1"  
## [1] 19.84  
## [1] "K: 6 i: 0.0180689655172414 size: 1"  
## [1] 21.82  
## [1] "K: 7 i: 0.0214827586206897 size: 1"  
## [1] 22.95  
## [1] "K: 8 i: 0.0248965517241379 size: 1"  
## [1] 23.72  
## [1] "K: 9 i: 0.0283103448275862 size: 1"  
## [1] 24.73  
## [1] "K: 10 i: 0.0317241379310345 size: 1"  
## [1] 25.66  
…and so on

min(errtrainfull)

## [1] 4.73

View(errtrainfull\_df)

#error for test data

errtestfull <- vector("numeric", 300)  
dtestfull <- seq(0.001, .1, length.out = 30)  
sizestestfull <- seq(1,10, length.out = 10)  
ktestfull =1  
errtestfull\_df <- data.frame(  
 size=integer(),  
 decay=numeric(),  
 error = numeric()  
)  
  
for (stestfull in sizestestfull) {  
 for (itestfull in dtestfull)  
 {  
 print(paste("K:", ktestfull, "i:", itestfull,"size: ", stestfull))  
 mymodelfull\_test <-  
 nnet(  
 Target ~ .,  
 data = testfull,  
 decay = itestfull,  
 size = stestfull,  
 maxit = 1000,  
 trace = F  
 )  
 pred.classtestfull <- predict(mymodelfull\_test, newdata = testfull)  
 errtestfull\_tmp <-  
 round((sqrt(mean((pred.classtestfull - testfull$Target) ^ 2  
 )) / mean(testfull$Target)) \* 100, 2)  
 errtestfull\_df[nrow(errtestfull\_df)+1,] <- c(stestfull,itestfull,errtestfull\_tmp)  
 print(errtestfull\_tmp)  
 errtestfull[ktestfull]<-errtestfull\_tmp  
 ktestfull <- ktestfull + 1  
 }  
}

## [1] "K: 1 i: 0.001 size: 1"  
## [1] 4.87  
## [1] "K: 2 i: 0.00441379310344828 size: 1"  
## [1] 10.24  
## [1] "K: 3 i: 0.00782758620689655 size: 1"  
## [1] 14.29  
## [1] "K: 4 i: 0.0112413793103448 size: 1"  
## [1] 17.41  
## [1] "K: 5 i: 0.0146551724137931 size: 1"  
## [1] 19.95  
## [1] "K: 6 i: 0.0180689655172414 size: 1"  
## [1] 22.05  
## [1] "K: 7 i: 0.0214827586206897 size: 1"  
## [1] 23.74  
## [1] "K: 8 i: 0.0248965517241379 size: 1"  
## [1] 25.01  
## [1] "K: 9 i: 0.0283103448275862 size: 1"  
## [1] 25.88  
## [1] "K: 10 i: 0.0317241379310345 size: 1"  
## [1] 26.43  
…and so on  
testerrfull<-min(errtestfull)  
testerrfull

## [1] 2.81

View(errtestfull\_df)

#based on the d and size value for min error in training data set, we calculate

library(nnet)  
nnModelfullbest<- nnet(Target~ ., data = trainfull, linout = T, size = 5, decay = 0.00100, maxit = 1000)

## # weights: 116  
## initial value 9.882837   
## iter 10 value 0.133487  
## iter 20 value 0.050852  
## iter 30 value 0.032759  
## iter 40 value 0.026549  
## iter 50 value 0.021695

………………………….

## iter 570 value 0.015265  
## iter 580 value 0.015265  
## final value 0.015265   
## converged

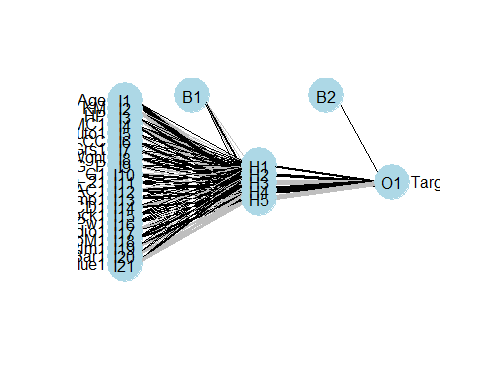
summary(nnModelfullbest)

## a 21-5-1 network with 116 weights  
## options were - linear output units decay=0.001  
## b->h1 i1->h1 i2->h1 i3->h1 i4->h1 i5->h1 i6->h1 i7->h1 i8->h1 i9->h1   
## -0.01 0.05 -0.03 0.10 0.11 0.00 0.09 0.01 0.04 0.02   
## i10->h1 i11->h1 i12->h1 i13->h1 i14->h1 i15->h1 i16->h1 i17->h1 i18->h1 i19->h1   
## -0.04 0.07 -0.07 0.11 0.13 0.08 0.08 -0.08 0.01 0.03   
## i20->h1 i21->h1   
## 0.02 0.18   
## b->h2 i1->h2 i2->h2 i3->h2 i4->h2 i5->h2 i6->h2 i7->h2 i8->h2 i9->h2   
## -0.03 -0.07 0.08 0.06 0.26 0.00 0.08 -0.06 0.01 0.00   
## i10->h2 i11->h2 i12->h2 i13->h2 i14->h2 i15->h2 i16->h2 i17->h2 i18->h2 i19->h2   
## -0.11 0.14 -0.10 0.22 0.25 0.15 0.15 -0.18 0.08 -0.05   
## i20->h2 i21->h2   
## 0.03 0.46   
## b->h3 i1->h3 i2->h3 i3->h3 i4->h3 i5->h3 i6->h3 i7->h3 i8->h3 i9->h3   
## 0.27 -0.82 0.75 -0.08 -0.29 0.00 -0.06 0.20 0.08 -0.06   
## i10->h3 i11->h3 i12->h3 i13->h3 i14->h3 i15->h3 i16->h3 i17->h3 i18->h3 i19->h3   
## 0.31 0.05 0.40 -0.05 -0.39 0.00 0.00 0.21 0.28 0.12   
## i20->h3 i21->h3   
## -0.04 -0.18   
## b->h4 i1->h4 i2->h4 i3->h4 i4->h4 i5->h4 i6->h4 i7->h4 i8->h4 i9->h4   
## -0.09 0.80 -0.54 0.65 -0.41 0.00 0.53 0.46 0.34 0.10   
## i10->h4 i11->h4 i12->h4 i13->h4 i14->h4 i15->h4 i16->h4 i17->h4 i18->h4 i19->h4   
## -0.15 0.21 -0.19 0.26 -0.15 0.24 0.24 -0.30 -0.03 0.27   
## i20->h4 i21->h4   
## -0.11 -0.67   
## b->h5 i1->h5 i2->h5 i3->h5 i4->h5 i5->h5 i6->h5 i7->h5 i8->h5 i9->h5   
## 0.21 0.78 -0.51 -0.33 0.39 0.00 -0.13 -0.21 -0.09 -0.12   
## i10->h5 i11->h5 i12->h5 i13->h5 i14->h5 i15->h5 i16->h5 i17->h5 i18->h5 i19->h5   
## 0.30 0.19 0.50 0.08 0.33 0.12 0.12 0.03 0.31 -0.48   
## i20->h5 i21->h5   
## -0.31 -0.52   
## b->o h1->o h2->o h3->o h4->o h5->o   
## 0.16 0.42 0.76 -1.15 1.53 -1.05

nnModelfullbest

## a 21-5-1 network with 116 weights  
## inputs: Age KM HP MC1 Auto1 CC Grs1 Wght G\_P Mfr\_G1 Abag\_21 AC1 Comp1 CD1 Clock1 Pw1 Radio1 SpM1 M\_Rim1 Tow\_Bar1 GreynBlue1   
## output(s): Target   
## options were - linear output units decay=0.001

library(NeuralNetTools)  
plotnet(nnModelfullbest)



nn.predsfullbest <- as.factor(predict(nnModelfullbest, testfull))  
nn.predsfullbest

## [1] 0.508454870908436 0.413527994906599 0.322599303281996   
## [4] 0.317530711627343 0.342408023677569 0.418164788340756   
## [7] 0.237976056078388 0.346091516678538 0.624073035982511   
## [10] -0.0482889130719366  
## 10 Levels: -0.0482889130719366 0.237976056078388 ... 0.624073035982511

# we worked on another model by removing the variables that were low in correlation

library(dplyr)  
mynormalization <- function(x)  
{  
 (x - min(x))/ (max(x)-min(x))  
}  
FData <- FData %>% mutate\_if(is.numeric, mynormalization)  
summary(FData)

## Target Age KM Fuel   
## Min. :0.0000 Min. :0.0000 Min. :0.0000 Length:31   
## 1st Qu.:0.2857 1st Qu.:0.3000 1st Qu.:0.2710 Class :character   
## Median :0.3367 Median :0.5000 Median :0.3880 Mode :character   
## Mean :0.4338 Mean :0.4935 Mean :0.4123   
## 3rd Qu.:0.5102 3rd Qu.:0.7000 3rd Qu.:0.5605   
## Max. :1.0000 Max. :1.0000 Max. :1.0000   
##   
## HP MC Colour Auto   
## Min. :0.0000 Min. :0.0000 Length:31 Min. :0.00000   
## 1st Qu.:0.0000 1st Qu.:0.0000 Class :character 1st Qu.:0.00000   
## Median :0.1368 Median :1.0000 Mode :character Median :0.00000   
## Mean :0.2832 Mean :0.6452 Mean :0.03226   
## 3rd Qu.:0.1368 3rd Qu.:1.0000 3rd Qu.:0.00000   
## Max. :1.0000 Max. :1.0000 Max. :1.00000   
##   
## 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 Mfr\_G ABS Abag\_1   
## Min. :0.00000 Min. :0.0000 Min. : NA Min. : NA   
## 1st Qu.:0.00000 1st Qu.:1.0000 1st Qu.: NA 1st Qu.: NA   
## Median :0.00000 Median :1.0000 Median : NA Median : NA   
## Mean :0.05645 Mean :0.9032 Mean :NaN Mean :NaN   
## 3rd Qu.:0.00000 3rd Qu.:1.0000 3rd Qu.: NA 3rd Qu.: NA   
## Max. :1.00000 Max. :1.0000 Max. : NA Max. : NA   
## NA's :31 NA's :31   
## Abag\_2 AC Comp CD   
## Min. :0.0000 Min. :0.0000 Min. :0.0000 Min. :0.0000   
## 1st Qu.:1.0000 1st Qu.:0.0000 1st Qu.:1.0000 1st Qu.:0.0000   
## Median :1.0000 Median :1.0000 Median :1.0000 Median :1.0000   
## Mean :0.9355 Mean :0.5484 Mean :0.9032 Mean :0.6129   
## 3rd Qu.:1.0000 3rd Qu.:1.0000 3rd Qu.:1.0000 3rd Qu.:1.0000   
## Max. :1.0000 Max. :1.0000 Max. :1.0000 Max. :1.0000   
##   
## Clock Pw PStr Radio   
## Min. :0.0000 Min. :0.0000 Min. : NA Min. :0.00000   
## 1st Qu.:1.0000 1st Qu.:1.0000 1st Qu.: NA 1st Qu.:0.00000   
## Median :1.0000 Median :1.0000 Median : NA Median :0.00000   
## Mean :0.9032 Mean :0.9032 Mean :NaN Mean :0.06452   
## 3rd Qu.:1.0000 3rd Qu.:1.0000 3rd Qu.: NA 3rd Qu.:0.00000   
## Max. :1.0000 Max. :1.0000 Max. : NA Max. :1.00000   
## NA's :31   
## SpM M\_Rim Tow\_Bar   
## Min. :0.0000 Min. :0.0000 Min. :0.0000   
## 1st Qu.:1.0000 1st Qu.:0.0000 1st Qu.:0.0000   
## Median :1.0000 Median :0.0000 Median :0.0000   
## Mean :0.8065 Mean :0.3871 Mean :0.1613   
## 3rd Qu.:1.0000 3rd Qu.:1.0000 3rd Qu.:0.0000   
## Max. :1.0000 Max. :1.0000 Max. :1.0000   
##

#creating a dummy variable to tackle colour  
  
GreynBlue<-ifelse(FData$Colour == 'Grey' | FData$Colour == 'Blue',1,0)  
  
  
set.seed(123)  
#remove redundant variables   
FData <- subset (FData, select = -c(4,7,10,11,16,17,18,20,23,25,24))  
FData <- FData %>% mutate(  
 Target,  
 Age,  
 KM,  
 HP,  
 CC,  
 Grs=as.factor(Grs),  
 Wght,  
 G\_P,  
 Auto=as.factor(Auto),  
 MC=as.factor(MC),  
 Mfr\_G=as.factor(Mfr\_G),  
 AC=as.factor(AC),  
 CD=as.factor(CD),  
 Clock=as.factor(Clock),  
 SpM=as.factor(SpM),  
 M\_Rim=as.factor(M\_Rim),  
 Tow\_Bar=as.factor(Tow\_Bar),  
 GreynBlue=as.factor(GreynBlue)  
)  
# divide into test and training data set  
set.seed(124)  
ind <- sample(2, nrow(FData), replace = T, prob = c(0.7, 0.3))  
train <- FData[ind == 1, ]  
test <- FData[ind == 2, ]  
str(FData)

## tibble [31 × 18] (S3: tbl\_df/tbl/data.frame)  
## $ Target : num [1:31] 0.816 0.714 0.679 0.872 0.974 ...  
## $ Age : num [1:31] 0.3 0 0.3 0.9 1 0.6 0.8 0.2 0.2 0.8 ...  
## $ KM : num [1:31] 0.375 0.585 0.387 0.229 0.421 ...  
## $ HP : num [1:31] 1 1 1 1 1 ...  
## $ MC : Factor w/ 2 levels "0","1": 1 1 1 2 2 1 2 2 1 2 ...  
## $ Auto : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ CC : num [1:31] 1 1 1 1 1 1 1 0.5 0.5 0.5 ...  
## $ Grs : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 1 1 1 1 ...  
## $ Wght : num [1:31] 1 1 1 1 1 ...  
## $ G\_P : num [1:31] 0.375 0 0 0 0 0 0 1 0 0 ...  
## $ Mfr\_G : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 1 1 2 ...  
## $ AC : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 1 2 1 ...  
## $ 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 ...  
## $ 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 ...  
## $ GreynBlue: Factor w/ 2 levels "0","1": 1 1 1 1 2 2 2 2 2 2 ...

library(nnet)  
nnModel<- nnet(Target~ ., data = train, linout = T, size = 10, decay = 0.01, maxit = 1000)

## # weights: 191  
## initial value 3.880534   
## iter 10 value 0.245082  
## iter 20 value 0.110357  
## iter 30 value 0.099268

……………………..  
## iter 220 value 0.097612  
## iter 230 value 0.097611  
## iter 240 value 0.097611  
## final value 0.097611   
## converged

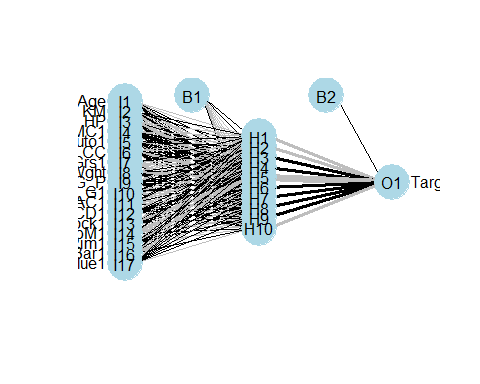
summary(nnModel)

## a 17-10-1 network with 191 weights  
## options were - linear output units decay=0.01  
## b->h1 i1->h1 i2->h1 i3->h1 i4->h1 i5->h1 i6->h1 i7->h1 i8->h1 i9->h1   
## 0.04 -0.16 0.06 -0.17 -0.01 0.00 -0.09 0.07 -0.09 0.00   
## i10->h1 i11->h1 i12->h1 i13->h1 i14->h1 i15->h1 i16->h1 i17->h1   
## 0.10 0.09 -0.07 -0.13 0.05 -0.06 0.03 -0.12   
## b->h2 i1->h2 i2->h2 i3->h2 i4->h2 i5->h2 i6->h2 i7->h2 i8->h2 i9->h2   
## 0.04 -0.16 0.06 -0.17 -0.01 0.00 -0.09 0.07 -0.09 0.00   
## i10->h2 i11->h2 i12->h2 i13->h2 i14->h2 i15->h2 i16->h2 i17->h2   
## 0.10 0.09 -0.07 -0.13 0.05 -0.06 0.03 -0.12   
## b->h3 i1->h3 i2->h3 i3->h3 i4->h3 i5->h3 i6->h3 i7->h3 i8->h3 i9->h3   
## -0.04 0.17 -0.06 0.18 0.01 0.00 0.09 -0.07 0.09 0.00   
## i10->h3 i11->h3 i12->h3 i13->h3 i14->h3 i15->h3 i16->h3 i17->h3   
## -0.11 -0.10 0.07 0.13 -0.06 0.06 -0.03 0.12   
## b->h4 i1->h4 i2->h4 i3->h4 i4->h4 i5->h4 i6->h4 i7->h4 i8->h4 i9->h4   
## -0.04 0.17 -0.06 0.18 0.01 0.00 0.09 -0.07 0.09 0.00   
## i10->h4 i11->h4 i12->h4 i13->h4 i14->h4 i15->h4 i16->h4 i17->h4   
## -0.11 -0.10 0.07 0.14 -0.06 0.06 -0.03 0.12   
## b->h5 i1->h5 i2->h5 i3->h5 i4->h5 i5->h5 i6->h5 i7->h5 i8->h5 i9->h5   
## 0.00 -0.37 0.23 -0.62 0.37 0.00 -0.38 -0.11 -0.37 0.01   
## i10->h5 i11->h5 i12->h5 i13->h5 i14->h5 i15->h5 i16->h5 i17->h5   
## 0.19 0.12 0.27 -0.45 -0.19 -0.32 0.05 0.18   
## b->h6 i1->h6 i2->h6 i3->h6 i4->h6 i5->h6 i6->h6 i7->h6 i8->h6 i9->h6   
## -0.04 0.17 -0.06 0.18 0.01 0.00 0.09 -0.07 0.09 0.00   
## i10->h6 i11->h6 i12->h6 i13->h6 i14->h6 i15->h6 i16->h6 i17->h6   
## -0.11 -0.10 0.07 0.14 -0.06 0.06 -0.03 0.12   
## b->h7 i1->h7 i2->h7 i3->h7 i4->h7 i5->h7 i6->h7 i7->h7 i8->h7 i9->h7   
## -0.04 0.17 -0.06 0.18 0.01 0.00 0.09 -0.07 0.09 0.00   
## i10->h7 i11->h7 i12->h7 i13->h7 i14->h7 i15->h7 i16->h7 i17->h7   
## -0.11 -0.10 0.07 0.13 -0.06 0.06 -0.03 0.12   
## b->h8 i1->h8 i2->h8 i3->h8 i4->h8 i5->h8 i6->h8 i7->h8 i8->h8 i9->h8   
## -0.04 0.17 -0.06 0.18 0.01 0.00 0.09 -0.07 0.09 0.00   
## i10->h8 i11->h8 i12->h8 i13->h8 i14->h8 i15->h8 i16->h8 i17->h8   
## -0.11 -0.10 0.07 0.14 -0.06 0.06 -0.03 0.12   
## b->h9 i1->h9 i2->h9 i3->h9 i4->h9 i5->h9 i6->h9 i7->h9 i8->h9 i9->h9   
## -0.04 0.17 -0.06 0.18 0.01 0.00 0.09 -0.07 0.09 0.00   
## i10->h9 i11->h9 i12->h9 i13->h9 i14->h9 i15->h9 i16->h9 i17->h9   
## -0.11 -0.10 0.07 0.14 -0.06 0.06 -0.03 0.12   
## b->h10 i1->h10 i2->h10 i3->h10 i4->h10 i5->h10 i6->h10 i7->h10   
## 0.04 -0.16 0.06 -0.17 -0.01 0.00 -0.09 0.07   
## i8->h10 i9->h10 i10->h10 i11->h10 i12->h10 i13->h10 i14->h10 i15->h10   
## -0.09 0.00 0.10 0.09 -0.07 -0.13 0.05 -0.06   
## i16->h10 i17->h10   
## 0.03 -0.12   
## b->o h1->o h2->o h3->o h4->o h5->o h6->o h7->o h8->o h9->o h10->o   
## 0.01 -0.39 -0.39 0.42 0.42 -1.12 0.42 0.42 0.42 0.42 -0.39

nnModel

## a 17-10-1 network with 191 weights  
## inputs: Age KM HP MC1 Auto1 CC Grs1 Wght G\_P Mfr\_G1 AC1 CD1 Clock1 SpM1 M\_Rim1 Tow\_Bar1 GreynBlue1   
## output(s): Target   
## options were - linear output units decay=0.01

library(NeuralNetTools)  
plotnet(nnModel)



nn.preds <- as.factor(predict(nnModel, test))  
nn.preds

## [1] 0.41516682316512 0.378916227959371 0.514198564213712   
## [4] 0.334769606437482 0.281643636484912 0.427609488087814   
## [7] 0.417854220776309 0.326364876017335 0.346225185063257   
## [10] -0.0352068040274442  
## 10 Levels: -0.0352068040274442 0.281643636484912 ... 0.514198564213712

#nested for loop for decay and size  
#library(plyr)  
#df <- ldply(output, data.frame)  
#setNames(df, c("", "Values"))

set.seed(156)  
indx <- sample(2, nrow(train), replace = T, prob = c(0.5, 0.5))  
train2 <- train[indx == 1,]  
validation <- train[indx == 2, ]  
  
err <- vector("numeric", 300)  
d <- seq(0.001, .1, length.out = 30)  
sizes <- seq(1,10, length.out = 10)  
k =1  
err\_df <- data.frame(  
 size=integer(),  
 decay=numeric(),  
 error = numeric()  
)  
  
for (s in sizes) {  
 for (i in d)  
 {  
 print(paste("K:", k, "i:", i,"size: ", s))  
 mymodel <-  
 nnet(  
 Target ~ .,  
 data = train2,  
 decay = i,  
 size = s,  
 maxit = 1000,  
 trace = F  
 )  
 pred.class <- predict(mymodel, newdata = validation)  
 err\_tmp <-  
 round((sqrt(mean((pred.class - validation$Target) ^ 2  
 )) / mean(validation$Target)) \* 100, 2)  
 err\_df[nrow(err\_df)+1,] <- c(s,i,err\_tmp)  
 print(err\_tmp)  
 err[k]<-err\_tmp  
 k <- k + 1  
 }  
}

## [1] "K: 1 i: 0.001 size: 1"  
## [1] 34.23  
## [1] "K: 2 i: 0.00441379310344828 size: 1"  
## [1] 30.03  
## [1] "K: 3 i: 0.00782758620689655 size: 1"  
## [1] 28.67  
## [1] "K: 4 i: 0.0112413793103448 size: 1"  
## [1] 28.27  
## [1] "K: 5 i: 0.0146551724137931 size: 1"  
## [1] 30.08  
## [1] "K: 6 i: 0.0180689655172414 size: 1"  
## [1] 30.64  
## [1] "K: 7 i: 0.0214827586206897 size: 1"  
## [1] 31.34  
## [1] "K: 8 i: 0.0248965517241379 size: 1"  
## [1] 32.13  
## [1] "K: 9 i: 0.0283103448275862 size: 1"  
## [1] 32.97  
## [1] "K: 10 i: 0.0317241379310345 size: 1"  
## [1] 31.25  
…and so on for 300 iterations

min(err)

## [1] 28.27

#check for the index of the minimum value and note the decay and size value  
View(err\_df)

#error for training data

errtrain <- vector("numeric", 300)  
dtrain <- seq(0.001, .1, length.out = 30)  
sizestrain <- seq(1,10, length.out = 10)  
ktrain =1  
errtrain\_df <- data.frame(  
 size=integer(),  
 decay=numeric(),  
 error = numeric()  
)  
  
for (strain in sizestrain) {  
 for (itrain in dtrain)  
 {  
 print(paste("K:", ktrain, "i:", itrain,"size: ", strain))  
 mymodel\_train <-  
 nnet(  
 Target ~ .,  
 data = train,  
 decay = itrain,  
 size = strain,  
 maxit = 1000,  
 trace = F  
 )  
 pred.classtrain <- predict(mymodel\_train, newdata = train)  
 errtrain\_tmp <-  
 round((sqrt(mean((pred.classtrain - train$Target) ^ 2  
 )) / mean(train$Target)) \* 100, 2)  
 errtrain\_df[nrow(errtrain\_df)+1,] <- c(strain,itrain,errtrain\_tmp)  
 print(errtrain\_tmp)  
 errtrain[ktrain]<-errtrain\_tmp  
 ktrain <- ktrain + 1  
 }  
}

## [1] "K: 1 i: 0.001 size: 1"  
## [1] 11.31  
## [1] "K: 2 i: 0.00441379310344828 size: 1"  
## [1] 14.5  
## [1] "K: 3 i: 0.00782758620689655 size: 1"  
## [1] 17.28  
## [1] "K: 4 i: 0.0112413793103448 size: 1"  
## [1] 19.41  
## [1] "K: 5 i: 0.0146551724137931 size: 1"  
## [1] 21.07  
## [1] "K: 6 i: 0.0180689655172414 size: 1"  
## [1] 22.32  
## [1] "K: 7 i: 0.0214827586206897 size: 1"  
## [1] 23.13  
## [1] "K: 8 i: 0.0248965517241379 size: 1"  
## [1] 24.43  
## [1] "K: 9 i: 0.0283103448275862 size: 1"  
## [1] 25.39  
## [1] "K: 10 i: 0.0317241379310345 size: 1"  
## [1] 26.31  
…and so on for 300 iterations

min(errtrain)

## [1] 5.09

View(errtrain\_df)

#error for test data

errtest <- vector("numeric", 300)  
dtest <- seq(0.001, .1, length.out = 30)  
sizestest <- seq(1,10, length.out = 10)  
ktest =1  
errtest\_df <- data.frame(  
 size=integer(),  
 decay=numeric(),  
 error = numeric()  
)  
  
for (stest in sizestest) {  
 for (itest in dtest)  
 {  
 print(paste("K:", ktest, "i:", itest,"size: ", stest))  
 mymodel\_test <-  
 nnet(  
 Target ~ .,  
 data = test,  
 decay = itest,  
 size = stest,  
 maxit = 1000,  
 trace = F  
 )  
 pred.classtest <- predict(mymodel\_test, newdata = test)  
 errtest\_tmp <-  
 round((sqrt(mean((pred.classtest - test$Target) ^ 2  
 )) / mean(test$Target)) \* 100, 2)  
 errtest\_df[nrow(errtest\_df)+1,] <- c(stest,itest,errtest\_tmp)  
 print(errtest\_tmp)  
 errtest[ktest]<-errtest\_tmp  
 ktest <- ktest + 1  
 }  
}

## [1] "K: 1 i: 0.001 size: 1"  
## [1] 5.7  
## [1] "K: 2 i: 0.00441379310344828 size: 1"  
## [1] 11.37  
## [1] "K: 3 i: 0.00782758620689655 size: 1"  
## [1] 15.54  
## [1] "K: 4 i: 0.0112413793103448 size: 1"  
## [1] 18.69  
## [1] "K: 5 i: 0.0146551724137931 size: 1"  
## [1] 21.21  
## [1] "K: 6 i: 0.0180689655172414 size: 1"  
## [1] 23.22  
## [1] "K: 7 i: 0.0214827586206897 size: 1"  
## [1] 24.75  
## [1] "K: 8 i: 0.0248965517241379 size: 1"  
## [1] 25.79  
## [1] "K: 9 i: 0.0283103448275862 size: 1"  
## [1] 26.42  
## [1] "K: 10 i: 0.0317241379310345 size: 1"  
## [1] 26.8  
….and so on for 300 iterations

testerr<-min(errtest)  
testerr

## [1] 3.3

View(errtest\_df)

library(nnet)  
nnModelbest<- nnet(Target~ ., data = train, linout = T, size = 9, decay = 0.00100, maxit = 1000)

## # weights: 172  
## initial value 41.895806   
## iter 10 value 0.112112  
## iter 20 value 0.051770  
## iter 30 value 0.038879  
## iter 40 value 0.030584  
## iter 50 value 0.024596  
## iter 60 value 0.021530  
## iter 70 value 0.020451  
## iter 80 value 0.019900  
## iter 90 value 0.019387

…………………………………  
## iter 430 value 0.016957  
## iter 440 value 0.016957  
## final value 0.016957   
## converged

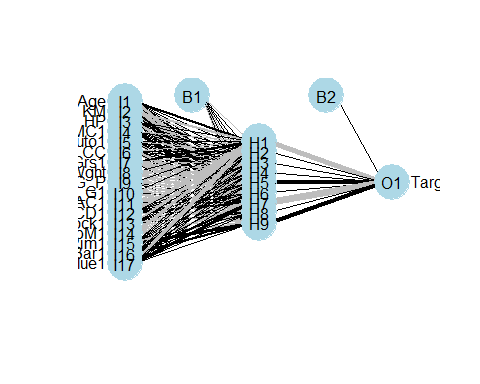
summary(nnModelbest)

## a 17-9-1 network with 172 weights  
## options were - linear output units decay=0.001  
## b->h1 i1->h1 i2->h1 i3->h1 i4->h1 i5->h1 i6->h1 i7->h1 i8->h1 i9->h1   
## -0.05 0.31 -0.45 0.11 -0.16 0.00 -0.07 0.28 0.07 0.18   
## i10->h1 i11->h1 i12->h1 i13->h1 i14->h1 i15->h1 i16->h1 i17->h1   
## 0.10 -0.02 -0.71 -0.56 -0.53 0.09 -0.09 -1.00   
## b->h2 i1->h2 i2->h2 i3->h2 i4->h2 i5->h2 i6->h2 i7->h2 i8->h2 i9->h2   
## 0.01 0.04 -0.08 0.02 0.02 0.00 0.05 -0.05 0.00 0.00   
## i10->h2 i11->h2 i12->h2 i13->h2 i14->h2 i15->h2 i16->h2 i17->h2   
## -0.04 -0.09 0.05 0.04 -0.02 0.05 0.02 0.05   
## b->h3 i1->h3 i2->h3 i3->h3 i4->h3 i5->h3 i6->h3 i7->h3 i8->h3 i9->h3   
## 0.01 0.04 -0.08 0.02 0.02 0.00 0.05 -0.05 0.00 -0.01   
## i10->h3 i11->h3 i12->h3 i13->h3 i14->h3 i15->h3 i16->h3 i17->h3   
## -0.04 -0.09 0.05 0.04 -0.02 0.05 0.02 0.05   
## b->h4 i1->h4 i2->h4 i3->h4 i4->h4 i5->h4 i6->h4 i7->h4 i8->h4 i9->h4   
## 0.01 0.04 -0.08 0.02 0.02 0.00 0.05 -0.05 0.00 0.00   
## i10->h4 i11->h4 i12->h4 i13->h4 i14->h4 i15->h4 i16->h4 i17->h4   
## -0.04 -0.09 0.05 0.04 -0.02 0.05 0.02 0.05   
## b->h5 i1->h5 i2->h5 i3->h5 i4->h5 i5->h5 i6->h5 i7->h5 i8->h5 i9->h5   
## -0.06 0.34 -0.47 -0.04 0.11 0.00 0.12 -0.28 -0.11 -0.03   
## i10->h5 i11->h5 i12->h5 i13->h5 i14->h5 i15->h5 i16->h5 i17->h5   
## -0.23 -0.40 0.25 0.09 -0.16 0.11 0.08 0.11   
## b->h6 i1->h6 i2->h6 i3->h6 i4->h6 i5->h6 i6->h6 i7->h6 i8->h6 i9->h6   
## 0.01 0.04 -0.08 0.02 0.02 0.00 0.05 -0.05 0.00 0.00   
## i10->h6 i11->h6 i12->h6 i13->h6 i14->h6 i15->h6 i16->h6 i17->h6   
## -0.04 -0.09 0.05 0.04 -0.02 0.05 0.02 0.05   
## b->h7 i1->h7 i2->h7 i3->h7 i4->h7 i5->h7 i6->h7 i7->h7 i8->h7 i9->h7   
## -0.07 -1.06 0.73 -0.62 0.39 0.00 -0.87 -0.29 -0.34 0.12   
## i10->h7 i11->h7 i12->h7 i13->h7 i14->h7 i15->h7 i16->h7 i17->h7   
## 0.25 0.09 0.31 -0.31 -0.14 -0.30 0.48 0.71   
## b->h8 i1->h8 i2->h8 i3->h8 i4->h8 i5->h8 i6->h8 i7->h8 i8->h8 i9->h8   
## 0.01 0.04 -0.08 0.02 0.02 0.00 0.05 -0.05 0.00 0.00   
## i10->h8 i11->h8 i12->h8 i13->h8 i14->h8 i15->h8 i16->h8 i17->h8   
## -0.04 -0.09 0.05 0.04 -0.02 0.05 0.02 0.05   
## b->h9 i1->h9 i2->h9 i3->h9 i4->h9 i5->h9 i6->h9 i7->h9 i8->h9 i9->h9   
## -0.11 -0.50 0.34 0.17 -0.22 0.00 0.13 0.00 -0.01 0.04   
## i10->h9 i11->h9 i12->h9 i13->h9 i14->h9 i15->h9 i16->h9 i17->h9   
## -0.32 -0.63 -0.11 -0.05 -0.34 0.55 0.27 0.13   
## b->o h1->o h2->o h3->o h4->o h5->o h6->o h7->o h8->o h9->o   
## 0.14 -1.23 0.22 0.22 0.22 0.85 0.22 -1.57 0.22 0.96

nnModelbest

## a 17-9-1 network with 172 weights  
## inputs: Age KM HP MC1 Auto1 CC Grs1 Wght G\_P Mfr\_G1 AC1 CD1 Clock1 SpM1 M\_Rim1 Tow\_Bar1 GreynBlue1   
## output(s): Target   
## options were - linear output units decay=0.001

library(NeuralNetTools)  
plotnet(nnModelbest)



nn.predsbest <- as.factor(predict(nnModelbest, test))  
nn.predsbest

## [1] 0.484905947221432 0.298792630902601 0.44449645721807   
## [4] 0.39829659646864 0.405425758882406 0.598870728849939   
## [7] 0.368428042587192 0.344600789840269 0.561268712106879   
## [10] -0.0347688946819508  
## 10 Levels: -0.0347688946819508 0.298792630902601 ... 0.598870728849939