1. Removing the Employee count, employee number , Over18,Standardhours columns from the sheet.

**hrData\_NN <- read.table("HR\_Employee\_Attrition\_Data.csv", sep = ",", header = T)**

**hrData\_NN <- hrData\_NN[-c(9,10,22,27)]**

1. Converting the categorial string values to numeric discrete values

**hrdata\_subset=hrData\_NN[-c(2)]**

**hrData\_subset1=sapply(hrdata\_subset,unclass)**

**hrData\_subset1\_scaled=scale(hrData\_subset1)**

**hrData\_NN\_scaled=cbind(hrData\_subset1\_scaled,hrData\_NN[2])**

1. Revalue the Attrition to 1 and 0 s

**library(plyr)**

**hrData\_NN\_scaled$Attrition <- revalue(hrData\_NN\_scaled$Attrition, c("Yes"=1))**

**hrData\_NN\_scaled$Attrition <- revalue(hrData\_NN\_scaled$Attrition, c("No"=0))**

1. Creating the training and test data

**s <- sample(c(1:2940), size = 2058)**

**hrData.Train=hrData\_NN\_scaled[s,]**

**hrData.Test=hrData\_NN\_scaled[-s,]**

1. Converting the attrition to numeric datatype as neural net needs all the columns to be in numeric datatype.

**hrData.Train$Attrition=as.numeric(as.character(hrData.Train$Attrition))**

1. Set the seed and executing the neuralnet

As there are 35 variables the hidden neurons will be continuous square roots i.e

Sqrt(35) ~ 5 -> Sqrt(6)->2

So there will be two hidden layers with 5 and 2 counts

**set.seed(1000)**

**nn1 <- neuralnet(formula = Attrition ~ Age+BusinessTravel+DailyRate+Department+DistanceFromHome+Education+EducationField+EnvironmentSatisfaction+Gender+HourlyRate+JobInvolvement+JobLevel+JobRole+JobSatisfaction+MaritalStatus+MonthlyIncome+MonthlyRate+NumCompaniesWorked+OverTime+PercentSalaryHike+PerformanceRating+RelationshipSatisfaction+StockOptionLevel+TotalWorkingYears+TrainingTimesLastYear+WorkLifeBalance+YearsAtCompany+YearsInCurrentRole+YearsSinceLastPromotion+YearsWithCurrManager ,**

**data = hrData.Train,**

**hidden = c(5,2),**

**err.fct = "sse",**

**linear.output = FALSE,**

**lifesign = "full",**

**lifesign.step = 10,**

**threshold = 0.01,**

**stepmax = 2000**

**)**

Output:

hidden: 5, 2 thresh: 0.01 rep: 1/1 steps: 10 min thresh: 4.488238804

20 min thresh: 4.488238804

30 min thresh: 2.651672294

40 min thresh: 1.521972276

50 min thresh: 0.9496091202

60 min thresh: 0.677883468

70 min thresh: 0.677883468

80 min thresh: 0.677883468

90 min thresh: 0.677883468

100 min thresh: 0.6245093006

110 min thresh: 0.6245093006

120 min thresh: 0.4461118859

130 min thresh: 0.4287551997

140 min thresh: 0.4287551997

150 min thresh: 0.4287551997

160 min thresh: 0.4287551997

170 min thresh: 0.4287551997

180 min thresh: 0.3613334845

190 min thresh: 0.3079583258

200 min thresh: 0.2374665189

210 min thresh: 0.2071414236

220 min thresh: 0.1475605448

230 min thresh: 0.1213096964

240 min thresh: 0.1213096964

250 min thresh: 0.1213096964

260 min thresh: 0.1213096964

270 min thresh: 0.1213096964

280 min thresh: 0.1213096964

290 min thresh: 0.1213096964

300 min thresh: 0.1213096964

310 min thresh: 0.1213096964

320 min thresh: 0.1213096964

330 min thresh: 0.1213096964

340 min thresh: 0.1131631471

350 min thresh: 0.08149687237

360 min thresh: 0.07841850364

370 min thresh: 0.07841850364

380 min thresh: 0.07841850364

390 min thresh: 0.07841850364

400 min thresh: 0.07841850364

410 min thresh: 0.07841850364

420 min thresh: 0.07841850364

430 min thresh: 0.07841850364

440 min thresh: 0.07841850364

450 min thresh: 0.04817119333

460 min thresh: 0.0460251747

470 min thresh: 0.03259233814

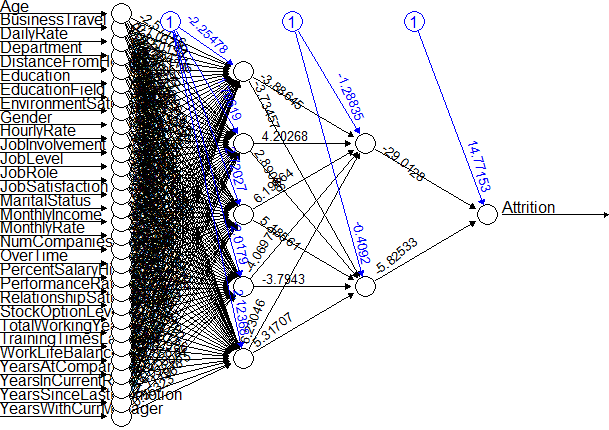
480 min thresh: 0.03259233814

490 min thresh: 0.02370930571

500 min thresh: 0.01733211733

510 min thresh: 0.01566712465

514 error: 46.51901 time: 2.67 secs



1. The error is :

**sum((hrData.Train$Attrition - hrData.Train$Prob)^2) / 2**

i.e [1] 46.51901191

and it is the same value that has been showed in neural run.

1. Model evaluations:
2. We are evaluating the model on KS,AUC and GINI

**library(ROCR)**

**pred <- ROCR::prediction(hrData.Train$Prob,hrData.Train$Attrition)**

**perf <- performance(pred, "tpr", "fpr")**

**KS <- max(attr(perf, 'y.values')[[1]]-attr(perf, 'x.values')[[1]])**

**auc <- performance(pred,"auc");**

**auc <- as.numeric(auc@y.values)**

**gini = 2 \* auc – 1**

output:

auc

[1] 0.8705260696

> KS

[1] 0.7440302839

> gini

[1] 0.7410521392

1. Probability threshold value:
2. **table(hrData.Train$Attrition**)🡪 this shows a percentage of 83.72

So the probability as 83.72 quantile should be found out

**quantile(hrData.Train$Prob, c(0,1,5,10,25,50,75,83.72,90,95,98,99,100)/100)**

output :

0% 1% 5% 10% 25%

0.000000001928396032 0.000000001928402226 0.000000001928822128 0.000000001930033752 0.000000001943511791

50% 75% 83.72% 90% 95%

0.000000002488822128 0.000000826341947582 0.003437439395958249 0.988095366743728221 0.999978456199586141

98% 99% 100%

0.999999384499165478 0.999999507556603939 0.999999599283574403

So the value is 0.0034

1. Creating the class column with this threshold

**hrData.Train$Class = ifelse(hrData.Train$Prob>0.0034,1,0)**

1. **table(hrData.Train$Attrition,hrData.Train$Class)**

output:

0 1

0 1637 86

1 84 251

The accuracy is 0.9173955296

# Building neural network on Test data:

1. Computing the values for test data using the neural network which we have built for the training data

**compute.output=compute(nn1,hrData.Test\_prediction1)**

1. Creating the score column in the test data

**hrData.Test$Predict.score = compute.output$net.result**

1. Evaluating the Model :
2. **library(ROCR)**
3. **pred <- ROCR::prediction(hrData.Test$Predict.score,hrData.Test$Attrition)**
4. **perf <- performance(pred, "tpr", "fpr")**
5. **KS <- max(attr(perf, 'y.values')[[1]]-attr(perf, 'x.values')[[1]])**
6. **auc <- performance(pred,"auc");**
7. **auc <- as.numeric(auc@y.values)**
8. **gini = 2 \* auc – 1**

Output:

auc

[1] 0.8705260696

> KS

[1] 0.7440302839

> gini

[1] 0.7410521392

The model evaluation parameters are really overfitting and model seems to be convincing.

1. Creating a threshold for the quantile

**table(hrData.Test$Attrition)**

**743/(743+139)**

quantile(hrData.Train$Prob, c(0,1,5,10,25,50,75,84.24,90,95,98,99,100)/100)

0% 1% 5% 10% 25%

0.000000001928396032 0.000000001928402226 0.000000001928822128 0.000000001930033752 0.000000001943511791

50% 75% 84.24% 90% 95%

0.000000002488822128 0.000000826341947582 0.005060681167562523 0.988095366743728221 0.999978456199586141

98% 99% 100%

0.999999384499165478 0.999999507556603939 0.999999599283574403

1. **hrData.Test$Class = ifelse(hrData.Test$Predict.score>0.005,1,0)**

Adding a class column in the dataset

1. Confusion matrix for accuracy on the test data:
2. (conf.matrix[1]+conf.matrix[4])/882

[1] 0.887755102