**“Assignment - Artificial Neural Networks”**

**Homework 7**

**Stat 6620 Section - 1**

**Asati Surabhi**

# Question 1: Perform the ANN analysis on the concrete data. Produce a report explaining the data, the analysis, and the findings.

# ANN analysis on the concrete data

##### Part 1: Neural Networks -------------------  
## Example: Modeling the Strength of Concrete ----  
  
## Step 1: Collecting the data ----  
# read in data and examine structure  
concrete <- read.csv("concrete.csv")  
str(concrete)

## 'data.frame': 1030 obs. of 9 variables:  
## $ cement : num 141 169 250 266 155 ...  
## $ slag : num 212 42.2 0 114 183.4 ...  
## $ ash : num 0 124.3 95.7 0 0 ...  
## $ water : num 204 158 187 228 193 ...  
## $ superplastic: num 0 10.8 5.5 0 9.1 0 0 6.4 0 9 ...  
## $ coarseagg : num 972 1081 957 932 1047 ...  
## $ fineagg : num 748 796 861 670 697 ...  
## $ age : int 28 14 28 28 28 90 7 56 28 28 ...  
## $ strength : num 29.9 23.5 29.2 45.9 18.3 ...

## Step 2: Exploring and Preparing the data —-

# The nine variables in the data frame correspond to the eight features and one outcome.

# Neural networks work best when the input data are scaled to a narrow range around zero, and here we see values ranging anywhere from zero up to over a thousand. Therefore, we would normalize the data.

#Examine the ranges  
summary(concrete)

## cement slag ash water   
## Min. :102.0 Min. : 0.0 Min. : 0.00 Min. :121.8   
## 1st Qu.:192.4 1st Qu.: 0.0 1st Qu.: 0.00 1st Qu.:164.9   
## Median :272.9 Median : 22.0 Median : 0.00 Median :185.0   
## Mean :281.2 Mean : 73.9 Mean : 54.19 Mean :181.6   
## 3rd Qu.:350.0 3rd Qu.:142.9 3rd Qu.:118.30 3rd Qu.:192.0   
## Max. :540.0 Max. :359.4 Max. :200.10 Max. :247.0   
## superplastic coarseagg fineagg age   
## Min. : 0.000 Min. : 801.0 Min. :594.0 Min. : 1.00   
## 1st Qu.: 0.000 1st Qu.: 932.0 1st Qu.:731.0 1st Qu.: 7.00   
## Median : 6.400 Median : 968.0 Median :779.5 Median : 28.00   
## Mean : 6.205 Mean : 972.9 Mean :773.6 Mean : 45.66   
## 3rd Qu.:10.200 3rd Qu.:1029.4 3rd Qu.:824.0 3rd Qu.: 56.00   
## Max. :32.200 Max. :1145.0 Max. :992.6 Max. :365.00   
## strength   
## Min. : 2.33   
## 1st Qu.:23.71   
## Median :34.45   
## Mean :35.82   
## 3rd Qu.:46.13   
## Max. :82.60

# custom normalization function  
normalize <- function(x) {   
 return((x - min(x)) / (max(x) - min(x)))  
}  
  
# apply normalization to entire data frame  
concrete\_norm <- as.data.frame(lapply(concrete, normalize))  
  
# confirm that the range is now between zero and one  
summary(concrete\_norm$strength)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.0000 0.2664 0.4001 0.4172 0.5457 1.0000

# compared to the original minimum and maximum  
summary(concrete$strength)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 2.33 23.71 34.45 35.82 46.13 82.60

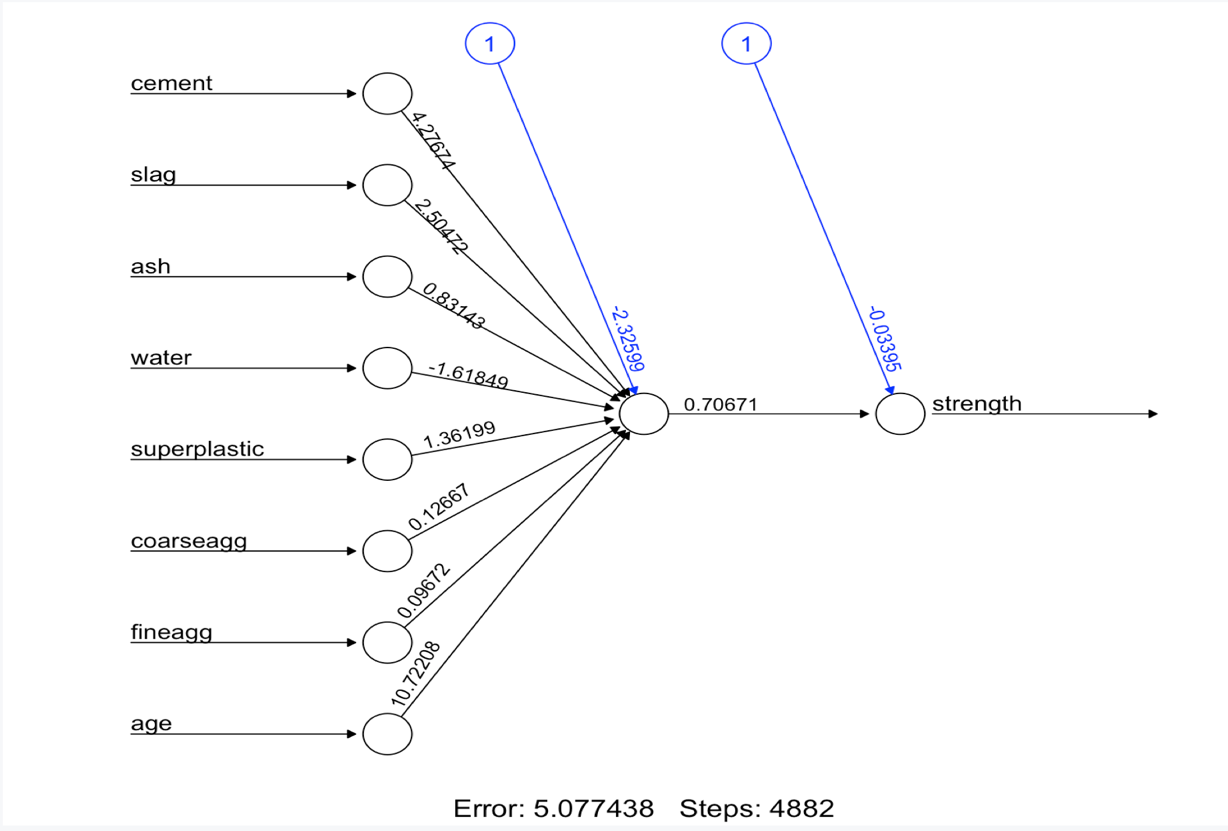
# create training and test data  
concrete\_train <- concrete\_norm[1:773, ]  
concrete\_test <- concrete\_norm[774:1030, ]

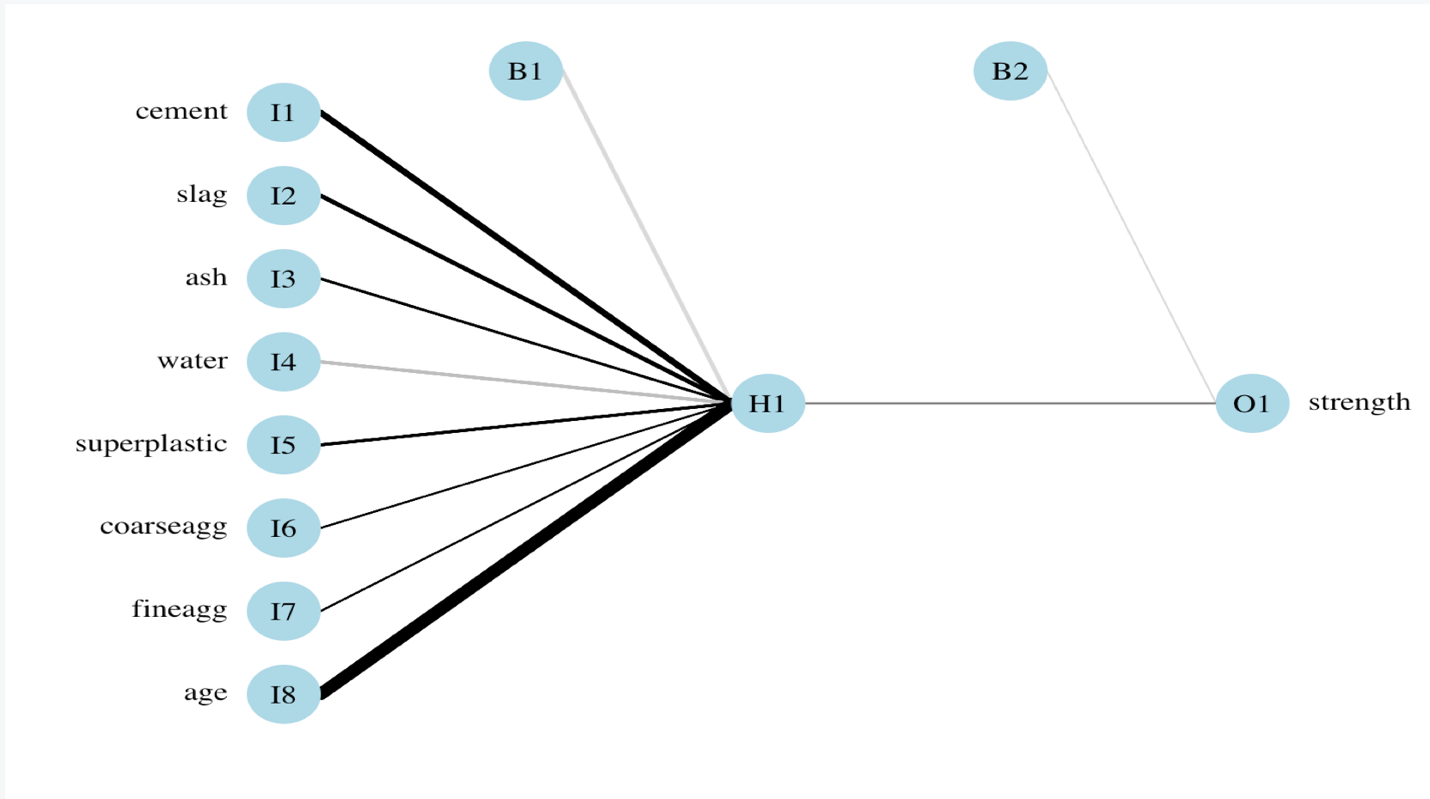
## 

## Step 3: Training a model on the data —-

To model the relationship between the ingredients used in concrete and the strength of the finished product, we are using a multilayer feedforward neural network

# train the neuralnet model  
library(neuralnet)  
  
# simple ANN with only a single hidden neuron  
set.seed(12345) # to guarantee repeatable results  
concrete\_model <- neuralnet(formula = strength ~ cement + slag +  
 ash + water + superplastic +   
 coarseagg + fineagg + age,  
 data = concrete\_train)  
  
# visualize the network topology  
plot(concrete\_model)  
  
  
library(NeuralNetTools)  
  
# plotnet  
par(mar = numeric(4), family = 'serif')  
plotnet(concrete\_model, alpha = 0.6)





# number of training steps = 4882

# Sum of Squared Errors (SSE) = 5.077

# A lower SSE implies better model performance on training data. 5.077 is clearly very high.

**## Step 4: Evaluating model performance ----**# obtain model results  
  
model\_results <- compute(concrete\_model, concrete\_test[1:8])  
# obtain predicted strength values  
predicted\_strength <- model\_results$net.result  
# examine the correlation between predicted and actual values  
cor(predicted\_strength, concrete\_test$strength)

## [,1]  
## [1,] 0.8064655576

# produce actual predictions by   
  
head(predicted\_strength)

## [,1]  
## 774 0.3258991537  
## 775 0.4677425372  
## 776 0.2370268181  
## 777 0.6718811029  
## 778 0.4663428766  
## 779 0.4685272270

concrete\_train\_original\_strength <- concrete[1:773,"strength"]  
  
strength\_min <- min(concrete\_train\_original\_strength)  
strength\_max <- max(concrete\_train\_original\_strength)  
  
head(concrete\_train\_original\_strength)

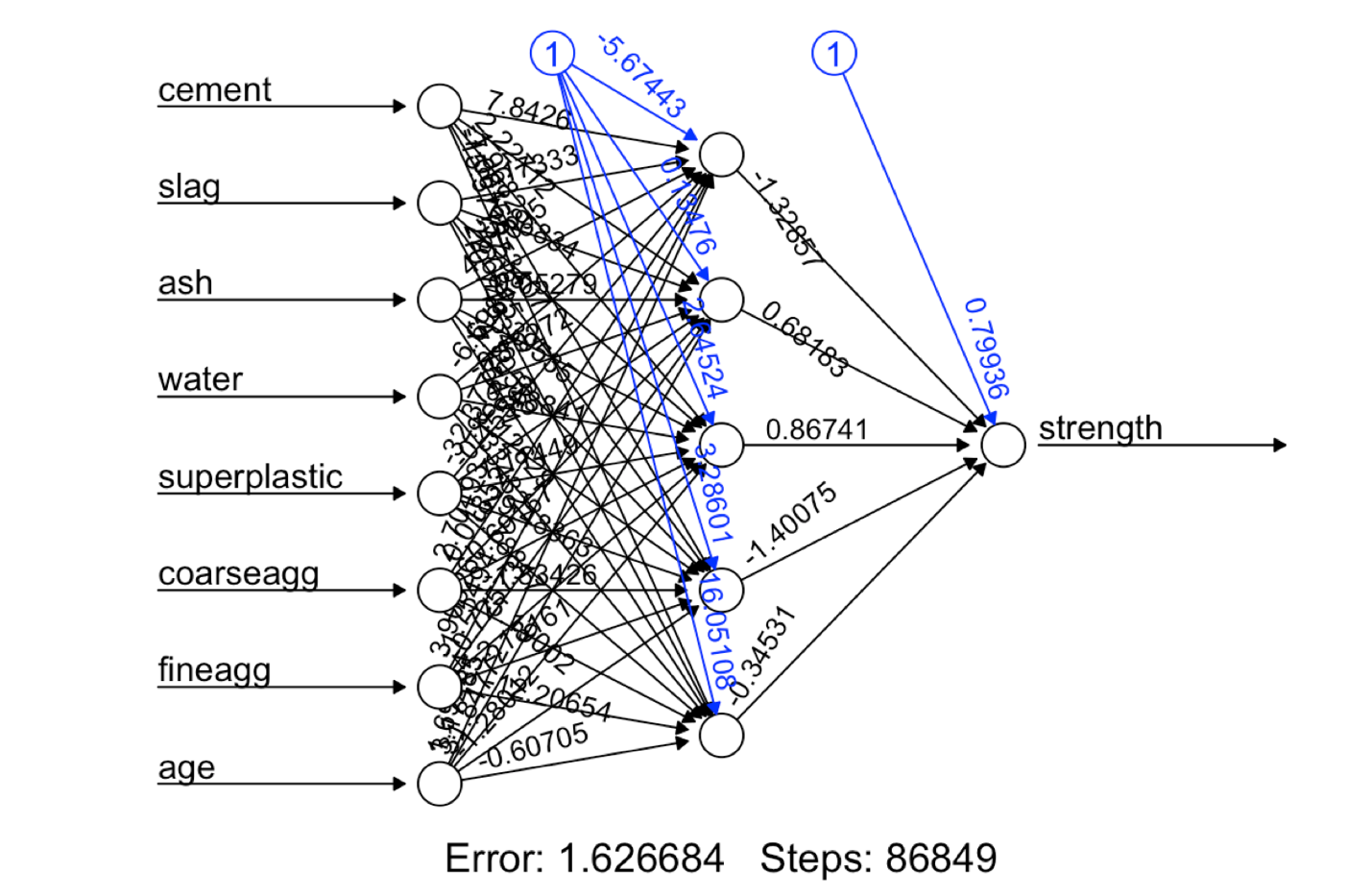
## [1] 29.89 23.51 29.22 45.85 18.29 21.86

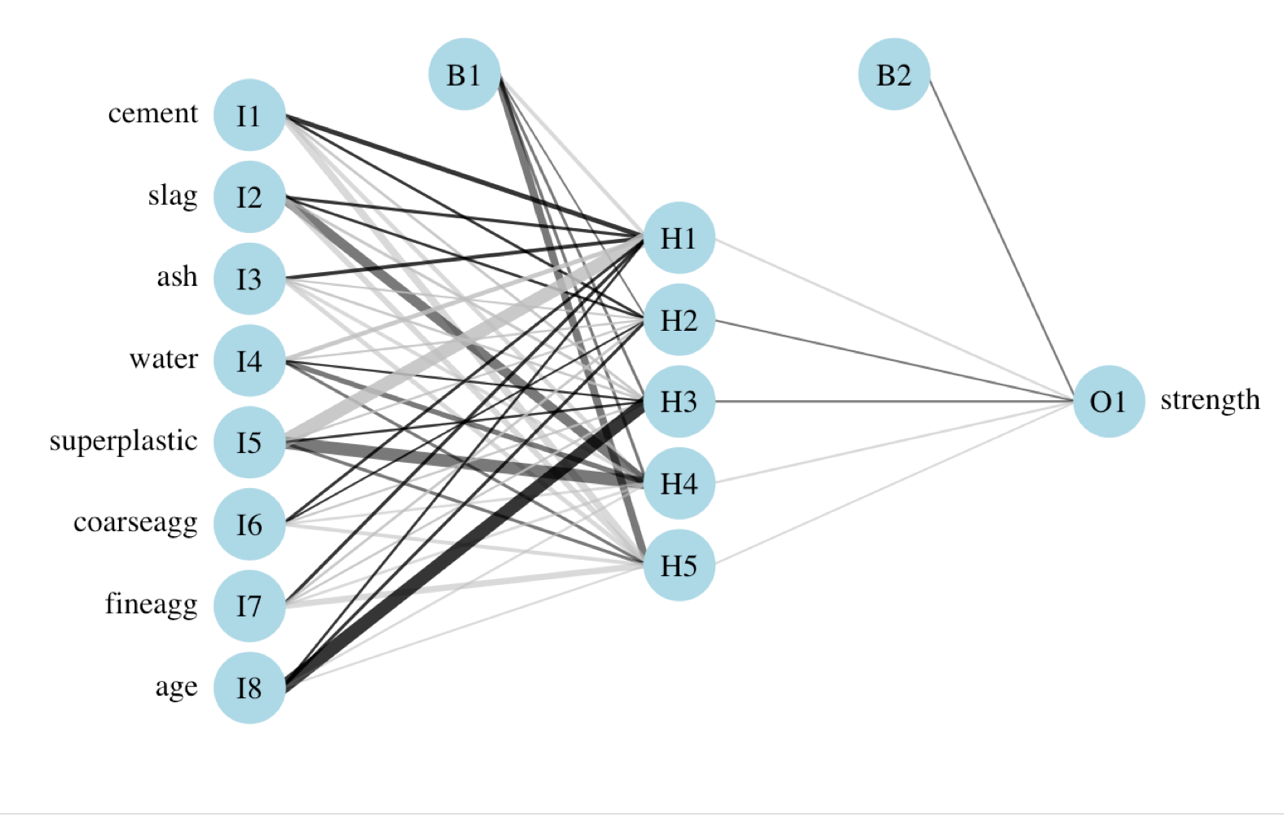
# custom normalization function  
unnormalize <- function(x, min, max) {   
 return( (max - min)\*x + min )  
}  
  
strength\_pred <- unnormalize(predicted\_strength, strength\_min, strength\_max)  
  
summary(strength\_pred)

## V1   
## Min. : 6.938092   
## 1st Qu.:24.183085   
## Median :39.366951   
## Mean :36.286375   
## 3rd Qu.:47.763397   
## Max. :55.759561

# Correlation = 0.806

**## Step 5: Improving model performance ----**# a more complex neural network topology with 5 hidden neurons  
set.seed(12345) # to guarantee repeatable results  
concrete\_model2 <- neuralnet(strength ~ cement + slag +  
 ash + water + superplastic +   
 coarseagg + fineagg + age,  
 data = concrete\_train, hidden = 5, act.fct = "logistic")  
  
# plot the network  
plot(concrete\_model2)  
  
# plotnet  
par(mar = numeric(4), family = 'serif')  
plotnet(concrete\_model2, alpha = 0.6)



  
# evaluate the results as we did before  
model\_results2 <- compute(concrete\_model2, concrete\_test[1:8])  
predicted\_strength2 <- model\_results2$net.result  
cor(predicted\_strength2, concrete\_test$strength)

## [,1]  
## [1,] 0.9244533426

# number of training steps = 86849

# Sum of Squared Errors (SSE) = 1.63

# The error has been reduced from 5.077 to 1.63 which indicates better predicted performance.

# Also, the number of training stps has rose from 4,882 to 86,849 that shows the model has become more complex i.e. it takes more iterations to find the optimal weights.

# Correlation = 0.924, higher than 0.806 (model with single hidden layer)

# Trying different activation function to improve accuracy

# a more complex neural network topology with 5 hidden neurons  
set.seed(12345) # to guarantee repeatable results  
concrete\_model2 <- neuralnet(strength ~ cement + slag +  
 ash + water + superplastic +   
 coarseagg + fineagg + age,  
 data = concrete\_train, hidden = 5, act.fct = "tanh")  
  
# evaluate the results as we did before  
model\_results2 <- compute(concrete\_model2, concrete\_test[1:8])  
predicted\_strength2 <- model\_results2$net.result  
cor(predicted\_strength2, concrete\_test$strength)

## [,1]  
## [1,] 0.5741729322

# Correlation = 0.574. dropped from model with activation function = Logistic.

# Question 2: Develop an ANN for the redwines.csv data from Homework 5. Produce a report using an Rnotebook explaining the data, the analysis, and the findings.

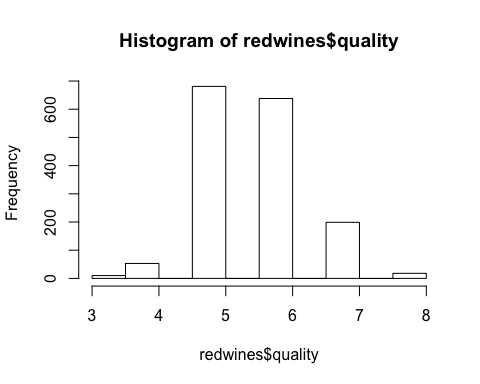
**## Step 1: Collecting the data ----**# read in data and examine structure  
redwines <- read.csv("redwines1.csv")  
str(redwines)

## 'data.frame': 1599 obs. of 12 variables:  
## $ fixed.acidity : num 6.5 9.1 6.9 7.3 12.5 5.4 10.4 7.9 7.3 9.5 ...  
## $ volatile.acidity : num 0.9 0.22 0.52 0.59 0.28 0.74 0.28 0.4 0.39 0.37 ...  
## $ citric.acid : num 0 0.24 0.25 0.26 0.54 0.09 0.54 0.3 0.31 0.52 ...  
## $ residual.sugar : num 1.6 2.1 2.6 2 2.3 1.7 2.7 1.8 2.4 2 ...  
## $ chlorides : num 0.052 0.078 0.081 0.08 0.082 0.089 0.105 0.157 0.074 0.088 ...  
## $ free.sulfur.dioxide : num 9 1 10 17 12 16 5 2 9 12 ...  
## $ total.sulfur.dioxide: num 17 28 37 104 29 26 19 45 46 51 ...  
## $ density : num 0.995 0.999 0.997 0.996 1 ...  
## $ pH : num 3.5 3.41 3.46 3.28 3.11 3.67 3.25 3.31 3.41 3.29 ...  
## $ sulphates : num 0.63 0.87 0.5 0.52 1.36 0.56 0.63 0.91 0.54 0.58 ...  
## $ alcohol : num 10.9 10.3 11 9.9 9.8 11.6 9.5 9.5 9.4 11.1 ...  
## $ quality : int 6 6 5 5 7 6 5 6 6 6 ...

**## Step 2: Exploring and Preparing the data ----**#Examine the ranges  
summary(redwines)

## fixed.acidity volatile.acidity citric.acid   
## Min. : 4.600000 Min. :0.1200000 Min. :0.0000000   
## 1st Qu.: 7.100000 1st Qu.:0.3900000 1st Qu.:0.0900000   
## Median : 7.900000 Median :0.5200000 Median :0.2600000   
## Mean : 8.319637 Mean :0.5278205 Mean :0.2709756   
## 3rd Qu.: 9.200000 3rd Qu.:0.6400000 3rd Qu.:0.4200000   
## Max. :15.900000 Max. :1.5800000 Max. :1.0000000   
## residual.sugar chlorides free.sulfur.dioxide  
## Min. : 0.900000 Min. :0.01200000 Min. : 1.00000   
## 1st Qu.: 1.900000 1st Qu.:0.07000000 1st Qu.: 7.00000   
## Median : 2.200000 Median :0.07900000 Median :14.00000   
## Mean : 2.538806 Mean :0.08746654 Mean :15.87492   
## 3rd Qu.: 2.600000 3rd Qu.:0.09000000 3rd Qu.:21.00000   
## Max. :15.500000 Max. :0.61100000 Max. :72.00000   
## total.sulfur.dioxide density pH   
## Min. : 6.00000 Min. :0.9900700 Min. :2.740000   
## 1st Qu.: 22.00000 1st Qu.:0.9956000 1st Qu.:3.210000   
## Median : 38.00000 Median :0.9967500 Median :3.310000   
## Mean : 46.46779 Mean :0.9967467 Mean :3.311113   
## 3rd Qu.: 62.00000 3rd Qu.:0.9978350 3rd Qu.:3.400000   
## Max. :289.00000 Max. :1.0036900 Max. :4.010000   
## sulphates alcohol quality   
## Min. :0.3300000 Min. : 8.40000 Min. :3.000000   
## 1st Qu.:0.5500000 1st Qu.: 9.50000 1st Qu.:5.000000   
## Median :0.6200000 Median :10.20000 Median :6.000000   
## Mean :0.6581488 Mean :10.42298 Mean :5.636023   
## 3rd Qu.:0.7300000 3rd Qu.:11.10000 3rd Qu.:6.000000   
## Max. :2.0000000 Max. :14.90000 Max. :8.000000

# the distribution of quality ratings  
hist(redwines$quality)



# custom normalization function  
normalize <- function(x) {   
 return((x - min(x)) / (max(x) - min(x)))  
}  
  
# apply normalization to entire data frame  
redwines\_norm <- as.data.frame(lapply(redwines, normalize))  
  
# confirm that the range is now between zero and one  
summary(redwines\_norm$quality)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.0000000 0.4000000 0.6000000 0.5272045 0.6000000 1.0000000

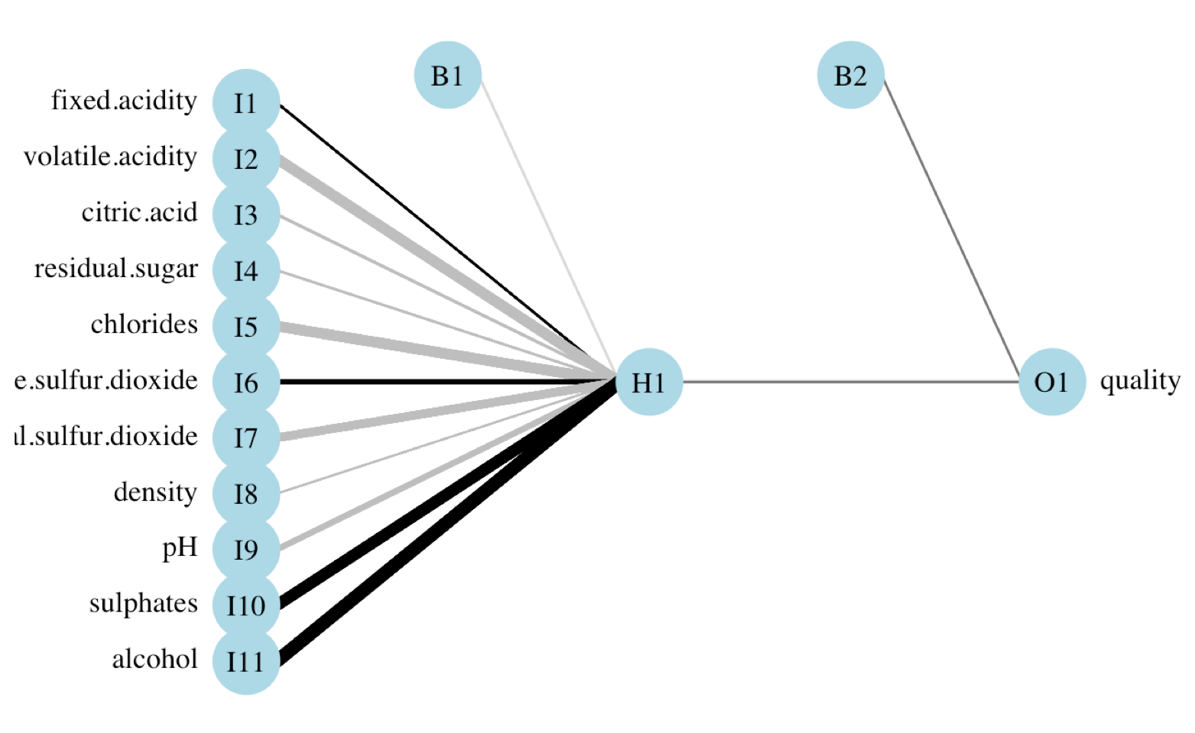
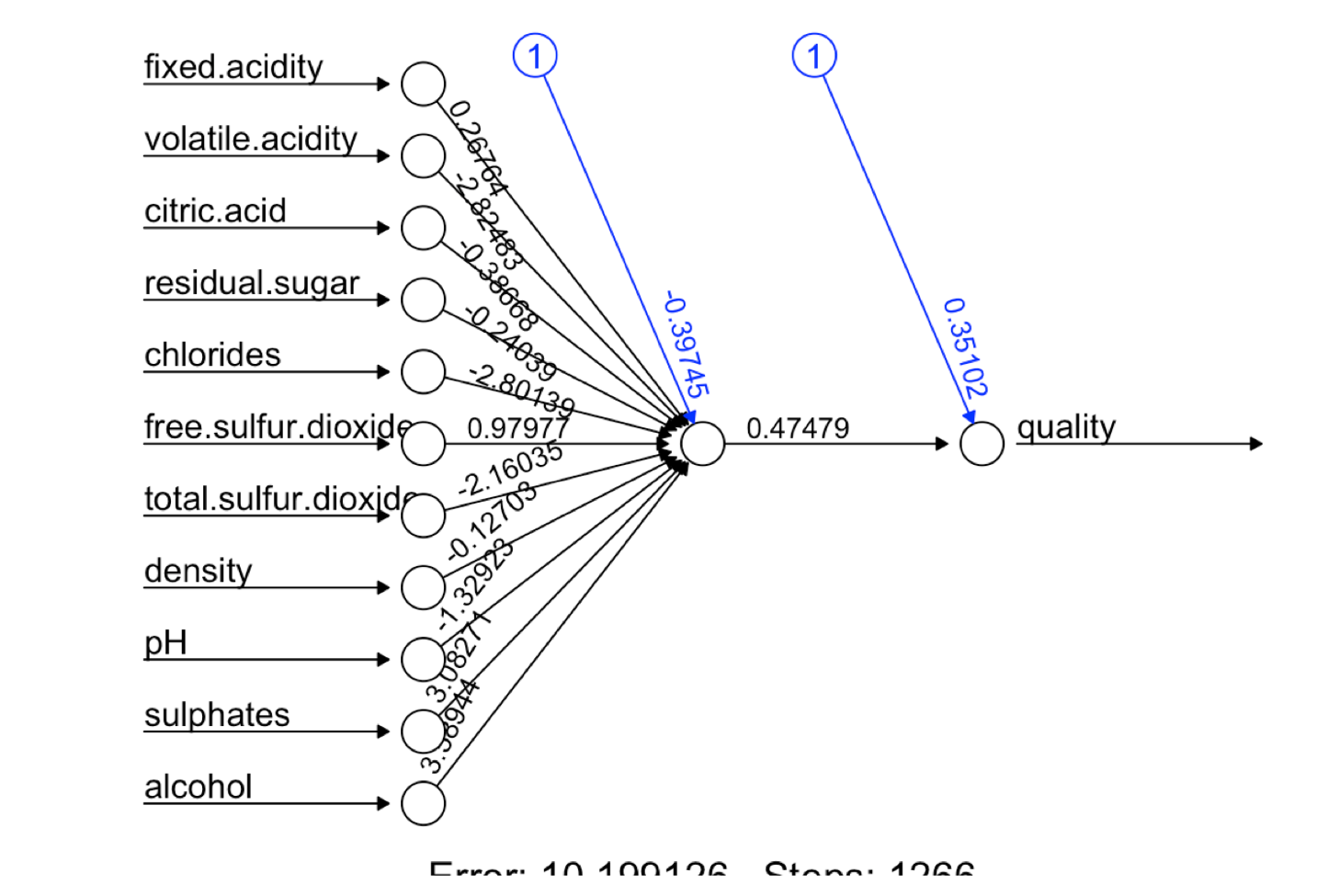
# compared to the original minimum and maximum  
summary(redwines$quality)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 3.000000 5.000000 6.000000 5.636023 6.000000 8.000000

# create training and test data  
redwines\_train <- redwines\_norm[1:1199, ]  
redwines\_test <- redwines\_norm[1200:1599, ]

## Step 3: Training a model on the data —-

# train the neuralnet model  
library(neuralnet)  
  
# simple ANN with only a single hidden neuron  
set.seed(12345) # to guarantee repeatable results  
redwines\_model <- neuralnet(formula = quality ~ fixed.acidity + volatile.acidity +  
 citric.acid + residual.sugar + chlorides +   
 free.sulfur.dioxide + total.sulfur.dioxide + density + pH + sulphates + alcohol,  
 data = redwines\_train)  
  
# visualize the network topology  
plot(redwines\_model)  
  
# alternative plot  
library(NeuralNetTools)   
  
# plotnet  
par(mar = numeric(4), family = 'serif')  
plotnet(redwines\_model, alpha = 0.6)



# number of training steps = 1266

# Sum of Squared Errors (SSE) = 10.1

## Step 4: Evaluating model performance —-

# obtain model results  
model\_results <- compute(redwines\_model, redwines\_test[1:11])  
  
# obtain predicted strength values  
predicted\_quality <- model\_results$net.result  
  
# examine the correlation between predicted and actual values  
cor(predicted\_quality, redwines\_test$quality)

## [,1]  
## [1,] 0.6694363292

# produce actual predictions by   
  
head(predicted\_quality)

## [,1]  
## 1200 0.5901721411  
## 1201 0.4343620139  
## 1202 0.4748186815  
## 1203 0.4016661467  
## 1204 0.6400643590  
## 1205 0.6936971636

redwines\_train\_original\_quality <- redwines[1:1199,"quality"]  
  
quality\_min <- min(redwines\_train\_original\_quality)  
quality\_max <- max(redwines\_train\_original\_quality)  
  
head(redwines\_train\_original\_quality)

## [1] 6 6 5 5 7 6

# custom normalization function  
unnormalize <- function(x, min, max) {   
 return( (max - min)\*x + min )  
}  
  
quality\_pred <- unnormalize(predicted\_quality, quality\_min, quality\_max)  
summary(quality\_pred)

## V1   
## Min. :4.931130   
## 1st Qu.:5.221504   
## Median :5.467203   
## Mean :5.601905   
## 3rd Qu.:5.964339   
## Max. :6.756393

# Correlation = 0.669

# The correlation has a lot of scope of improvement

# As networks with more complex topologies are capable of learning more difficult

# concepts, we will try to increase the number of hidden nodes to 5

## Step 5: Improving model performance —-

# A more complex neural network topology with 5 hidden neurons  
set.seed(12345) # to guarantee repeatable results  
redwines\_model2 <- neuralnet(formula = quality ~ fixed.acidity + volatile.acidity +  
 citric.acid + residual.sugar + chlorides +   
 free.sulfur.dioxide + total.sulfur.dioxide + density + pH + sulphates + alcohol,  
 data = redwines\_train, hidden = 5, act.fct = "logistic")  
  
# plot the network  
plot(redwines\_model2)  
  
# plotnet  
par(mar = numeric(4), family = 'serif')  
plotnet(redwines\_model2, alpha = 0.6)  
  
# evaluate the results as we did before  
model\_results2 <- compute(redwines\_model2, redwines\_test[1:11])  
predicted\_quality2 <- model\_results2$net.result  
cor(predicted\_quality2, redwines\_test$quality)

## [,1]  
## [1,] 0.6740467916

# 

# 

# The correlation has improved to 0.674 with more complex network topology.

**# try different activation function**# a more complex neural network topology with 5 hidden neurons  
set.seed(12345) # to guarantee repeatable results  
redwines\_model2 <- neuralnet(formula = quality ~ fixed.acidity + volatile.acidity +  
 citric.acid + residual.sugar + chlorides +   
 free.sulfur.dioxide + total.sulfur.dioxide + density + pH + sulphates + alcohol,  
 data = redwines\_train, hidden = 5, act.fct = "tanh")  
  
# evaluate the results as we did before  
model\_results2 <- compute(redwines\_model2, redwines\_test[1:11])  
predicted\_quality2 <- model\_results2$net.result  
cor(predicted\_quality2, redwines\_test$quality)

## [,1]  
## [1,] 0.7173802345

# The correlation has further improved to 0.717 with more complex network topology and a different activation function - tanh.

# Question 3: Read the blog post Multilable classification with neuralnet package and run the code.

#Multilabel classification with neuralnet package  
  
# load libs  
require(neuralnet)  
require(nnet)

## Loading required package: nnet

require(ggplot2)

## Loading required package: ggplot2

set.seed(10)

# The dataset

# From UCI Machine Learning Repository - wine dataset.

# This dataset contains the results of a chemical analysis on 3 different kind of wines.

# The target variable is the label of the wine which is a factor with 3 (unordered) levels.

# The predictors are all continuous and represent 13 variables obtained as a result of chemical measurements.

# Step 1: Loading the data———

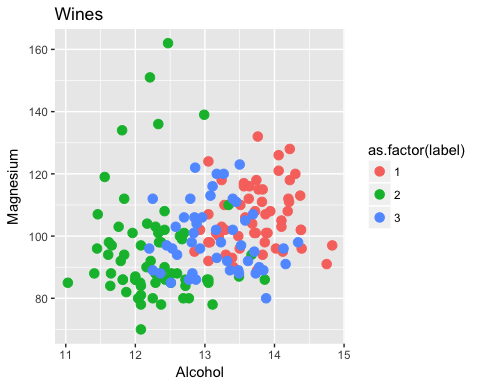
# Load data and set variables names  
wines <- read.csv("wines.csv")  
names(wines) <- c("label",  
 "Alcohol",  
 "Malic\_acid",  
 "Ash",  
 "Alcalinity\_of\_ash",  
 "Magnesium",  
 "Total\_phenols",  
 "Flavanoids",  
 "Nonflavanoid\_phenols",  
 "Proanthocyanins",  
 "Color\_intensity",  
 "Hue",  
 "OD280\_OD315\_of\_diluted\_wines",  
 "Proline")  
head(wines)

## label Alcohol Malic\_acid Ash Alcalinity\_of\_ash Magnesium Total\_phenols  
## 1 1 13.20 1.78 2.14 11.2 100 2.65  
## 2 1 13.16 2.36 2.67 18.6 101 2.80  
## 3 1 14.37 1.95 2.50 16.8 113 3.85  
## 4 1 13.24 2.59 2.87 21.0 118 2.80  
## 5 1 14.20 1.76 2.45 15.2 112 3.27  
## 6 1 14.39 1.87 2.45 14.6 96 2.50  
## Flavanoids Nonflavanoid\_phenols Proanthocyanins Color\_intensity Hue  
## 1 2.76 0.26 1.28 4.38 1.05  
## 2 3.24 0.30 2.81 5.68 1.03  
## 3 3.49 0.24 2.18 7.80 0.86  
## 4 2.69 0.39 1.82 4.32 1.04  
## 5 3.39 0.34 1.97 6.75 1.05  
## 6 2.52 0.30 1.98 5.25 1.02  
## OD280\_OD315\_of\_diluted\_wines Proline  
## 1 3.40 1050  
## 2 3.17 1185  
## 3 3.45 1480  
## 4 2.93 735  
## 5 2.85 1450  
## 6 3.58 1290

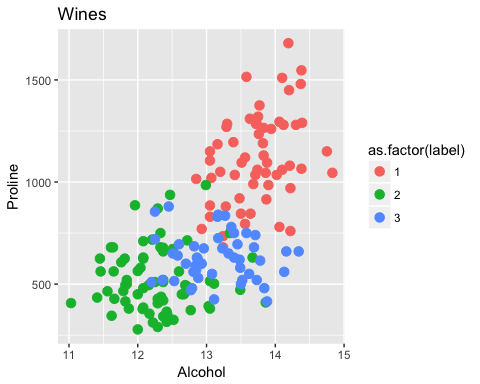
str(wines)

## 'data.frame': 177 obs. of 14 variables:  
## $ label : int 1 1 1 1 1 1 1 1 1 1 ...  
## $ Alcohol : num 13.2 13.2 14.4 13.2 14.2 ...  
## $ Malic\_acid : num 1.78 2.36 1.95 2.59 1.76 1.87 2.15 1.64 1.35 2.16 ...  
## $ Ash : num 2.14 2.67 2.5 2.87 2.45 2.45 2.61 2.17 2.27 2.3 ...  
## $ Alcalinity\_of\_ash : num 11.2 18.6 16.8 21 15.2 14.6 17.6 14 16 18 ...  
## $ Magnesium : int 100 101 113 118 112 96 121 97 98 105 ...  
## $ Total\_phenols : num 2.65 2.8 3.85 2.8 3.27 2.5 2.6 2.8 2.98 2.95 ...  
## $ Flavanoids : num 2.76 3.24 3.49 2.69 3.39 2.52 2.51 2.98 3.15 3.32 ...  
## $ Nonflavanoid\_phenols : num 0.26 0.3 0.24 0.39 0.34 0.3 0.31 0.29 0.22 0.22 ...  
## $ Proanthocyanins : num 1.28 2.81 2.18 1.82 1.97 1.98 1.25 1.98 1.85 2.38 ...  
## $ Color\_intensity : num 4.38 5.68 7.8 4.32 6.75 5.25 5.05 5.2 7.22 5.75 ...  
## $ Hue : num 1.05 1.03 0.86 1.04 1.05 1.02 1.06 1.08 1.01 1.25 ...  
## $ OD280\_OD315\_of\_diluted\_wines: num 3.4 3.17 3.45 2.93 2.85 3.58 3.58 2.85 3.55 3.17 ...  
## $ Proline : int 1050 1185 1480 735 1450 1290 1295 1045 1045 1510 ...

#Visualize the data  
plt1 <- ggplot(wines, aes(x = Alcohol, y = Magnesium, colour = as.factor(label))) +  
 geom\_point(size=3) +  
 ggtitle("Wines")  
  
plt2 <- ggplot(wines, aes(x = Alcohol, y = Proline, colour = as.factor(label))) +  
 geom\_point(size=3) +  
 ggtitle("Wines")  
  
plt1



plt2



# Step 2: Preprocessing———

# A. Encode target variable as a one hot vector multilabel data  
#The encoding of the categorical variables is needed when using neuralnet  
#since it does not like factors at all.  
  
train <- cbind(wines[, 2:14], class.ind(as.factor(wines$label)))  
  
# Set labels name  
names(train) <- c(names(wines)[2:14],"l1","l2","l3")  
  
# B. Standardize the predictors  
# Scale data  
scl <- function(x){ (x - min(x))/(max(x) - min(x)) }  
train[, 1:13] <- data.frame(lapply(train[, 1:13], scl))  
head(train)

## Alcohol Malic\_acid Ash Alcalinity\_of\_ash Magnesium  
## 1 0.5710526316 0.2055335968 0.4171122995 0.03092783505 0.3260869565  
## 2 0.5605263158 0.3201581028 0.7005347594 0.41237113402 0.3369565217  
## 3 0.8789473684 0.2391304348 0.6096256684 0.31958762887 0.4673913043  
## 4 0.5815789474 0.3656126482 0.8074866310 0.53608247423 0.5217391304  
## 5 0.8342105263 0.2015810277 0.5828877005 0.23711340206 0.4565217391  
## 6 0.8842105263 0.2233201581 0.5828877005 0.20618556701 0.2826086957  
## Total\_phenols Flavanoids Nonflavanoid\_phenols Proanthocyanins  
## 1 0.5758620690 0.5105485232 0.2452830189 0.2744479495  
## 2 0.6275862069 0.6118143460 0.3207547170 0.7570977918  
## 3 0.9896551724 0.6645569620 0.2075471698 0.5583596215  
## 4 0.6275862069 0.4957805907 0.4905660377 0.4447949527  
## 5 0.7896551724 0.6434599156 0.3962264151 0.4921135647  
## 6 0.5241379310 0.4599156118 0.3207547170 0.4952681388  
## Color\_intensity Hue OD280\_OD315\_of\_diluted\_wines Proline  
## 1 0.2645051195 0.4634146341 0.7802197802 0.5506419401  
## 2 0.3754266212 0.4471544715 0.6959706960 0.6469329529  
## 3 0.5563139932 0.3089430894 0.7985347985 0.8573466476  
## 4 0.2593856655 0.4552845528 0.6080586081 0.3259629101  
## 5 0.4667235495 0.4634146341 0.5787545788 0.8359486448  
## 6 0.3387372014 0.4390243902 0.8461538462 0.7218259629  
## l1 l2 l3  
## 1 1 0 0  
## 2 1 0 0  
## 3 1 0 0  
## 4 1 0 0  
## 5 1 0 0  
## 6 1 0 0

# Step 3: Fitting the model with neuralnet———

# Set up formula as neuralnet does not like the formula y~.  
n <- names(train)  
f <- as.formula(paste("l1 + l2 + l3 ~", paste(n[!n %in% c("l1","l2","l3")], collapse = " + ")))  
f

## l1 + l2 + l3 ~ Alcohol + Malic\_acid + Ash + Alcalinity\_of\_ash +   
## Magnesium + Total\_phenols + Flavanoids + Nonflavanoid\_phenols +   
## Proanthocyanins + Color\_intensity + Hue + OD280\_OD315\_of\_diluted\_wines +   
## Proline

#train the neural network with the full dataset  
  
nn <- neuralnet(f,  
 data = train,  
 hidden = c(13, 10, 3),  
 act.fct = "logistic",  
 linear.output = FALSE,  
 lifesign = "minimal")

## hidden: 13, 10, 3 thresh: 0.01 rep: 1/1 steps: 88 error: 0.03039 time: 0.08 secs

#Plotting the model  
plot(nn)

# 

# Step 4: Evaluating model performance———

# Compute predictions  
pr.nn <- compute(nn, train[, 1:13])  
  
# Extract results  
pr.nn\_ <- pr.nn$net.result  
head(pr.nn\_)

## [,1] [,2] [,3]  
## [1,] 0.9897528761 0.003171322443 0.000006987838514  
## [2,] 0.9908394248 0.002331321781 0.000008693900073  
## [3,] 0.9914977585 0.002103254765 0.000008649814003  
## [4,] 0.9855622778 0.004418327885 0.000008738518880  
## [5,] 0.9916175055 0.002119520153 0.000008319926342  
## [6,] 0.9915542288 0.002144844815 0.000008337763696

# Accuracy (training set)  
original\_values <- max.col(train[, 14:16])  
pr.nn\_2 <- max.col(pr.nn\_)  
mean(pr.nn\_2 == original\_values)

## [1] 1

# 100% accuracy! this may be because the model over fitted the data.

# In order to assess the ???true accuracy??? of the model we need to perform cross validation.

# Step 5: Improving model performance———-

#Cross Validation  
  
# Set seed for reproducibility purposes  
set.seed(500)  
  
# 10 fold cross validation  
k <- 10  
  
# Results from cv  
outs <- NULL  
  
# Train test split proportions  
#95% of the dataset will be used as training set while the remaining 5% as test set.  
proportion <- 0.95 # Set to 0.995 for LOOCV  
  
# Crossvalidate, go!  
for(i in 1:k)  
{  
 index <- sample(1:nrow(train), round(proportion\*nrow(train)))  
 train\_cv <- train[index, ]  
 test\_cv <- train[-index, ]  
 nn\_cv <- neuralnet(f,  
 data = train\_cv,  
 hidden = c(13, 10, 3),  
 act.fct = "logistic",  
 linear.output = FALSE)  
  
 # Compute predictions  
 pr.nn <- compute(nn\_cv, test\_cv[, 1:13])  
 # Extract results  
 pr.nn\_ <- pr.nn$net.result  
 # Accuracy (test set)  
 original\_values <- max.col(test\_cv[, 14:16])  
 pr.nn\_2 <- max.col(pr.nn\_)  
 outs[i] <- mean(pr.nn\_2 == original\_values)  
}  
  
mean(outs)

## [1] 0.9888888889

# Accuracy = 98.8%