Assignment 4: Neural Networks Using R

Written by Daanish Ahmed

DATA 630 9041

Semester Summer 2017

Professor Edward Herranz

UMUC

July 16, 2017

Introduction

Neural networks are becoming increasingly important in the field of data mining. A neural network is a set of input and output nodes that are linked to each other through a series of weighted connections that may span one or more hidden layers. It can learn by adjusting the weights of its connections—allowing it to predict the class of the dependent variable (Han, Kamber, & Pei, 2011). This is achieved through a process known as backpropagation, in which the algorithm adjusts the values of the weights from the output layer to the input layer—causing the weights to eventually converge (Han et al., 2011). Neural networks are designed to perform similarly to how a human brain functions, making them useful for a wide variety of supervised and unsupervised learning situations (Bati, 2015). My goal in this analysis is to implement a neural network model using R software on a dataset containing heart disease information (Hosmer, Lemeshow, & May, 2014). The focus of my model is on predicting a patient's vital status at the time of their last follow-up session—indicating whether the patient is alive or dead. I will experiment with the parameters by creating three models, each with a different number of hidden layers and nodes within each layer. I will then compare the accuracies of the three models to determine which set of parameters is the best match for my dataset. I expect that the results of my analysis will provide an accurate model that can be used to predict the likelihood of a patient dying from heart disease. Such a model could greatly contribute to preventing deaths in the future.

Data Exploration

I will begin this analysis by describing the structure and variables of the initial dataset. This dataset contains 14 variables and 481 observations (see Figure 1 in Appendix B). These

variables include the patient's age, gender, peak cardiac enzyme, cardiogenic shock complications, left heart failure complications, MI type, and length of their hospital stay (Hosmer & Lemeshow, 1998). The dependent variable of my model is "FSTAT," which is the patient's vital status during their final follow-up session. All variables in the dataset are integers, though many of them represent categorical values such as 0 and 1. For further exploration, I generated a set of descriptive statistics for the data (see Figure 2 in Appendix B). Note that this figure is missing the ID variable because it was removed during preprocessing. These numbers reveal that most patients are male, the median patient age is 68 years old, and the average length of hospital stays is about 14 days. The figure also indicates that there are no missing values in any of the variables. In addition, we see that the dependent variable FSTAT—which has values of 0 and 1—has an average value of 0.52. This suggests that there is a slightly higher number of patients in this dataset who have died by the time of their last follow-up session. It also means that the dataset is balanced, and therefore it is unlikely that the model will require oversampling.

Preprocessing

Prior to model implementation, I performed several important preprocessing steps on the data. First, I installed and activated the "neuralnet" package in R, which is a requirement to using the neural network algorithm. Next, I removed the unique identifier variable (ID) from the dataset because it is unnecessary for my analysis and may cause the model to be overly complex. After this step, I checked for any missing values and verified that all variables are in numeric form (see Figure 2 in Appendix B). As stated earlier, all of the variables are already in integer format and there are no missing values in any of the variables. Finally, the neural network method in R requires that we set all input variables to the same scale such that they have the same values for

mean and standard deviation ("Neural Network Analysis," 2017). I performed this step on the 12 remaining independent variables, and then I verified that they all have the same mean value of 0 (see Figure 3 in Appendix B). This step completes the preprocessing stage, and it is now possible to implement the neural network method on my dataset.

Initial Neural Network

I will now discuss the first of three neural networks that I will create for my analysis. This network will consist of a single hidden layer containing two nodes. First, I split the dataset into a training set with 70% of the data and a test set with 30% of the data. This step is needed to prevent overfitting in our model, which occurs when there are too many weights in the network (Hastie, Tibshirani, and Friedman, 2017). The next step is to implement the "neuralnet" method on the training data. This method uses backpropagation to determine the weights of the connections. Its input parameters include the formula, dataset, number of hidden layers and nodes, and whether it creates a linear output. For the formula, I used "FSTAT" as the dependent variable and all other variables as independent variables. The reason why I included all remaining variables is that I removed the unnecessary variables during the preprocessing stage. I used the training set as my dataset and used one hidden layer with two nodes. And since my goal is the classification of FSTAT, I set the linear output to false. Once the model was created, I examined the model's properties (see Figure 4 in Appendix B). This figure includes the number of rows in each property as well as that property's class and type. For instance, the "response" property has 315 instances and is of numeric type. I proceeded by analyzing the first 20 expected values of the response variable, and I found that most of them have values of 1 (see Figure 5 in Appendix B). I then examined the net result property, which shows the first 20 predictions for the response variable

(see Figure 6 in Appendix B). This figure is consistent with my findings in Figure 5 and indicates that the predictions themselves are accurate up to many decimal places.

Next, I examined the network result matrix, which includes information such as the number of training steps, the error, and the weights of each connection between nodes (see Figure 7 in Appendix B). We see that the error rate is 3.43 and that the model required 2369 steps to build it. Each row that ends with "1layhid1" indicates the weight of a connection to the first hidden layer node. For example, we see that the connection between the intercept and the first hidden layer node has a weight of 1.51. Similar information is displayed for the connections to the second hidden layer node ("1layhid2") and the output layer node ("FSTAT"). I then created a plot of the neural network, which shows the connections between the nodes and each connection's associated weight (see Figure 8 in Appendix B). For instance, the figure once again shows that the link between the intercept and the first hidden node has a weight of 1.51. Afterwards, I evaluated the predicted classes for the first 20 observations in the training set (see Figure 9 in Appendix B). These predictions are consistent with the actual values listed in Figure 5. Finally, I created confusion matrices for the training and test sets to evaluate the model's classification accuracy (see Figures 10 and 11 in Appendix B). These figures reveal that the accuracy on the training set is 99.7%, while the accuracy on the test set is 94.6%. Such findings suggest that the model is extremely effective at predicting the values in my dataset.

Initial Results

The results of my first neural network seem to be very promising. The predicted classes for the dependent variable (see Figure 9 in Appendix B) match perfectly with the first 20 expected

classes (see Figure 5 in Appendix B). Entries 11, 12, and 14 have values of 0 in both figures, while the remaining instances have values of 1. Additionally, the probabilities for the response variable also match the expected results with a high degree of precision (see Figure 6 in Appendix B). There is nearly a 0% chance of predicting a class of "1" for entries 11, 12, and 14, while there is roughly a 100% chance of doing so for the remaining entries. This finding suggests that the current neural network is extremely accurate at predicting the values in my dataset. This claim is reinforced by the classification accuracies obtained from the confusion matrices (see Figures 10 and 11 in Appendix B). As stated before, the model has a 99.7% accuracy on the training data and a 94.6% accuracy on the test data. Although the test accuracy is clearly lower than the training accuracy, it still appears to be a very accurate model. There are two additional findings from these confusion matrices. Firstly, the majority and minority classes appear to be balanced—meaning that there is no bias in the algorithm and no need to oversample the dataset. Secondly, there is a slightly higher number of instances having the class of 1. This means that there is a slightly larger number of patients who have died by the time of their last follow-up session. These findings are consistent with my earlier observations when examining the descriptive statistics during the data exploration phase (see Figure 3 in Appendix B).

Furthermore, I examined the structure of the network itself to gain further insights about my dataset. The result matrix indicates that the model's error is 3.43 (see Figure 7 in Appendix B), which is a very low error rate. This figure also shows that it took 2369 steps to build the model, meaning that it is not too complex or time-consuming to build. The plot of the neural network offers additional insight about the weights associated with each connection (see Figure 8 in Appendix B). Each of the 12 independent variables is connected to both hidden layer nodes, which are in turn connected to the output node "FSTAT." According to the image, the link between the

variable “DSTAT” and the first hidden layer node has a weight of 4.81—making it the strongest connection from the input layer to the hidden layer. Furthermore, we see that the link between the first hidden layer node and the output node has a weight of 62.6, making it the strongest connection between the hidden layer and the output layer. Unfortunately, one of the issues with neural networks is the difficulty of interpreting their structure (Han et al., 2011). Because of this, it is hard to gain meaning out of the weights in this model. However, the accuracy of the model’s predictions suggests that the current neural network is an effective tool for predicting the likelihood of a patient’s death from heart disease.

Second Neural Network

The next component of my analysis is to create a second neural network using different parameters. I wish to determine if changing the number of hidden layers and nodes will impact the accuracy of the network’s predictions. My new model will contain two hidden layers, one with 4 nodes and one with 2 nodes. The model was built using the same steps from the original model’s construction—the only difference is that I used a different number of hidden layers and nodes when calling the “neuralnet” method. After using this method to build the model, I examined the first 20 predictions for the dependent variable “FSTAT” (see Figure 12 in Appendix B). This figure reveals that the predicted classes match both my earlier predictions and the actual values from Figures 6 and 5 respectively. However, the precision of these predictions is much lower, since they are accurate to a smaller number of decimal places. Looking at the result matrix reveals that the new model has an error rate of 0.01 and that it was built after 673 steps (see Figure 13 in Appendix B). This indicates that the new model has a much lower error rate than the original model and that it is less complex and time-consuming to build.

The next step is to examine the visualization of this model (see Figure 14 in Appendix B). This image shows that each of the 12 input nodes is connected to all 4 nodes in the first hidden layer. These 4 nodes are connected to the 2 nodes in the second hidden layer, which are connected to the output layer. The network appears much more complex than the original network due to having a much higher number of connections and associated weights. Therefore, it is surprising that the new model was built using a smaller number of steps than the original model. Finally, I examined the confusion matrices on the training and test datasets to evaluate the model's classification accuracy (see Figures 15 and 16 in Appendix B). These figures reveal that the model's accuracy on training data is 100%, while its accuracy on test data is 95.2%. Both percentages are slightly higher than their respective accuracies from my original model. These matrices also predict a slightly higher number of death cases, which is consistent with the data in this dataset. Due to the higher classification accuracy and the lower error rate, it seems that this model is more accurate than the previous model. However, the probabilities from Figure 12 are still less precise than those from the first neural network. Therefore, I would claim that my second model has a higher accuracy but a lower precision than my first model.

Third Neural Network

The final part of my analysis is to create a third neural network for comparison with my first two models. This network will be simpler than the previous ones, since it will only have one hidden layer with a single node. I built this model using the same steps as before, but this time I used only one hidden layer and one node as input for the "neuralnet" method. After creating the model, I examined the first 20 probabilities for the dependent variable (see Figure 17 in Appendix B). These predictions match the expected values and the predicted classes from the first two

models. However, they are even less precise than the predictions for my second model, since they are only accurate to two or three decimal places. The result matrix reveals that the model has an error rate of 12.4 and that it required 3621 steps during its construction (see Figure 18 in Appendix B). This error rate is by far the highest of the three models, and the required number of steps causes this model to be more complex and time-consuming to build than either of the previous networks. This may seem surprising at first, given that the plot of this network indicates that it has fewer connections and weights (see Figure 19 in Appendix B). But this finding makes sense when comparing it to my second model—which required fewer steps to build even though it had a higher number of nodes and connections.

Finally, I examined the confusion matrices to evaluate the model's accuracy on training and test data (see Figures 20 and 21 in Appendix B). Based on these figures, the model has accuracies of 99.3% on the training data and 96.4% on the test data. Its training accuracy is slightly lower than those of the two previous models, but its test accuracy is slightly higher than those of either model. Since test results are more important to the analysis, it appears that my third neural network has the highest classification accuracy. The confusion matrices also reveal that there is a slightly higher number of heart disease death cases, which once again matches the values in this dataset. Additionally, this model has the lowest precision out of all three models due to the probabilities shown in Figure 17. This finding is similar to the results of my second model, which has a higher accuracy but a lower precision than the first model. However, one issue with the results is that my new model has the highest error rate of any model used in this analysis. This finding seems to contradict with the classification accuracies obtained from the confusion matrices. Thus, for future analysis I would recommend examining the data further to see why the model has a high accuracy while still having a high error rate.

Conclusion

In this analysis, I designed three distinct neural networks to predict the likelihood of a patient dying from heart disease. Each of these models involved using a different number of hidden layers and nodes. I found that all three models predicted a slightly higher probability of a patient dying from heart disease, which matches the distribution of values in my dataset. Furthermore, each of these models has a classification accuracy on the test data that is above 94%. This suggests that all three models are highly accurate for predicting the dependent variable in my dataset. However, determining a victim's survival rate from heart disease requires an algorithm with the highest level of accuracy. Therefore, I would argue that my second neural network—which uses the largest number of hidden layers and nodes—is the most effective model. It has the lowest error rate, requires the smallest number of steps to build it, and has the second highest classification accuracy. The third model, which has the highest classification accuracy, unfortunately has the largest error rate by far and requires the greatest number of steps. Both the error rate and the number of steps seem to be lower for models with a higher number of hidden layers and nodes. Because of this, I consider the second model to be more accurate than the third model. And though the first model has a greater level of precision than the second model, it also has a lower accuracy and a higher error rate. Thus, I conclude that my second network is most useful for preventing heart disease-related deaths in the future. This model has a higher number of connections and weights, but it still has a reasonable level of complexity.

One of the main limitations in this analysis was the difficulty of interpreting the structure of these models. Our current understanding of neural networks means that it is very difficult to find meaning from the weights in the model (Han et al., 2011). Because of this, it is not easy to determine which variables have the greatest impact towards predicting the dependent variable. I

would therefore recommend using other algorithms—such as decision tree classification or the Apriori algorithm—to better understand the relationships between the independent variables and the response variable. Another limitation in my analysis was the fact that my third model had the highest test accuracy, yet it also had the highest rate of error. I would recommend further research to discover why these seemingly contradicting values are occurring in this model. If future research proves that this model's error rate is lower than that of my other models, then I may recommend using it to prevent deaths related to heart disease.

References

Bati, F. (2015, Fall). Classification using Artificial Neural Networks (ANN). Lecture presented at UMUC. Retrieved July 9, 2017.

Han, J., Kamber, M., & Pei, J. (2011). *Data mining: concepts and techniques* (3rd ed.). Retrieved June 27, 2017.

Hastie, T., Tibshirani, R., & Friedman, J. H. (2017). *The elements of statistical learning: data mining, inference, and prediction*. Retrieved July 14, 2017.

Hosmer, D. W., & Lemeshow, S. (1998). Description of the variables in the WHAS. Retrieved July 15, 2017, from ftp://ftp.wiley.com/public/sci_tech_med/survival/First-Edition%20Data.zip

Hosmer, D. W., Lemeshow, S., & May, S. (2014, July 5). Worcester Heart Attack Study, Nested Control, 1st Ed. Retrieved July 15, 2017, from http://www.statground.org/index.php?document_srl=1130&mid=Datasets

Neural Network Analysis Using R – Analyzing Diabetes Data (2017). Retrieved July 10, 2017.

Appendix A

R Script for Assignment 4

```
# DATA 630 Assignment 3

# Written by Daanish Ahmed

# Semester Summer 2017

# July 14, 2017

# Professor Edward Herranz


# This R script involves the creation of three neural networks using a dataset

# on heart disease information.  The purpose of this assignment is to build a

# model that accurately predicts the class of the dependent variable "FSTAT,"

# which represents a patient's vital status at the time of their last follow-

# up session.  I will experiment with using different numbers of hidden layers

# and hidden layer nodes in each model.  At the end, I will evaluate the

# accuracies of the three models to determine which set of parameters will

# produce the most accurate predictions.



# This section of code covers opening the dataset and initializing the packages

# that are used in this script.


# Sets the working directory for this assignment.  Please change this directory
```

```
# to whichever directory you are using, and make sure that all files are placed
# in that location.
setwd("~/Class Documents/2016-17 Summer/DATA 630/R/Assignment 4")

# In order to run the neural network commands, we need to install the neuralnet
# package:

# If you have not installed this package yet, remove the # symbol below.
# install.packages("neuralnet")

# Loads the neuralnet package into the system.
library("neuralnet")

# Opens the CSV file "whas1.csv".
heart_dis <- read.csv("whas1.csv", head=TRUE, sep=",")

# End of opening the dataset.


# This section of code covers data preprocessing.  It includes exploration of
# the original dataset, removing variables, and dealing with missing values.

# Previews the heart disease dataset.
```

```
View(heart_dis)

# Displays the structure of the data.  This is necessary to see if there are
# any unique identifiers (IDs) that can be removed.  Such variables are not
# useful for the analysis and should be removed.
str(heart_dis)

# The first variable is an ID, and we remove it.
heart_dis <- heart_dis[, -1]

# Verifies that the ID variable has been removed.
str(heart_dis)

# Displays the descriptive statistics for all variables in the dataset.  This
# shows whether all variables are numeric and if there are any missing values.
summary(heart_dis)

# Since all variables are numeric and there are no missing values, we do not
# need to worry about these issues.

# Sets the input variables to the same scale, such that they have the same
# mean and standard deviation.
heart_dis[1:12] <- scale(heart_dis[1:12])
```

```
# Verifies that the variables are set to the same scale.  All variables
# except for "fstat" should have a mean equal to 0.
summary(heart_dis)

# End of data preprocessing.


# This section of code covers the creation of the first neural network model.
# It includes dividing the data into training and test datasets, creating and
# visualizing the model, and analyzing the model on the training and test
# datasets.

# Generates a random seed to allow us to reproduce the results.
set.seed(12345)

# The following code splits the dataset into a training set consisting of
# 70% of the data and a test set containing 30% of the data.
ind <- sample(2, nrow(heart_dis), replace = TRUE, prob = c(0.7, 0.3))
train.data <- heart_dis[ind == 1, ]
test.data <- heart_dis[ind == 2, ]

# This command implements the neural network method on training data using
```

```
# class as the dependent variable and all other variables as independent
# variables.  We use one hidden layer with 2 nodes, and linear output set to
# false for classification models.
nn <- neuralnet(formula = FSTAT ~ AGE + SEX + CPK + SHO + CHF + MIORD +
                  MITYPE + YEAR + YRGRP + LENSTAY + DSTAT + LENFOL,
                data = train.data, hidden = 2,
                err.fct = "ce", linear.output = FALSE)

# Shows the summary of the model.
summary(nn)

# We will examine the following properties: the response values, result
# matrix, and net result probabilities.

# Shows the values of the dependent variable for the first 20 observations.
nn$response[1:20]

# Shows the probability for the first 20 records.
nn$net.result[[1]][1:20]

# Shows the network result matrix, which includes information on the number
# of training steps, the error, and the weights.
nn$result.matrix
```

```
# Creates a visualization of the neural network.
plot(nn)

# Computes the predicted values for the training set and rounds them to the
# nearest integer (either 0 or 1).
mypredict <- compute(nn, nn$covariate)$net.result
mypredict <- apply(mypredict, c(1), round)

# Shows the first 20 predicted values.
mypredict[1:20]

# Creates the confusion matrix on the training data.
table(mypredict, train.data$FSTAT, dnn =c("Predicted", "Actual"))

# Computes the predicted values for the test set and rounds them to the
# nearest integer (either 0 or 1).
testPred <- compute(nn, test.data[, 0:12])$net.result
testPred <- apply(testPred, c(1), round)

# Creates the confusion matrix on the test data.
table(testPred, test.data$FSTAT, dnn = c("Predicted", "Actual"))
```

```
# End of creating the first neural network model.



# This section of code covers the creation of the second neural network
# model.  This model will have two hidden layers, one with 4 nodes and one
# with 2 nodes.  Everything else will stay the same.  It will require many
# of the same steps as the previous model.

# Generates a random seed to allow us to reproduce the results.
set.seed(12345)

# Implements the neural network model on the same training set using the
# same variables as input and output parameters.  The only difference is
# that there are now two hidden layers, one with 4 nodes and one with 2
# nodes.
nn <- neuralnet(formula = FSTAT ~ AGE + SEX + CPK + SHO + CHF + MIORD +
                  MITYPE + YEAR + YRGRP + LENSTAY + DSTAT + LENFOL,
                data = train.data, hidden = c(4, 2),
                err.fct = "ce", linear.output = FALSE)

# Shows the probability for the first 20 records.
nn$net.result[[1]][1:20]
```

```
# Shows the network result matrix, which includes information on the number
# of training steps, the error, and the weights.
nn$result.matrix

# Creates a visualization of the neural network.
plot(nn)

# Computes the predicted values for the training set and rounds them to the
# nearest integer (either 0 or 1).
mypredict <- compute(nn, nn$covariate)$net.result
mypredict <- apply(mypredict, c(1), round)

# Creates the confusion matrix on the training data.
table(mypredict, train.data$FSTAT, dnn =c("Predicted", "Actual"))

# Computes the predicted values for the test set and rounds them to the
# nearest integer (either 0 or 1).
testPred <- compute(nn, test.data[, 0:12])$net.result
testPred <- apply(testPred, c(1), round)

# Creates the confusion matrix on the test data.
table(testPred, test.data$FSTAT, dnn = c("Predicted", "Actual"))
```

```
# End of creating the second neural network model.



# This section of code covers the creation of the third neural network
# model.  This model will only have one hidden layer with a single node.
# Everything else will stay the same.  It will require many of the same
# steps as the previous model.

# Generates a random seed to allow us to reproduce the results.
set.seed(12345)

# Implements the neural network model on the training set using the same
# variables as input and output parameters.  There is now only one hidden
# layer with one node.
nn <- neuralnet(formula = FSTAT ~ AGE + SEX + CPK + SHO + CHF + MIORD +
                  MITYPE + YEAR + YRGRP + LENSTAY + DSTAT + LENFOL,
                data = train.data, hidden = 1,
                err.fct = "ce", linear.output = FALSE)

# Shows the probability for the first 20 records.
nn$net.result[[1]][1:20]

# Shows the network result matrix, which includes information on the number
```

```
# of training steps, the error, and the weights.
nn$result.matrix

# Creates a visualization of the neural network.
plot(nn)

# Computes the predicted values for the training set and rounds them to the
# nearest integer (either 0 or 1).
mypredict <- compute(nn, nn$covariate)$net.result
mypredict <- apply(mypredict, c(1), round)

# Creates the confusion matrix on the training data.
table(mypredict, train.data$FSTAT, dnn =c("Predicted", "Actual"))

# Computes the predicted values for the test set and rounds them to the
# nearest integer (either 0 or 1).
testPred <- compute(nn, test.data[, 0:12])$net.result
testPred <- apply(testPred, c(1), round)

# Creates the confusion matrix on the test data.
table(testPred, test.data$FSTAT, dnn = c("Predicted", "Actual"))

# End of creating the third neural network model.
```

```
# End of script.
```

Appendix B

Relevant R Output Images

```
> str(heart_dis)
'data.frame':   481 obs. of  14 variables:
 $ ID      : int  1 2 3 4 5 6 7 8 9 10 ...
 $ AGE     : int  62 78 81 79 60 72 60 83 78 72 ...
 $ SEX     : int  1 1 1 1 1 0 1 1 0 1 ...
 $ CPK     : int  485 910 320 3290 2500 99 1200 160 66 99 ...
 $ SHO     : int  1 0 1 1 1 0 0 0 0 0 ...
 $ CHF     : int  1 1 1 1 1 0 0 0 1 0 ...
 $ MIORD   : int  0 1 0 1 1 0 0 0 1 0 ...
 $ MITYPE  : int  1 1 1 1 1 1 1 1 1 1 ...
 $ YEAR    : int  1 1 1 1 1 1 1 1 1 1 ...
 $ YRGRP   : int  1 1 1 1 1 1 1 1 1 1 ...
 $ LENSTAY: int  1 1 1 1 2 2 2 3 3 3 ...
 $ DSTAT   : int  1 1 1 1 1 1 0 1 1 0 ...
 $ LENFOL  : int  1 1 1 1 2 2 2 3 3 5586 ...
 $ FSTAT   : int  1 1 1 1 1 1 1 1 1 0 ...
>
```

Figure 1. Initial Data Structure of Heart Disease Dataset.

```
> summary(heart_dis)
      AGE              SEX              CPK              SHO              CHF              MIORD            MITYPE
 Min.   :24.00   Min.   :0.0000   Min.   :  10.0   Min.   :0.000   Min.   :0.0000   Min.   :0.0000   Min.   :1.00
 1st Qu.:59.00   1st Qu.:0.0000   1st Qu.: 270.0   1st Qu.:0.000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:1.00
 Median :68.00   Median :0.0000   Median : 587.0   Median :0.000   Median :0.0000   Median :0.0000   Median :1.00
 Mean   :67.48   Mean   :0.4033   Mean   : 941.5   Mean   :0.079   Mean   :0.4075   Mean   :0.3597   Mean   :1.43
 3rd Qu.:77.00   3rd Qu.:1.0000   3rd Qu.:1146.0   3rd Qu.:0.000   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:2.00
 Max.   :98.00   Max.   :1.0000   Max.   :9000.0   Max.   :1.000   Max.   :1.0000   Max.   :1.0000   Max.   :3.00
      YEAR            YRGRP          LENSTAY          DSTAT            LENFOL          FSTAT
 Min.   :1.000   Min.   :1.000   Min.   : 1.00   Min.   :0.0000   Min.   :   1   Min.   :0.0000
 1st Qu.:2.000   1st Qu.:1.000   1st Qu.: 8.00   1st Qu.:0.0000   1st Qu.: 150   1st Qu.:0.0000
 Median :3.000   Median :2.000   Median :12.00   Median :0.0000   Median :1420   Median :1.0000
 Mean   :3.422   Mean   :1.971   Mean   :13.86   Mean   :0.1705   Mean   :1735   Mean   :0.5177
 3rd Qu.:5.000   3rd Qu.:3.000   3rd Qu.:17.00   3rd Qu.:0.0000   3rd Qu.:2551   3rd Qu.:1.0000
 Max.   :6.000   Max.   :3.000   Max.   :71.00   Max.   :1.0000   Max.   :5843   Max.   :1.0000
>
```

Figure 2. Descriptive Statistics of Variables After ID Removal.

```
> heart_dis[1:12] <- scale(heart_dis[1:12])
> summary(heart_dis)
      AGE                SEX                CPK                SHO                CHF
 Min.   :-3.42922347   Min.   :-0.8213117   Min.   :-0.8228566   Min.   :-0.2925755   Min.   :-0.8284259
 1st Qu.:-0.66908878   1st Qu.:-0.8213117   1st Qu.:-0.5931916   1st Qu.:-0.2925755   1st Qu.:-0.8284259
 Median : 0.04066014   Median :-0.8213117   Median :-0.3131770   Median :-0.2925755   Median :-0.8284259
 Mean   : 0.00000000   Mean   : 0.0000000   Mean   : 0.0000000   Mean   : 0.0000000   Mean   : 0.0000000
 3rd Qu.: 0.75040906   3rd Qu.: 1.2150333   3rd Qu.: 0.1806027   3rd Qu.:-0.2925755   3rd Qu.: 1.2045989
 Max.   : 2.40648987   Max.   : 1.2150333   Max.   : 7.1182516   Max.   : 3.4108149   Max.   : 1.2045989
     MIORD              MITYPE              YEAR               YRGRP              LENSTAY
 Min.   :-0.7486792   Min.   :-0.827205   Min.   :-1.4620249   Min.   :-1.2168052   Min.   :-1.4392436
 1st Qu.:-0.7486792   1st Qu.:-0.827205   1st Qu.:-0.8583906   1st Qu.:-1.2168052   1st Qu.:-0.6557459
 Median :-0.7486792   Median :-0.827205   Median :-0.2547563   Median : 0.0364781   Median :-0.2080329
 Mean   : 0.0000000   Mean   : 0.000000   Mean   : 0.0000000   Mean   : 0.0000000   Mean   : 0.0000000
 3rd Qu.: 1.3329087   3rd Qu.: 1.094948   3rd Qu.: 0.9525124   3rd Qu.: 1.2897614   3rd Qu.: 0.3516083
 Max.   : 1.3329087   Max.   : 3.017100   Max.   : 1.5561467   Max.   : 1.2897614   Max.   : 6.3957333
     DSTAT               LENFOL              FSTAT
 Min.   :-0.4528648   Min.   :-1.0271612   Min.   :0.0000000
 1st Qu.:-0.4528648   1st Qu.:-0.9389009   1st Qu.:0.0000000
 Median :-0.4528648   Median :-0.1866152   Median :1.0000000
 Mean   : 0.0000000   Mean   : 0.0000000   Mean   :0.5176715
 3rd Qu.:-0.4528648   3rd Qu.: 0.4833338   3rd Qu.:1.0000000
 Max.   : 2.2035738   Max.   : 2.4333531   Max.   :1.0000000
>
```

Figure 3. Descriptive Statistics After Scaling Independent Variables.

```
> summary(nn)
                  Length Class      Mode
call                 6   -none-     call
response           315   -none-     numeric
covariate         3780   -none-     numeric
model.list           2   -none-     list
err.fct              1   -none-     function
act.fct              1   -none-     function
linear.output        1   -none-     logical
data                13   data.frame list
net.result           1   -none-     list
weights              1   -none-     list
startweights         1   -none-     list
generalized.weights  1   -none-     list
result.matrix       32   -none-     numeric
>
```

Figure 4. Summary of Properties in the Neural Network Model.

```
> nn$response[1:20]
 [1] 1 1 1 1 1 1 1 1 1 1 0 0 1 0 1 1 1 1 1 1
>
```

Figure 5. Actual Values for the First 20 Instances of the Response Variable.

```
> nn$net.result[[1]][1:20]
 [1] 1.0000000000000000000 1.0000000000000000000 1.0000000000000000000 1.0000000000000000000
 [5] 1.0000000000000000000 1.0000000000000000000 1.0000000000000000000 1.0000000000000000000
 [9] 1.0000000000000000000 1.0000000000000000000 0.0000000015522686831555 0.0000000000003054339720
[13] 1.0000000000000000000 0.0000000000002422270802 1.0000000000000000000 1.0000000000000000000
[17] 1.0000000000000000000 1.0000000000000000000 1.0000000000000000000 1.0000000000000000000
>
```

Figure 6. First 20 Predictions of the Response Variable for the First Neural Network.

```
> nn$result.matrix
                                    1
error                       3.428503617894
reached.threshold           0.009048578891
steps                    2369.000000000000
Intercept.to.1layhid1       1.512168525994
AGE.to.1layhid1            -0.264704917161
SEX.to.1layhid1            -0.376713878508
CPK.to.1layhid1             0.485371831216
SHO.to.1layhid1             1.800049344992
CHF.to.1layhid1             0.749634786849
MIORD.to.1layhid1          -0.147967268874
MITYPE.to.1layhid1         -1.469309983136
YEAR.to.1layhid1           -4.740337744661
YRGRP.to.1layhid1          -2.047044946227
LENSTAY.to.1layhid1        -0.329662904782
DSTAT.to.1layhid1           4.807828137107
LENFOL.to.1layhid1         -4.644296372292
Intercept.to.1layhid2      15.687466899850
AGE.to.1layhid2            -1.233036796521
SEX.to.1layhid2            -0.658076003270
CPK.to.1layhid2             1.123940493826
SHO.to.1layhid2            -1.663924972145
CHF.to.1layhid2            -1.375870041614
MIORD.to.1layhid2          -0.107135906620
MITYPE.to.1layhid2          3.741950304964
YEAR.to.1layhid2          -28.417609237953
YRGRP.to.1layhid2          -0.007928141917
LENSTAY.to.1layhid2         1.940995368923
DSTAT.to.1layhid2          -4.012760245926
LENFOL.to.1layhid2         -32.587500779603
Intercept.to.FSTAT         -27.132202815431
1layhid.1.to.FSTAT          62.634492020765
1layhid.2.to.FSTAT          29.174592885146
>
```

Figure 7. Result Matrix of the First Neural Network.

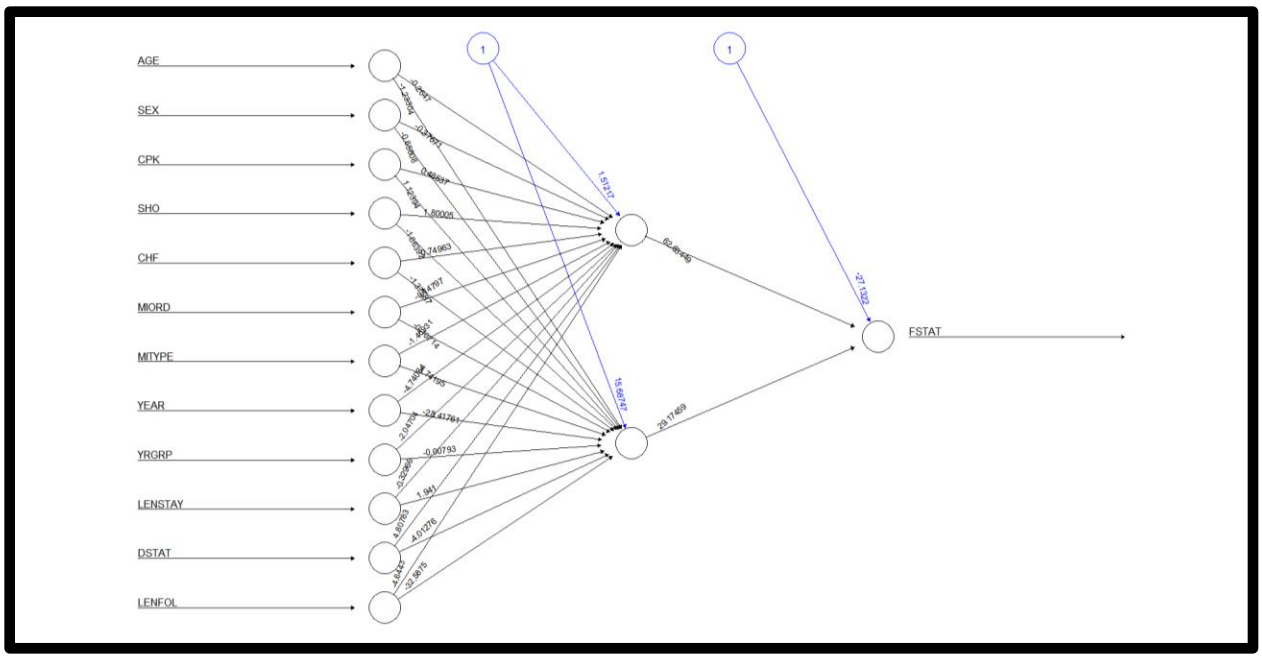

Figure 8. Plot of the First Neural Network.

```
> mypredict <- compute(nn, nn$covariate)$net.result
> mypredict <- apply(mypredict, c(1), round)
> mypredict[1:20]
 [1] 1 1 1 1 1 1 1 1 1 1 0 0 1 0 1 1 1 1 1 1
>
```

Figure 9. First 20 Predicted Classes for the First Neural Network.

```
> table(mypredict, train.data$FSTAT, dnn =c("Predicted", "Actual"))
         Actual
Predicted   0   1
        0 155   0
        1   1 159
>
```

Figure 10. Confusion Matrix for First Neural Network on Training Set.

```
> testPred <- compute(nn, test.data[, 0:12])$net.result
> testPred <- apply(testPred, c(1), round)
> table(testPred, test.data$FSTAT, dnn = c("Predicted", "Actual"))
         Actual
Predicted  0  1
        0 71  4
        1  5 86
>
```

Figure 11. Confusion Matrix for First Neural Network on Test Set.

```
> nn$net.result[[1]][1:20]
 [1] 0.99999998329875949831802 0.99999999054544763676943 8 0.99999999054541788279237 8 0.99999999054544697063562 4
 [5] 0.99999999054544186360971 0 0.99999999054544741472483 3 0.99999999054544741472483 3 0.99999999054399790754388 3
 [9] 0.99999999054526267361353 6 0.99999999054544475018957 4 0.00003179664002755819242 0 0.00000098831293636901757 6
[13] 0.99999999054537636045125 7 0.00000000000000482423050 8 0.99999999054470001258465 6 0.99999999054538679654768 9
[17] 0.99999989595674931841529 0.99999989873114558669442 0.99999999055074673126597 4 0.99999999054527866082509 0
>
```

Figure 12. First 20 Predictions of the Response Variable for the Second Neural Network.

```
> nn$result.matrix
                                    1
error                   0.011311513940
reached.threshold       0.008735380958
steps                 673.000000000000
Intercept.to.1layhid1  -3.161670987010
AGE.to.1layhid1        -0.591755425954
SEX.to.1layhid1         0.567543599878
CPK.to.1layhid1         1.933339236001
SHO.to.1layhid1         2.262671945830
CHF.to.1layhid1         0.513279583337
MIORD.to.1layhid1       0.695455549577
MITYPE.to.1layhid1     -1.231438844756
YEAR.to.1layhid1        3.142138834828
YRGRP.to.1layhid1       0.032695503025
LENSTAY.to.1layhid1     1.736988635467
DSTAT.to.1layhid1       0.677878715080
LENFOL.to.1layhid1      4.098793985106
Intercept.to.1layhid2  -2.199784274775
AGE.to.1layhid2        -0.113765112039
SEX.to.1layhid2        -0.390543485037
CPK.to.1layhid2        -0.968704391628
SHO.to.1layhid2        -0.371334344064
CHF.to.1layhid2         0.688564552427
MIORD.to.1layhid2      -0.283286308333
MITYPE.to.1layhid2      0.642828831544
YEAR.to.1layhid2        7.025943092860
YRGRP.to.1layhid2      -0.324368427190
LENSTAY.to.1layhid2     0.168297317137
DSTAT.to.1layhid2      -0.841003276687
LENFOL.to.1layhid2      9.588981101741
Intercept.to.1layhid3  -4.831840789981
AGE.to.1layhid3        -0.663778369874
SEX.to.1layhid3         1.001677941053
CPK.to.1layhid3        -0.072049264564
SHO.to.1layhid3        -0.682646944887
CHF.to.1layhid3         1.638212240840
```

Figure 13. Result Matrix of the Second Neural Network.

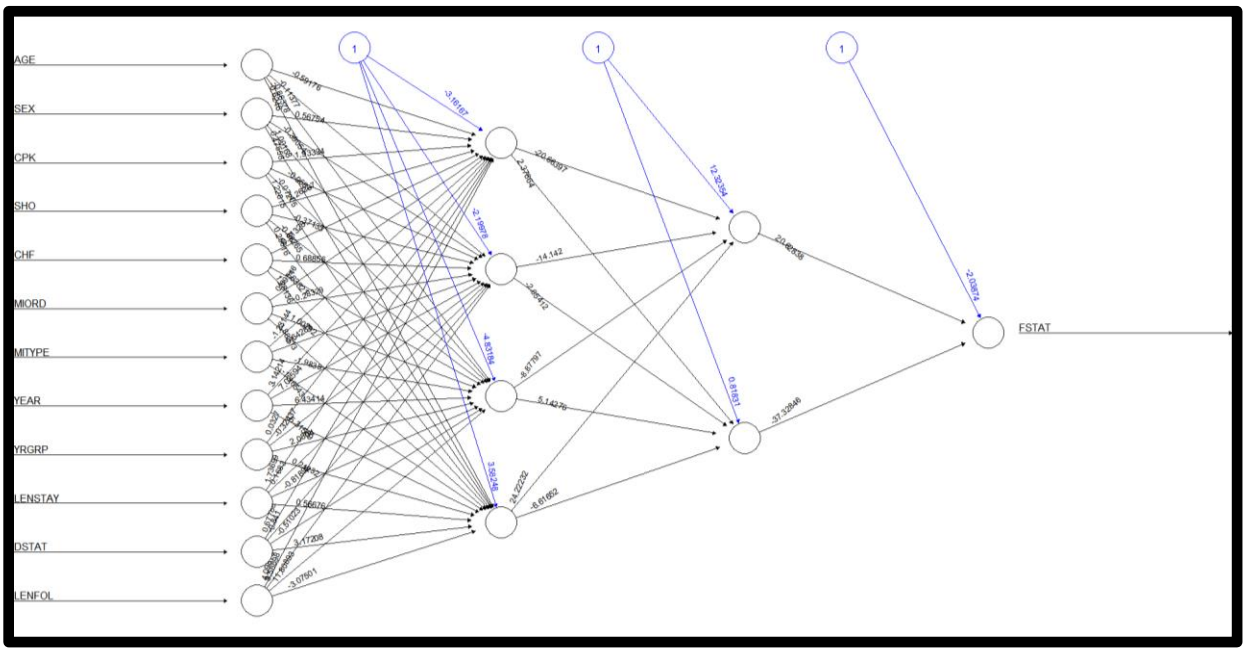

Figure 14. Plot of the Second Neural Network.

```
> mypredict <- compute(nn, nn$covariate)$net.result
> mypredict <- apply(mypredict, c(1), round)
> table(mypredict, train.data$FSTAT, dnn =c("Predicted", "Actual"))
         Actual
Predicted   0   1
        0 156   0
        1   0 159
>
```

Figure 15. Confusion Matrix for Second Neural Network on Training Set.

```
> testPred <- compute(nn, test.data[, 0:12])$net.result
> testPred <- apply(testPred, c(1), round)
> table(testPred, test.data$FSTAT, dnn = c("Predicted", "Actual"))
         Actual
Predicted  0  1
        0 74  6
        1  2 84
>
```

Figure 16. Confusion Matrix for Second Neural Network on Test Set.

```
> nn$net.result[[1]][1:20]
 [1] 0.994140229135 0.994140229135 0.994140229135 0.994140229135 0.994140229135 0.994140229135 0.994140229135
 [8] 0.994140229135 0.994140229135 0.994140229135 0.005884522889 0.005884546938 0.994140229135 0.005884556074
[15] 0.994140229135 0.994140229135 0.994139849845 0.994140229135 0.994140229135 0.994140229135
>
```

Figure 17. First 20 Predictions of the Response Variable for the Third Neural Network.

```
> nn$result.matrix
                                    1
error                   12.352284052243
reached.threshold        0.008594288181
steps                 3621.000000000000
Intercept.to.1layhid1  -18.700777848935
AGE.to.1layhid1          1.167588240470
SEX.to.1layhid1          0.703504702534
CPK.to.1layhid1         -1.539270903027
SHO.to.1layhid1         -0.155265159334
CHF.to.1layhid1          1.381530328906
MIORD.to.1layhid1        0.259205841459
MITYPE.to.1layhid1      -3.556848804777
YEAR.to.1layhid1        30.554554273098
YRGRP.to.1layhid1        1.363667370382
LENSTAY.to.1layhid1     -1.819688487581
DSTAT.to.1layhid1       -3.188736793299
LENFOL.to.1layhid1      35.544983251085
Intercept.to.FSTAT       5.133767771039
1layhid.1.to.FSTAT     -10.263295480250
>
```

Figure 18. Result Matrix of the Third Neural Network.

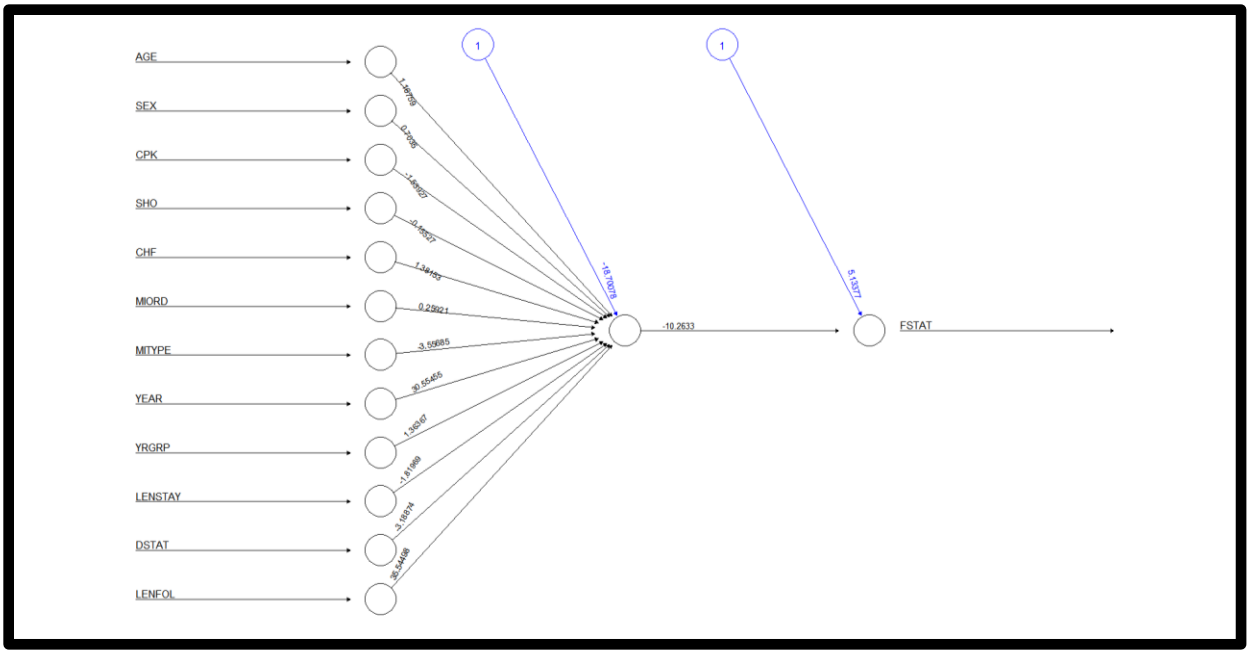

Figure 19. Plot of the Third Neural Network.

```
> mypredict <- compute(nn, nn$covariate)$net.result
> mypredict <- apply(mypredict, c(1), round)
> table(mypredict, train.data$FSTAT, dnn =c("Predicted", "Actual"))
         Actual
Predicted   0   1
        0 155   1
        1   1 158
>
```

Figure 20. Confusion Matrix for Third Neural Network on Training Set.

```
> testPred <- compute(nn, test.data[, 0:12])$net.result
> testPred <- apply(testPred, c(1), round)
> table(testPred, test.data$FSTAT, dnn = c("Predicted", "Actual"))
         Actual
Predicted  0  1
        0 75  5
        1  1 85
>
```

Figure 21. Confusion Matrix for Third Neural Network on Test Set.