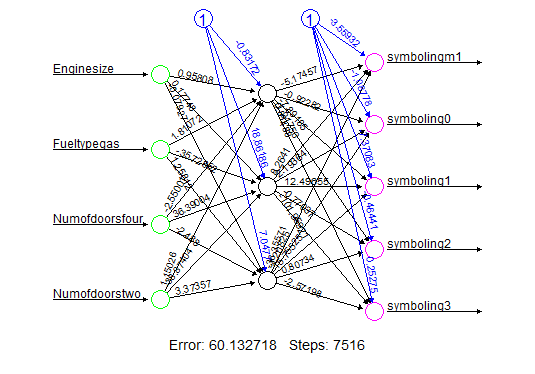
|  |
| --- |
| UMUC |
| Classification Via Multilayer Perceptron |
| Week 5 Exercise |
|  |
| **DBST 667** |
|  |

****

|  |
| --- |
| In this exercise, you will build the multilayer perceptron model to classify the car’s risk level, called symboling, when the mileage per gallon, engine location, car measurements, and number of doors are known. An exercise illustrates the steps for building and visualizing the network, and for evaluating the model accuracy. |

Table of Contents

[Cars Data 2](#_Toc429215928)

[Launch the Program 2](#_Toc429215929)

[Data preprocessing 4](#_Toc429215930)

[Missing Values 4](#_Toc429215931)

[Remove the ID 5](#_Toc429215932)

[Convert symboling into a factor 5](#_Toc429215933)

[Run the Method 5](#_Toc429215934)

[Build the Matrix 6](#_Toc429215935)

[Neuralnet Function 7](#_Toc429215936)

[Network Visualization 13](#_Toc429215937)

[Model Evaluation 14](#_Toc429215938)

[Further Exploration 15](#_Toc429215939)

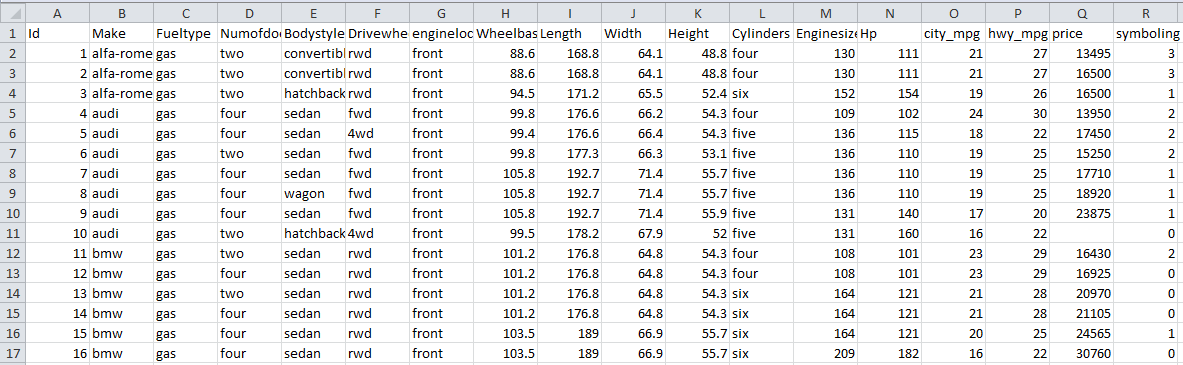
Neural Network Method for Cars Classification

The purpose of this exercise is to build a Multilayer Perceptron network model for classifying the car’s risk level, called symboling, when the mileage per gallon, engine location, car measurements, and number of doors are known. An exercise illustrates the steps for data preprocessing and for building the network. The analyses include the accuracy evaluation of a model and model visualization.

Your results might be slightly different depending on R and R Studio version, and depending on your operating system.

# Cars Data

Figure 1 shows the partial content of the cars.csv file. The column headings in the first row of the file are the cars attribute names called variables. The remaining 205 rows are the data, where each row is a single car record.



Each data row is a car record

Column headers are the variable names

Figure 1: Cars Data

# Launch the Program

Launch R studio program to open an interface on Figure 2.

To run the neural network method, you need to install the neuralnet package if you have not installed it before. Enter the following command into an application console and hit enter.

install.packages("neuralnet")

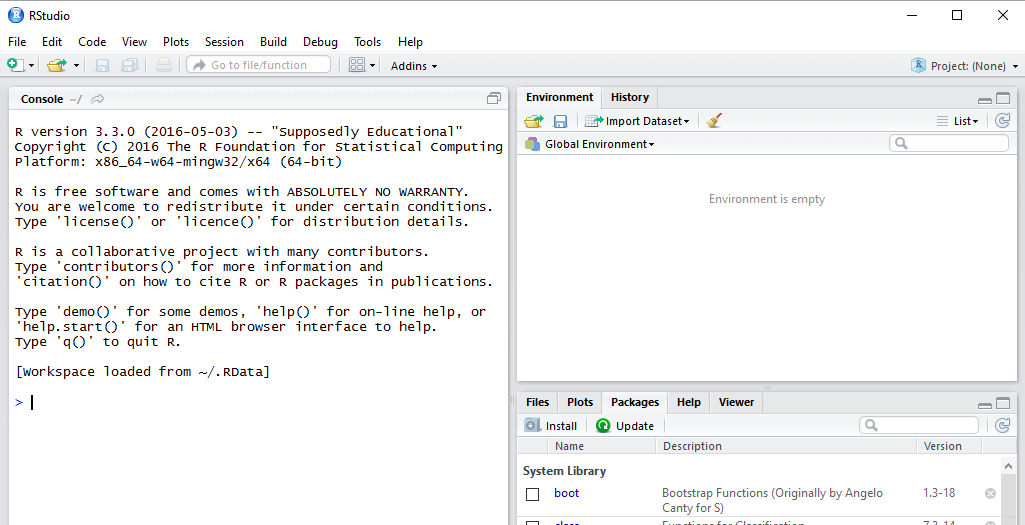


Figure 2: R Studio Interface

Before a package can be used in the current session, it needs to be loaded into memory.

Select the Packages tab at the bottom right window of an interface. Check the checkbox next to neuralnet on Figure 3.

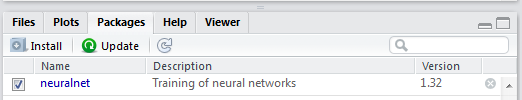


Figure 3: Select the Package to Load

Suppose that the cars.csv file we want to load is in the E:/Datasets folder. To set the working directory to E:/Datasets, enter the following setw command in the console window and hit the enter key. The directory path is specified in parentheses enclosed in double quotes.

setwd("E:/Datasets")

Use **read.csv** command to read the cars file content into a data frame variable called cars. The first input parameter for the read.csv function is the data file name enclosed in double quotes. The second parameter, head=TRUE, specifies that the first row in the file contains the column headers. The sep parameter is the columns delimiter enclosed in double quotes. For example, sep=“,” means that the values in each data row are comma delimited.

The values delimiter

Command to Read from CSV file

cars<-read.csv(file="cars.csv", head=TRUE, sep=",")

Data frame name – stores data from the first sheet in CSV file

Read the column headings from the first row

File Name

Run the head command to preview the first 6 data rows. The head command takes the dataset name and the number of rows to preview as an input. When the number of rows argument is omitted, the first 6 rows are displayed.

head(cars)

Figure 4 shows the variable names followed by the first 6 data rows.

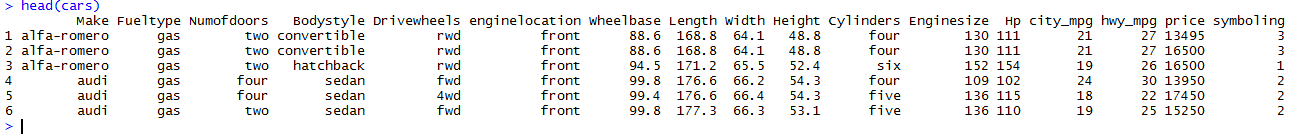
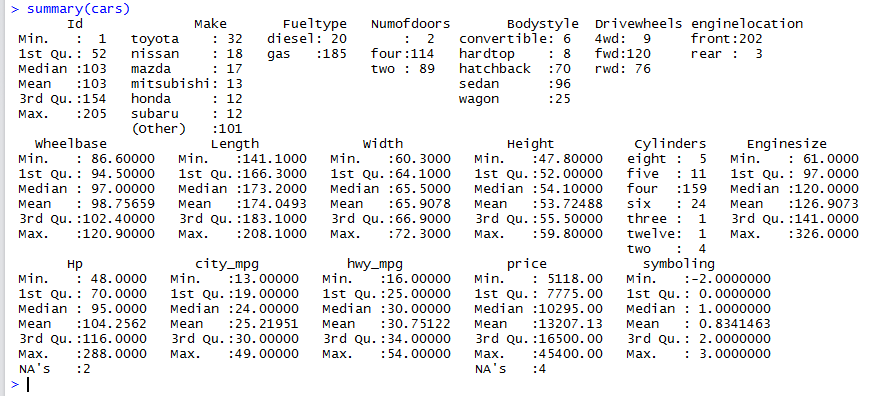


Figure 4: First 6 data Rows

Run the summary command to view the variable statistics, including the missing values.

summary(cars)

Figure 5 shows the descriptive statistics for each variable in cars dataset.



Categorical variables

price has missing values

Hp has missing values

Categorical depended variable was misread as numeric

Unique identifier variable

Figure 5: Cars Variables Statistics

# Data preprocessing

## Missing Values

Since only two data rows are missing the HP value and only four data rows are missing the price, we can replace the missing values with the attribute overall mean value.

The following command finds the rows with missing values for Hp variable and replaces the missing values with the attribute mean. The function is.na on the left hand side returns true when input value is null. The mean function on the right hand side takes the dataset column name as a first input. The second input na.rm=TRUE means to ignoring the missing values in the mean value computation.

cars$Hp[is.na(cars$Hp)]<-mean(cars$Hp, na.rm=TRUE)

To validate that missing values have been handled, run the summary command on Hp variable.

Figure 6 shows the statistics for Hp variable. The statistics no longer show the number of data rows missing Hp value.

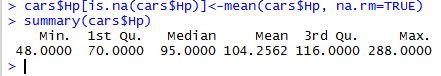


Figure 6: Hp Variable Statistics after Replacing Missing Values

Run the following command to replace the missing price values with an average price. Then run the summary command to verify the missing values have been replaced.

cars$price[is.na(cars$price)]<-mean(cars$price, na.rm=TRUE)

summary (cars$price)

Figure 7showsthe statistics for the price variable. The statistics no longer show the number of data rows missing price value.



Figure 7: Price Variable Statistics after Replacing Missing Values

## Remove the ID

The first variable in the dataset is a unique identifier. The unique identifiers may affect the algorithm results if they are not removed at the data pre-processing stage. In addition, the unique identifier values are irrelevant to the study. For instance, we are not going to classify the cars based on their unique identification number.

NULL needs to be in the upper case.

cars$Id<-NULL

## Convert symboling into a factor

Variable symbolling was read as numeric instead of factor. Run the following factor command to convert the attribute type to factor variable. Then run the summary command to check the variable statistics.

cars$symboling<-factor(cars$symboling)

summary(cars$symboling)

Figure 8 shows the symboling statistics after running the factor function. Instead of minimum value, maximum value, etc., the statistics are the discrete value and the number of instances that have each value.

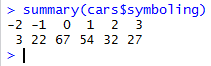


Figure 8: Symboling Statistics after Running factor Function

# Run the Method

The method result in part depends on initial weights assignment to the input variables. The initial weights are assigned randomly, and the assigned initial weights may vary when the method is rerun, even with the same input. Setting the seed value enables reproducing the results when the method is rerun.

Run the following command to set the seed value.

set.seed(1234)

## Build the Matrix

The method takes only numeric variables as an input. In the cars dataset, variables make, fueltype, numofdoors, bodystyle, drivewheels, enginelocation, cylinders, and symboling are categorical variables. We need to build a matrix that has a separate variable for each value except the least dominant value of the categorical variable. The summary command for the variable returns the frequency counts for each value, and we compare the counts to determine the list dominant value.

For example, the possible values of the engine location are front (202 instances) and rear (3 instances). The front value is the dominant value. Hence, the matrix will have a variable called enginelocationfront. If the engine location is front, the variable value will be 1. If the location is back, the variable value will be 0. Note that the variable name is the original variable name suffixed with the value.

The symboling dependent variable can have the values -2, -1, 0, 1, 2, 3, where -2 is the least dominant value. The symboling variable will be changed to 5 variables – symboling-1, symboling0, symboling1, symboling2, and symboling3. We would need to rename symboling-1 because R interprets it as a subtraction operation.

Run the model.matrix command to build the matrix and to store the matrix in a variable matris.train. The command takes a list of variable names in the dataset delimited by a plus sign. The variables can be in any order (symboling+make….). To use all available variables, the wildcard tilde ~ can be used instead of typing all names. The second model.matrix parameter is the dataset name.

matrix.train <- model.matrix(~ ., data=cars)

Run the view command to preview the matrix, including the column headers and data.

View(matrix.train)

Figure 9 shows the partial content of the matrix. The symboling variable has been changed into symboling-1, symboling0, symboling1, symboling2, and symboling3.

If symboling value is 1, symboling1 has a value 1; symboling-1, symboling0, symboling2, and symboling3 have a value 0. If symboling is -2(the value with the lowest frequency), symboling-1, symboling0, symboling1, symboling2, and symboling3 are 0.

Symboling variables

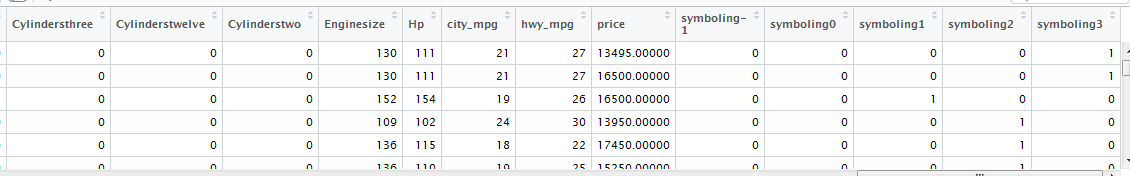


Figure 9: matrix.train

Variable name symboling-1 may be misinterpreted as a subtraction operation. We need to rename the variable, perhaps to symbolingm1. (m sands for minus)

colnames(matrix.train)[colnames(matrix.train)=="symboling-1"] <-"symbolingm1"

Symboling variable renamed

Figure 10 shows that the symboling -1 variable was renamed.

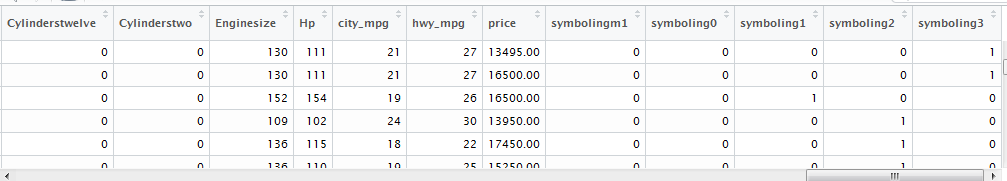


Figure 10: symboling -1 Renamed

## Neuralnet Function

We run the neuralnet function and store the results in a variable called nn. The method input parameters:

**Formula** - specifies the dependent variable on the left hand side and independent variables on the right hand side.

Dependent variable (symboling -1 - 3)

Independent variables

symbolingm1+symboling0+symboling1+symboling2+symboling3

~ city\_mpg+hwy\_mpg+Enginesize +Fueltypegas+Numofdoorsfour+Numofdoorstwo

**Dataset name** – in this case, the dataset is the matrix matrix.train

**Hidden** – the number of hidden layers and the number of nodes in each layer. The parameter input is a vector, and c(3) means 1 hidden layer with 3 nodes.

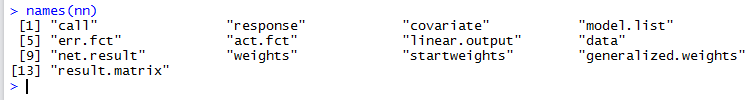
**Linear Output** – Set it to True or T for prediction, and set it to False or F for classification.

nn <- neuralnet(symbolingm1+symboling0+symboling1+symboling2+symboling3 ~ city\_mpg+hwy\_mpg+Enginesize +Fueltypegas+Numofdoorsfour+Numofdoorstwo, data=matrix.train, hidden=c(3), linear.output=F)

Run names command to see the network available properties/attributes on Figure 11.

Entire dataset, even unused variables

names(nn)



Output – fitted values

Used input data

Figure 11: Neural Network Properties

Enter nn at the prompt to view the output stored in nn variable. The output Figure 12 includes the call, numberof repetions, error, and number of steps information. Call is the command we ran to create the model.

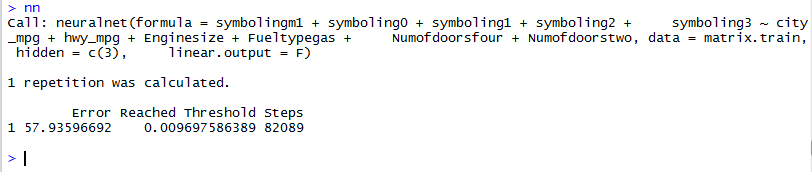


Figure 12: Default Output Sections

To see the remaining output, enter the nn followed by the $ and by the outut property name. For instance, enter the following command to see only the command we ran to generate the model.

nn$call

Figure 13 shows the command we ran to build the model.

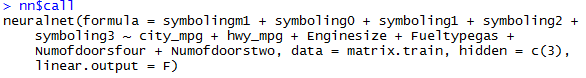


Figure 13: call

To see the actual values of the dependent variable(s), called response, enter the following command.

nn$response

Figure 14 shows the actual dependent variable symboling values for the first 20 data rows. The first column is the instance number, and the remaining columns are the corresponding symboling values from the matrix above. For instance, the actual value in the first and second data rows is 3 because the value of symboling3 is 1.

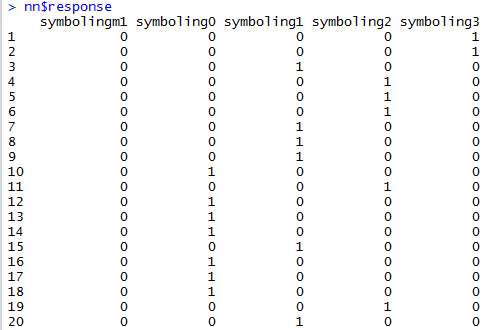
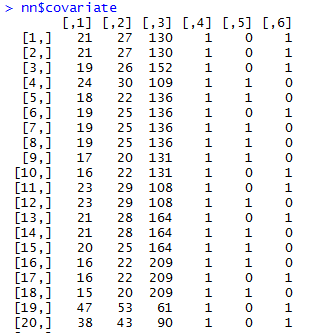


Figure 14: Actual Output Values for First 20 Data Rows

To see the values of the input variables that were used to build the model, run the following command.

nn$covariate

Figure 15 shows the input values for the first 20 instances.



Row number

city\_mpg,

hwy\_mpg,

enginesize,

fueltypegas,

numofdoorsfour,

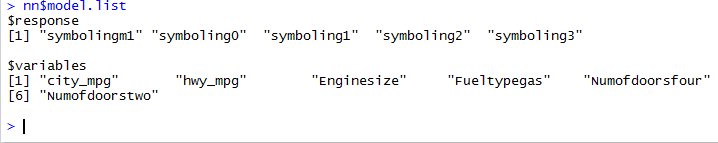
numofdoorstwo

Figure 15: Network Input Values

To get the list of dependent and independent variables in the model, run the following command.

nn$model.list

The response section at the top on Figure 16 is the list of dependent variables (left hand side of the model). The variables section is the list of independent variables (right hand side of the formula)



Independent Variables

Dependent Variables

Figure 16: Dependent and Independent Variables

The linear output parameter specifies if the model is for classification, or for prediction. To check the parameter value, run the following command.

nn$linear.output

The ouput on Figure 17 shows that it is a classification model because the linear output is set to false.



Figure 17: Linear Output Setting

To see the estimated weights, the number of steps, the intercepts, and an error on Figure 18, run the following command.

nn$result.matrix

Training process took 82089 Steps

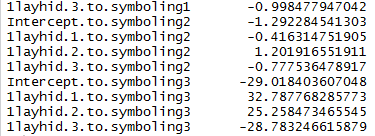
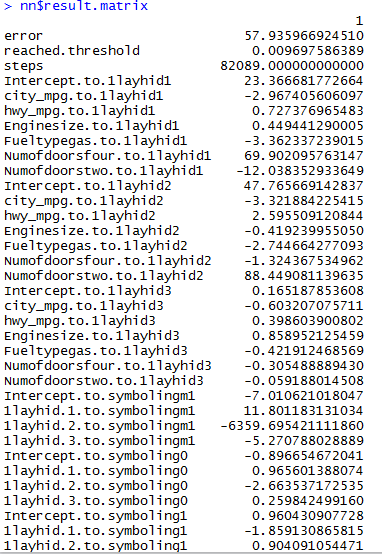


Figure 18: Result Matrix

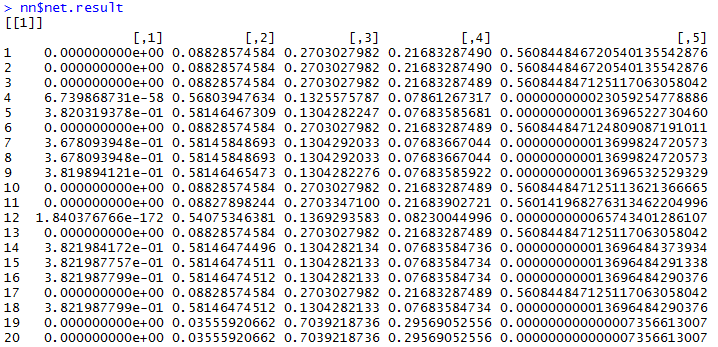
To view the predicted output, run the following command. For each instance, the network output is the probability for each symboling value. The value with the highest probability is the predicted symboling.

nn$net.result

Figure 19 shows the network outputs for the first 20 instances.

* The first column is the instance number.
* The column [, 1] is the value of the symbolingm1 output variable. It is the probability that symboling=-1.
* The column [, 2] is the value of the symboling0 output variable. It is the probability that symboling=0.
* The column [, 3] is the value of the symboling1 output variable. It is the probability that symboling=1.
* The column [, 4] is the value of the symboling2 output variable. It is the probability that symboling=2.
* The column [, 5] is the value of the symboling2 output variable. It is the probability that symboling=3.

The first data row has the highest probability value in the last column [, 5]. Hence, the predicted symboling is 3.



symboling-1 symboling0 symboling1 symboling2 symboling3

Column Index

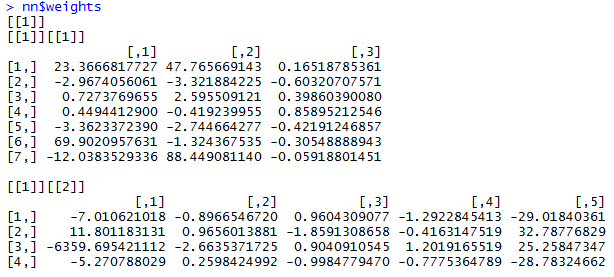
Figure 19: Network Outputs for First 20 Instances

Run the following command to display the network weights after the last method iteration.

nn$weights

The top section on Figure 20 shows the weights from input to the hidden layer nodes, including the weights for the node activation function intercepts. The column heading is the hidden layer node number.

The bottom section shows the weights from the hidden layer into the output layer, including the weights for the output nodes activation function intercepts.



Output layer nodes

Weights from hidden layer to the output layer nodes.

Weights for the output nodes intercepts

Hidden layer nodes

Weights from input layer to the hidden layer nodes.

Weights for the hidden nodes intercepts

Figure 20: Weights After Last Iteration

To see the weights on the first method iteration, run the following command.

nn$startweights

The top section on Figure 21 shows the initial weights from input to the hidden layer nodes, including the weights for the node activation function intercepts. The column heading is the hidden layer node number.

The bottom section shows the weights from the hidden layer into the output layer, including the weights for the output nodes activation function intercepts.

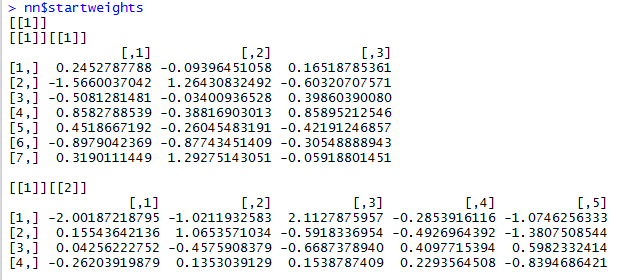


Figure 21: Initial Weights

# Network Visualization

To plot the network, you need to import the plot.nn function from Github. If you do not have the devtools package installed, run the following command to install it.

install.packages("devtools")

Run the library command to load the package into memory.

library(devtools)

Run the command to import the function from Github.

source\_url('<https://gist.githubusercontent.com/fawda123/7471137/raw/466c1474d0a505ff044412703516c34f1a4684a5/nnet_plot_update.r>')

Figure 22 shows an output from source\_url command.



Figure 22: Output from source\_url function

Run the plot.nn function to graph the netowork. You may ignore the warnng message on Figure 23



Figure 23: plot.nn Warning

The plot on Figure 24 shows the input nodes, 3 hidden layer nodes, and the output nodes. Each input node is connected to each hidden layer node, and each connection line has a weight. Each hidden layer node is connected to each output layer node, and each connection line has a weight.

The nodes in the same layer are not connected. The nodes represented as number 1 inside the blue circle are the intercepts for the activation functions. Each intercept is connected to all nodes in the corresponding layer, and each connection has a different weight.

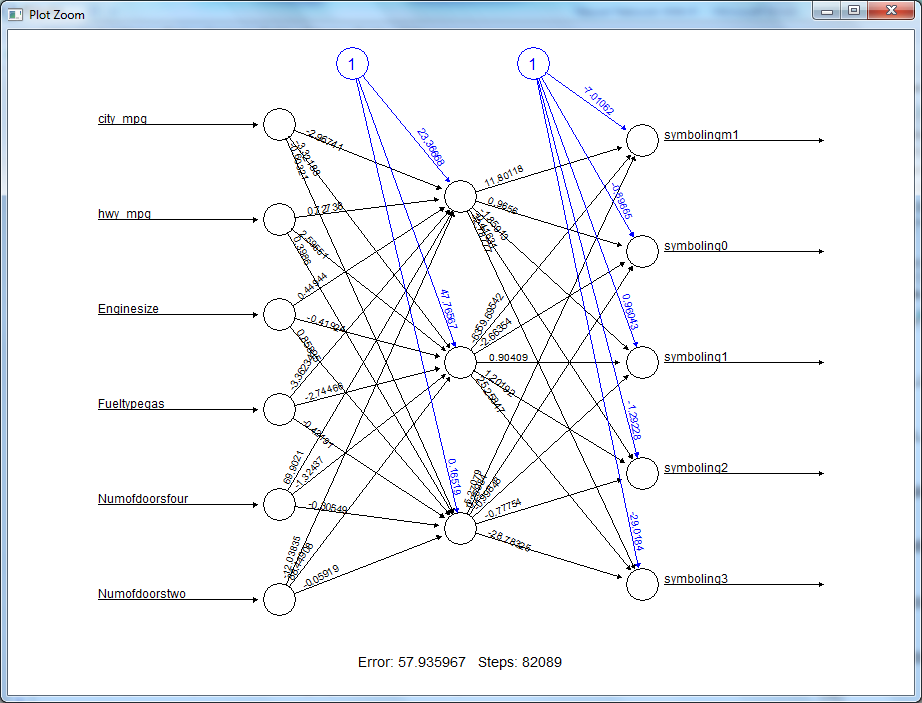


Figure 24: Network Model

# Model Evaluation

To build the confusion matrix, we need to build a confusion matrx that shows how many cars instances have been assigned each class label.

Run the code below to

* store the output probabilities in a variable mypredict
* create function that returns an index of the output variable with the highest probability value
* Store the indexes of the output variable with the highest probability value in the variable idx.
* Map the inx to the simboling value, and store the symboling value into a variable prediction.
* Run the table command to build the cross-tabulation between prediction and the actual symboling value in the pre-processed cars dataset.

mypredict<-compute(nn, nn$covariate)$net.result

maxidx <- function(arr) {

return(which(arr == max(arr)))

}

idx <- apply(mypredict, c(1), maxidx)

The output on Figure 25 shows an index for symbolling value with the highest probability. For example, symboling the index of a symboling value with the highest probability is 5 for the first instance. Index 5 corresponds to symbolyng=3

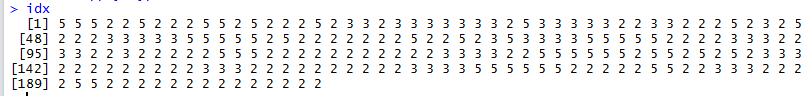


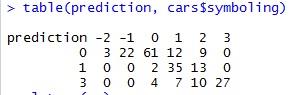
Figure 25: Index of an Output Node with Highest Probability

An array of symboling values

prediction <- c(-1, 0, 1, 2, 3)[idx]

The confusion matrix on Figure 26 shows how many cars are in each predicted class. The predicted classes -2, -1, and 2 are omitted because no data rows had a dominant probability for symboling -1 and 2. Class -2 is omitted because no data rows had dominant probability value less than 1-the probabilities for -1, 0, 1, 2, and 3.

The classification accuracy=(number of instances with actual symboling=predicted symboling)/total number of instances=123/205=60%



Actual symboling=predicted symboling

Predicted symboling

Actual symboling

Figure 26: Confusion Matrix

# Further Exploration

What method parameters can be adjusted to improve the accuracy of a model? Experiment with different values.

How does running the method with different independent variables subset affect the classification accuracy?