Student Name: Brandon Russell

Submission Date: 6/24/18

DBST 667 – Data Mining

Dr. Irene Tsapara

**Week 5 Individual Exercise**

**Deliverables:** Two Files: (1) Submit this lab report with answers to all questions including output screenshots into the ‘Individual Exercises Week 5’ assignment folder. (2) Submit an R script that contains all commands with comments that briefly describe each commands purpose.

**Grading: This exercise is worth 2% of the course grade.** All questions must be answered in your own words with any paraphrased references properly cited using in-text citations and a reference list as needed. In addition, grammatical and spelling errors may affect the grade.

**Part 2** – **Run an exercise on the *vertebral column* dataset from column.csv (note that we are NOT using the credit approval dataset this week), completing this report and providing the commands, output screenshots, and discussion/interpretation as requested. Ensure that all commands are saved in this report AND in an R script.**

**For Reference:** [**UCI Machine Learning Repository: Vertebral Column**](http://archive.ics.uci.edu/ml/datasets/Vertebral+Column)

1. **Introduction:** 
   1. **Identify the dependent variable and independent variables in the Vertebral Column data set.**

The dependent variable is “class”. The independent variables we will use to predict the dependent variable are “pelvic\_incidence”, “pelvic\_tilt”, “lumbar\_lordosis”, “sacral\_slope”, “pelvic\_radius”, and “degree\_spondylolisthesis”.

* 1. **Based on what you have learned this week about neural networks, provide a one-paragraph masters-level response describing what you anticipate that the neuralnet algorithm will accomplish for the Vertebral Column data? Be specific about the behavior and structure of neural network model.**

Based on the reading from this week, I believe the neural network algorithm in R will process the Vertebral Column dataset, stored in a data frame, through a specified number of hidden layers and nodes to determine the dependent variable “class” based on the other 6 independent variables. Based on the calculated weights and intercepts we should be able to apply a value to the 6 input variables and derive an output answer within a certain probability level. For example, if we input 6 input values, the neural network algorithm should move these into the hidden layer nodes based on the assigned weights and intercepts, and then do the same from the hidden layers to the output layer to provide an answer (Günther & Fritsch, 2010).

1. **Data Pre-Processing: Load the Vertebral Column data into R Studio using the read.csv command (do not use File > Import Dataset > From CSV in the R Studio GUI as this uses read\_csv() resulting in significant different variable types).**
   1. **What data pre-processing (if any) does the neuralnet method require for the Vertebral Column data? Include the commands you ran and the output screenshot.**

**Command(s): >**

#Part 2b - Read the data

VertCol <- read.csv("column.csv")

#Look for missing values

apply(VertCol, 2, function (VertCol) sum(is.na(VertCol)))

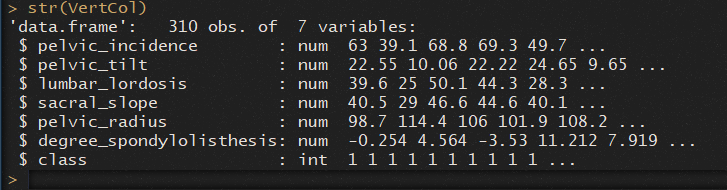
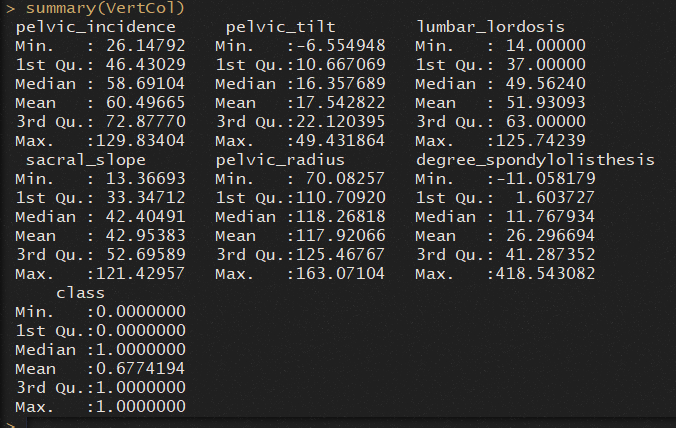
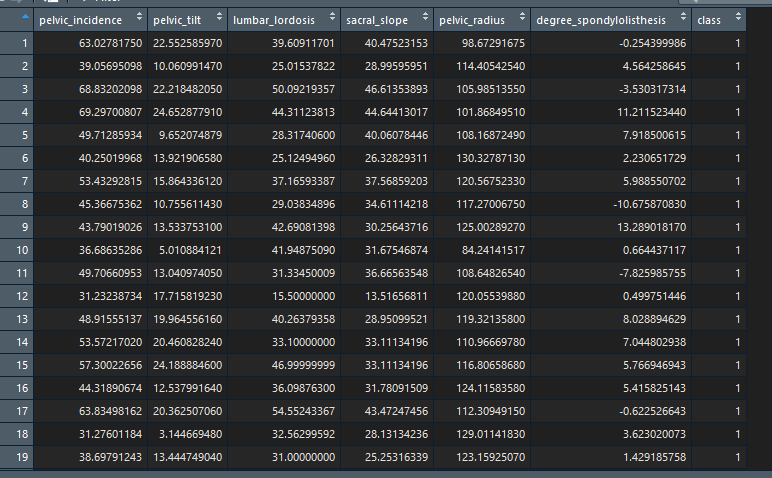
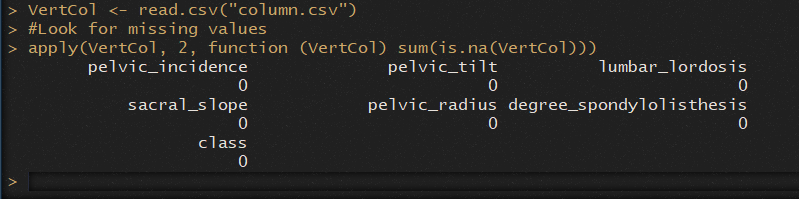
#Examine Data

View(VertCol)

summary(VertCol)

str(VertCol)

**Output:**

****

1. **Neural Network – Running the Method:**
   1. **Run ‘set.seed(12345) and then run the neuralnet() function to build the network storing the results in a variable called ‘nn’ with 2 hidden layers and setting linear.output to TRUE. Include the command and discuss the input parameters you used. *WARNING:*** *When building your neural network, you may notice that it takes a long time to build or periodically may error out with a failure to converge. If this happens, simply run the command again until it works****.***

**Note: Do not shortcut the independent variable list in your formula with a period, as in do not use ‘class~.’ as the formula.**

**Command: >**

#Part 2ci - Create Seed and NeuralNet function

set.seed(12345)

#Creative way of building NN formula

allVars <-colnames(VertCol)

allVars

predictorVars <- allVars[!allVars%in%"class"]

predictorVars

predictorVars <- paste(predictorVars, collapse = "+")

predictorVars

form=as.formula(paste("class~",predictorVars,collapse = "+"))

form

#NN Model

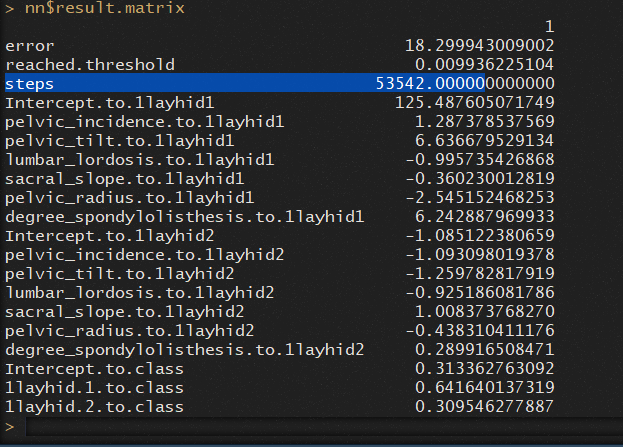
nn<-neuralnet(formula = form, data = VertCol, hidden=2, linear.output = T)

**Discussion:**

In the above command, we first set up our seed to ensure some randomized behavior from the algorithm engine. I used the “allVar”, “predictorVars”, and “form” variables to hold the neural network formula “class~pelvic\_incidence+pelvic\_tilt+lumbar\_lordosis+sacral\_slope+pelvic\_radius+degree\_spondylolisthesis”. Within the neural network function, we first assign the defined formula, which simply means the “class” variable will be predicted based on the other specified values. After that, we assign the dataset that will be used, our “VertCol” data frame. The “hidden” parameter means to have two nodes within the single hidden layer. Making linear output true is necessary when we are using quantitative data, instead of categorical data.

* 1. **Run the command ‘nn$result.matrix’. Include the output screenshot and answer the following questions:**

**Output:**

****

**How many steps were needed to build your neural network?**

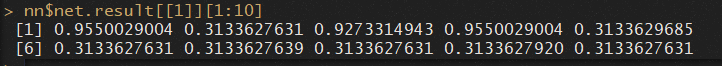
There are 53,542 steps.

**Describe how the relationships between the independent variables and hidden layers are represented? What about the hidden layers to the dependent variable?**

Once we look past the error, reached threshold, and steps we will see the weights for the independent variables going into the hidden layers. Each variable is listed for each hidden layer node with a corresponding weight. For example, the “pelvic\_incidence” variable has a weight of approximately 1.29 to the first hidden node, and approximately -1.09 to node 2. After the weights for each hidden node is specified, the weights for the hidden nodes to the dependent variable “class” are specified. In this case, hidden node 1 has a weight of approximately 0.64 and node 2 has an approximate weight of 0.31.

* 1. **Run the command ‘nn$net.result[[1]][1:10]’. Include the output screenshot and answer the following question:**

**Output:**

****

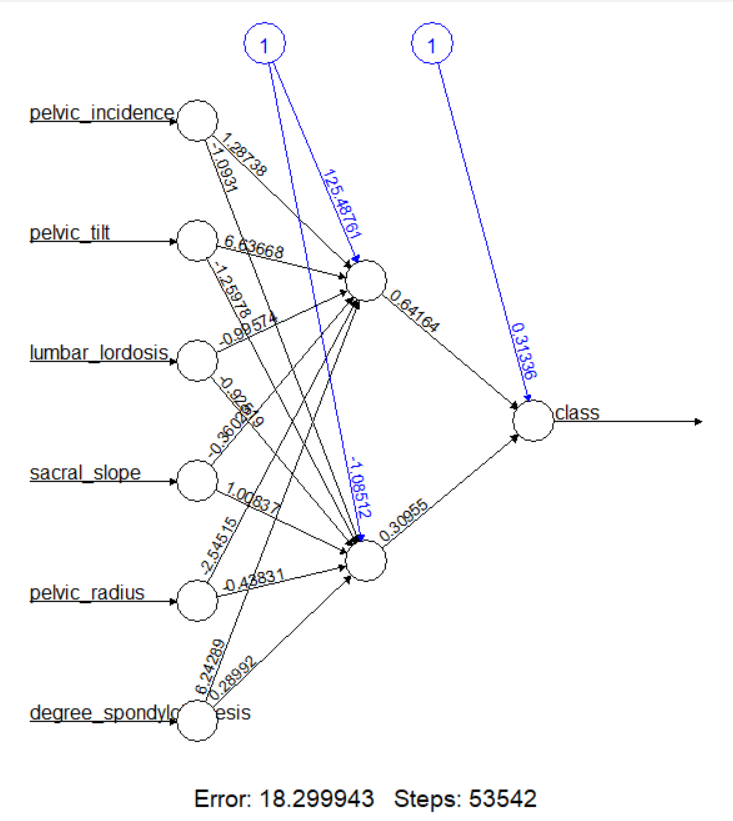
**What do each of the ten results above represent? Relate your answer back to the classification of patients from the Vertebral Column dataset. (Hint: patients diagnosed as normal are 0 and those with either disk hernia or spondylolisthesis are 1.)**

The command above prints the predicted probability that the specified row in the dataset has a class of 0 or 1. For example, row 1 has approximately a 96% chance of being a 1, while row 10 only has approximately a 31% chance.

1. **Neural Network - Visualization:** 
   1. **Run the plot() command on your neural network ‘nn’. In the space below provide a zoomed screenshot of the plot ONLY so that it is completely visible and all components legible. (Hint: Use the Plots tab Zoom). Include your command, the output screenshot of your plot, and a one-paragraph, masters-level interpretation of all visible aspects of your neural network.**

**Command: > plot(nn)**

**Output:**

****

**Interpretation:**

The plot diagram above shows an abstract view of how our neural network is making decisions on predicting the class variable based on the other 6 independent variables. As we can see from the diagram, this process took 53,542 steps with an error of approximately 18. The backpropagation algorithm is used here to determine the intercepts and weights for each node. In this instance, for example, we can see from the dependent variable to hidden node 2 there is an intercept value of 0.31 and a weight of 0.31. From hidden node 2 to the “pelvic\_radius” independent variable there is a weight of -0.44 with an intercept of -1.086 (Günther & Fritsch, 2010).

1. **Neural Network – Evaluate the Model:**
   1. **Run the two-step model evaluation process from the tutorial providing those two commands and the command to display the first 10 values from the variable ‘mypredict’ that you create. Include all three commands and the output screenshot.**

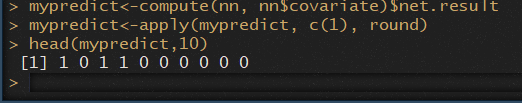
**Commands: >**

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

mypredict<-apply(mypredict, c(1), round)

head(mypredict,10)

**Output:**

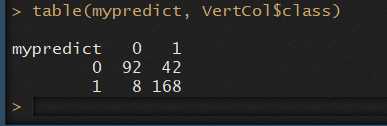
****

* 1. **Run the table command to build the confusion matrix using ‘mypredict’ as the first argument and the Vertebral Column dataset dependent variable as the second. Include the command, output screenshot of your matrix, and answer the following question:**

**Commands: >**

table(mypredict, VertCol$class)

**Output:**

****

**What is the classification accuracy of your neural network? Provide the complete formula used (i.e. show your work) along with the final percentage (rounded to two decimals places)**

Total correctly classified instances = (92+168) = 260

Total instances classified = (92+168+42+8) = 310

Accuracy = 260/310 = 0.8387 = 0.84 = 84%

1. **Neural Network – Running the Method Once More with Different Inputs:**
   1. **Repeat the steps from 2.c, 2.d, and 2.e (Running the Method, Visualization, and Evaluate the Model) but using a different combination of input parameters. At a minimum, you need to change the number of hidden layers and the number of nodes in the hidden layers. Explore the available customizations by reading help(neuralnet) to improve the accuracy of your model. (Hint: Use a vector for hidden = c(x,y) where x and y are the number of hidden layers and nodes.)**

**All commands from the steps listed must be included in the command block below. You are free to work with and modify your commands in R Studio prior to putting your final set here that ultimately show model improvement.**

**The only output required is the plot of the neural network.**

**Commands: >**

#Part 2 fi - Report c,d,e w/ new parameters

#New NN Model

nn.new<-neuralnet(formula = form, data = VertCol, hidden=c(4,2), linear.output = T)

nn.new

#Part 2 fi - cii - Results Matrix

nn.new$result.matrix

#Part 2 fi - ciii - Result Output

nn.new$net.result[[1]][1:10]

#Part 2 fi - d - Plot the Neural Network

plot(nn.new)

#Part 2 fi - ei - Eval the model

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

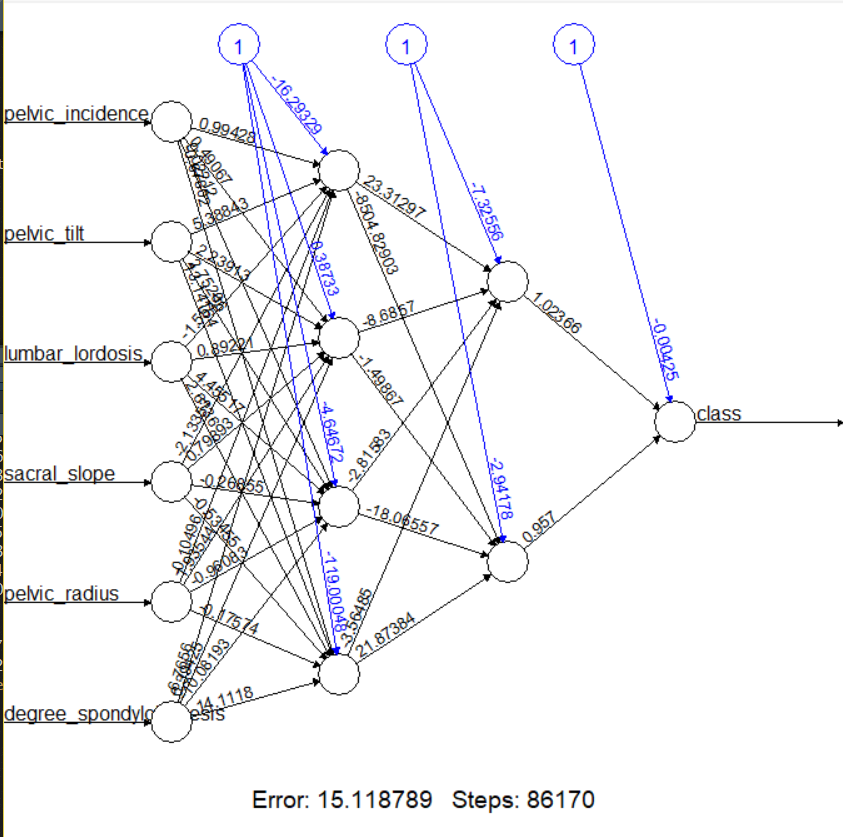
mypredict<-apply(mypredict, c(1), round)

head(mypredict,10)

# Part 2 fi - eii -confusion matrix

table(mypredict, VertCol$class)

**Output:**

****

* 1. **What is the classification accuracy for the new neural network you just built? Provide the complete formula used (i.e. show your work) along with the final percentage (rounded to two decimals places).**

Total correctly classified instances = (86+183) = 269

Total instances classified = (86+183+14+27) = 310

Accuracy = 269/310 = 0.8677 = 0.87 = 87%

* 1. **Compare the classification accuracy from the first neural network run to your final second run. Which has a higher classification accuracy? Provide a one-paragraph, masters-level response that provides reasonable justification for why one is higher than the other. The demonstration of your analysis and depth of understanding are being evaluated above simple right or wrong answers.**

In the first neural network run, we came up with 260 correctly identified instances, giving us an 84% accuracy in prediction. After increasing the number of hidden layers to two and total number of nodes to 6 in the new neural network function, we saw an overall increase of 269 correctly identified instances giving us an overall 87% accuracy in predicting class. One reason for the increase in accuracy can be directly correlated with the increase in nodes within the hidden section of the neural network. For example, the number of steps increased from 53,542 with an error of 18.3 to 86,170 steps with an error of 15.1. The increase in steps across additional nodes allows the algorithm to process different combinations of the independent variables to determine the most accurate combination for predicting the dependent variable. Looking at the new plot, we can see there is definitely higher value weights associated with many of the nodes than from the previous plot with only two nodes. Providing more nodes allows us to process more specific associations, leading to higher weight and intercept values, and ultimately leading to more accurate results (Han, Kamber, & Pei, 2011).

1. **Summary:**
   1. **What differences did you observe between the function and behavior of decision tree classification and neural network classification? Support your observations with external research and provide a two-paragraph, masters-level response.**

After running through the exercises for week 4 and week 5, there are clear differences between neural networks and decision trees, although there are some similarities at the surface-level. The first thing I noticed was how similar they looked when plotted out. Decision trees have a top-down look and neural networks work from left to right. They are also both supervised learning algorithms. The differences I noticed were how the results were derived, and the purpose of each algorithm. When running the decision tree algorithm and observing the plot I noted how each step was accomplished. For example, I could see where and why a split occurred on the specified criterion. With neural networks, however, while you can denote weights and intercepts it is not clear as to how these values were derived. Through observation alone, I would state decision trees could handle categorical data better, while neural networks could handle quantitative data better.

When decision trees need to decide, they are attempting to determine the best splitting criterion based on the values of an attribute. For example, if we have an income attribute the algorithm might decide to split at a certain value within the incomes where a decision needs to be made on approving a credit application. This same decision could be made in a neural network; however, it would not be obvious in a plot on how the decision was made. With a decision tree we could see, for example, those with an income of less than value “x” will probably not be approved, but in a neural network we would only see that the income attribute was assigned a weight of ‘a’ going into hidden node 1, and then a decision is declared in the output node (Han, Kamber, & Pei, 2011). Additionally, decision trees are well suited for handling multiple categorical classification problem sets, while neural networks are better suited for binary categorical decisions (Decision Trees Compared to Regression and Neural Networks, n.d.).

* 1. **(Not graded) Which part of this exercise did you find the most challenging and what steps did you take to resolve the challenge?**

**I had a difficult time figuring out how the weights and intercepts were used in the neural networks. However, I found a series of YouTube videos that explained the topic well.**

References

*Decision Trees Compared to Regression and Neural Networks*. (n.d.). Retrieved from DTREG: https://www.dtreg.com/methodology/view/decision-trees-compared-to-regression-and-neural-networks

Günther, F., & Fritsch, S. (2010). neuralnet: Training of neural networks. *The R journal, 2*, 30-38.

Han, J., Kamber, M., & Pei, J. (2011). *Data mining: concepts and techniques.* Elsevier.