T2\_DeLosSantos

Jonathan De Los Santos

3/9/2021

# Question 1

## Prompt

Consider the below simple bank data on consumers’ use of credit cared credit facilities. Assume that there is a small network that has three types of layers; input, hidden, and output. The network has a hidden layer with two nodes. Illustrate one pass of the network for the first observation through a simple neural network. In creating a network, the hidden layer consists of two nodes, each receiving input from the input nodes. To compute the output of a hidden layer nodes, you need to initiate the weights for the input nodes, and ones for the hidden nodes. The initialized values should be small, usually random, typically in the range 0.00 0.05.

Table

Description automatically generated

With the weights generated below, we sum the products of each input and its weight to arrive at the hidden nodes. The same operation is repeated for the output node, treating the hidden nodes as inputs.

For the model below we perform the following calculations:

Hidden Nodes

(4\*0.003) + (43\*0.0015) + (0\*0.007) = 0.0765

(4\*0.02) + (43\*0.012) + (0\*0.008) = 0.596

Output Node

(0.0765\*0.017) + (0.596\*0.011) = 0.0079

Diagram

Description automatically generated

# Question 2

## Prompt

The dataset was collected by the National Institute of Diabetes and Digestive and Kidney Diseases. The objective of the dataset is to diagnostically predict whether or not a patient has diabetes (1-Yes/0-No) based on certain diagnostic measurements included in the dataset.

Your task is to examine and predict those individuals that have diabetes or the risk factors that could lead to diabetes in this sample using SVM classification techniques. The data can be divided into training (70%) and testing dataset (30%). Develop a model using a different “kernel” algorithm, and interpret the best model.

The dataset was collected by the National Institute of Diabetes and Digestive and Kidney Diseases: pima2.csv. The objective of the dataset is to diagnostically predict whether a patient has diabetes (1-Yes/0-No) based on certain diagnostic measurements included in the dataset. Your task is to examine and predict those individuals that have diabetes or the risk factors that could lead to diabetes in this sample using SVM classification techniques. The data can be divided into training (70%) and testing (30%) datasets. Develop a model using a different “kernel” algorithm and interpret the best model.

## Data Prep

Begin by reading in the data and examining it. We immediately notice that type2 is likely a factor that needs to be converted. We will test the integer values for normalization to see if it’s necessary.

pima <- read.csv("pima2.csv")  
str(pima)

## 'data.frame': 200 obs. of 8 variables:  
## $ npreg: int 5 7 5 0 0 5 3 1 3 2 ...  
## $ glu : int 86 195 77 165 107 97 83 193 142 128 ...  
## $ bp : int 68 70 82 76 60 76 58 50 80 78 ...  
## $ skin : int 28 33 41 43 25 27 31 16 15 37 ...  
## $ bmi : num 30.2 25.1 35.8 47.9 26.4 35.6 34.3 25.9 32.4 43.3 ...  
## $ ped : num 0.364 0.163 0.156 0.259 0.133 ...  
## $ age : int 24 55 35 26 23 52 25 24 63 31 ...  
## $ type2: int 0 1 0 0 0 1 0 0 0 1 ...

#summary(pima$npreg)

### Factor Conversion

pima$type2 <- as.factor(pima$type2)  
str(pima)

## 'data.frame': 200 obs. of 8 variables:  
## $ npreg: int 5 7 5 0 0 5 3 1 3 2 ...  
## $ glu : int 86 195 77 165 107 97 83 193 142 128 ...  
## $ bp : int 68 70 82 76 60 76 58 50 80 78 ...  
## $ skin : int 28 33 41 43 25 27 31 16 15 37 ...  
## $ bmi : num 30.2 25.1 35.8 47.9 26.4 35.6 34.3 25.9 32.4 43.3 ...  
## $ ped : num 0.364 0.163 0.156 0.259 0.133 ...  
## $ age : int 24 55 35 26 23 52 25 24 63 31 ...  
## $ type2: Factor w/ 2 levels "0","1": 1 2 1 1 1 2 1 1 1 2 ...

### Test Distributions

To test multiple values at once, we create a dataframe without our target variable and leave just the numeric data. Then we apply the Shapiro-Wilks Normality Test to the remaining columns.

The output is hidden due to length, but normality is rejected for the following columns:

* npreg
* glu
* skin
* ped
* age

Based on this, we will go ahead and apply normalization to the entire data set.

### Normalization

Now we scale our data by creating a scaling function and applying it to the numeric variables in our data set. For consistency, I decided to scale all variables even if they were alreaduy normally distributed.

# Normaliztion Function  
  
scale.variable <- function(df, variables){  
 for (i in variables){  
 df[[i]] <- scale(df[[i]], center = T, scale = T)  
 }  
 return(df)  
}  
  
to.normalize <- c("npreg",  
 "glu",  
 "bp",  
 "skin",  
 "bmi",  
 "ped",  
 "age")  
  
n.pima <- scale.variable(pima, to.normalize)  
  
summary(n.pima)

## npreg.V1 glu.V1 bp.V1   
## Min. :-1.0605217 Min. :-2.1463832 Min. :-2.897313   
## 1st Qu.:-0.7634568 1st Qu.:-0.7569340 1st Qu.:-0.632426   
## Median :-0.4663919 Median :-0.1095770 Median :-0.109760   
## Mean : 0.0000000 Mean : 0.0000000 Mean : 0.000000   
## 3rd Qu.: 0.7218677 3rd Qu.: 0.6325152 3rd Qu.: 0.587128   
## Max. : 3.0983869 Max. : 2.3693266 Max. : 3.374681   
## skin.V1 bmi.V1 ped.V1   
## Min. :-1.894735 Min. :-2.3017149 Min. :-1.223094   
## 1st Qu.:-0.721987 1st Qu.:-0.7724040 1st Qu.:-0.674636   
## Median :-0.018338 Median : 0.0799320 Median :-0.287298   
## Mean : 0.000000 Mean : 0.0000000 Mean : 0.000000   
## 3rd Qu.: 0.578698 3rd Qu.: 0.6835000 3rd Qu.: 0.505281   
## Max. : 5.952018 Max. : 2.5431421 Max. : 5.947550   
## age.V1 type2   
## Min. :-1.0122604 0:132   
## 1st Qu.:-0.8300353 1: 68   
## Median :-0.3744726   
## Mean : 0.0000000   
## 3rd Qu.: 0.6505436   
## Max. : 2.8144667

### Partition Data

Using the caret package we partition our data, using 70% for training.

library(caret)  
partition <- createDataPartition(y = n.pima$type2, p = 0.7, list = FALSE)  
train <- n.pima[partition,]  
test <- n.pima[-partition,]   
  
str(train)

## 'data.frame': 141 obs. of 8 variables:  
## $ npreg: num [1:141, 1] 0.425 -1.061 -1.061 0.425 -0.169 ...  
## $ glu : num [1:141, 1] -1.483 1.296 -0.536 -0.852 -1.294 ...  
## $ bp : num [1:141, 1] 0.936 0.413 -0.981 0.413 -1.155 ...  
## $ skin : num [1:141, 1] 1.005 1.176 -0.36 -0.189 0.152 ...  
## $ bmi : num [1:141, 1] 0.569 2.543 -0.964 0.537 0.325 ...  
## $ ped : num [1:141, 1] -0.992 -0.657 -1.067 -0.269 -0.406 ...  
## $ age : num [1:141, 1] 0.263 -0.557 -0.83 1.812 -0.648 ...  
## $ type2: Factor w/ 2 levels "0","1": 1 1 1 2 1 1 1 1 2 1 ...

## A. Report SVM with kernel radial classification

(Training error, misclassification, accuracy, 95% confidence interval) (10 pts)

First build the radial SVM model which is the default. We notice it generates 88 support vectors, this can be evaluated below.

library(e1071)  
  
rbfSVM <- svm(formula = type2~., data = train)  
  
summary(rbfSVM)

##   
## Call:  
## svm(formula = type2 ~ ., data = train)  
##   
##   
## Parameters:  
## SVM-Type: C-classification   
## SVM-Kernel: radial   
## cost: 1   
##   
## Number of Support Vectors: 94  
##   
## ( 52 42 )  
##   
##   
## Number of Classes: 2   
##   
## Levels:   
## 0 1

### SVM Model Evaluation

The radial SVM model has a success rate of 71.1%. an error rate of 28.8%, and misclassified 17 predictions.

pred <- predict (rbfSVM, test)  
table(pred, test$type2)

##   
## pred 0 1  
## 0 31 10  
## 1 8 10

agreement <- pred == test$type2  
table(agreement)

## agreement  
## FALSE TRUE   
## 18 41

prop.table(table(agreement))

## agreement  
## FALSE TRUE   
## 0.3050847 0.6949153

## B. Report ksvm with kernel vanilladot classification

(Training error, misclassification, accuracy, 95% confidence interval) (10 pts)

Next build the linear SVM model with kernel = "vanilladot". This generates 70 support vectors and has a training error of 0.21.

library(kernlab)

## Warning: package 'kernlab' was built under R version 4.0.2

##   
## Attaching package: 'kernlab'

## The following object is masked from 'package:ggplot2':  
##   
## alpha

library(caret)  
  
vSVM <- ksvm(type2~., data = train,  
 kernel = "vanilladot")

## Setting default kernel parameters

# Print output  
vSVM

## Support Vector Machine object of class "ksvm"   
##   
## SV type: C-svc (classification)   
## parameter : cost C = 1   
##   
## Linear (vanilla) kernel function.   
##   
## Number of Support Vectors : 73   
##   
## Objective Function Value : -70.5342   
## Training error : 0.219858

### Linear SVM Evaluation

The linear SVM model has a success rate of 74.6%, an error rate of 25.4%, and misclassified 15 predictions. This performed slightly better than the default radial kernel.

pred.v <- predict (vSVM, test)  
table(pred.v, test$type2)

##   
## pred.v 0 1  
## 0 32 8  
## 1 7 12

agreement.v <- pred.v == test$type2  
table(agreement.v)

## agreement.v  
## FALSE TRUE   
## 15 44

prop.table(table(agreement.v))

## agreement.v  
## FALSE TRUE   
## 0.2542373 0.7457627

## C. Report ksvm with rbfdot Classification

(Training error, misclassification, accuracy, 95% confidence interval) (10 pts)

Next build the RBFdot SVM model with kernel = "rbfdot". This generates 92 support vectors with a training error of 0.15.

library(kernlab)  
library(caret)  
  
rbfSVM <- ksvm(type2~., data = train,  
 kernel = "rbf")  
  
# Print output  
rbfSVM

## Support Vector Machine object of class "ksvm"   
##   
## SV type: C-svc (classification)   
## parameter : cost C = 1   
##   
## Gaussian Radial Basis kernel function.   
## Hyperparameter : sigma = 0.122470070628455   
##   
## Number of Support Vectors : 91   
##   
## Objective Function Value : -63.03   
## Training error : 0.156028

### RBFdot SVM Evaluation

The RBFdot SVM model has a success rate of 69.5%, an error rate of 30.5%, and misclassified 18 predictions. This performed worse than both the default radial and linear kernels.

pred.rbf <- predict(rbfSVM, test)  
table(pred.rbf, test$type2)

##   
## pred.rbf 0 1  
## 0 31 10  
## 1 8 10

agreement.rbf <- pred.rbf == test$type2  
table(agreement.rbf)

## agreement.rbf  
## FALSE TRUE   
## 18 41

prop.table(table(agreement.rbf))

## agreement.rbf  
## FALSE TRUE   
## 0.3050847 0.6949153

## D. Which kernel is preferred?

Based on the evaluations above, the linear SVM (vanilladot) kernel evaluated as the most accurate when compared with the test data.

# Question 3

## Prompt

The BC3 text file includes news related to Blockchain technologies, and the DS text file contains news regarding Ebola issues. The preprocessing for text files are critical. Notably, in this assignment, you are requested to conduct not only mining text files but also text classification. Therefore, the preprocessing becomes much more critical to complete the below two tasks. Please carefully read the directions and answer the questions.

This test requires to conduct text classification using CART classification. Please conduct data pre-processing. It is recommended that you use what you did in the Module 7.

## Data Pre-Processing

Load the required data-cleaning packages:

library(tm)  
library(stringr)  
library(Matrix)  
library(tidytext)  
library(topicmodels)

### Create Initial Corpora

Create two corpora from the sample texts:

textDS = file("DS.txt", open = "r")  
textDS.decomposition = readLines(textDS, encoding="UTF-8")  
corpusDS <- Corpus(VectorSource(textDS.decomposition))  
#orpusDS  
  
textBC = file("BC3.txt", open = "r")  
textBC.decomposition = readLines(textBC, encoding="UTF-8")  
corpusBC3 <- Corpus(VectorSource(textBC.decomposition))

### Data Cleaning

Perform the standard cleaning steps on both corpus objects. I have selected stop words for each corpus for words that are not useful to our analysis.

# Basic cleaning  
corpusDS <- tm\_map(corpusDS, tolower)

## Warning in tm\_map.SimpleCorpus(corpusDS, tolower): transformation drops  
## documents

corpusDS <- tm\_map(corpusDS, removeNumbers)

## Warning in tm\_map.SimpleCorpus(corpusDS, removeNumbers): transformation drops  
## documents

corpusDS <- tm\_map(corpusDS, removePunctuation)

## Warning in tm\_map.SimpleCorpus(corpusDS, removePunctuation): transformation  
## drops documents

# Stopwords  
  
selfstopwordsDS <- c("Ebola", "UN", "WHO", "Virus")  
  
corpusDS<- tm\_map(corpusDS, removeWords, c(stopwords("en"), selfstopwordsDS))

## Warning in tm\_map.SimpleCorpus(corpusDS, removeWords, c(stopwords("en"), :  
## transformation drops documents

# Strip Whitespace  
corpusDS <- tm\_map(corpusDS, stripWhitespace)

## Warning in tm\_map.SimpleCorpus(corpusDS, stripWhitespace): transformation drops  
## documents

# Stemming  
corpusDS <- tm\_map(corpusDS, stemDocument)

## Warning in tm\_map.SimpleCorpus(corpusDS, stemDocument): transformation drops  
## documents

# Basic cleaning  
corpusBC3 <- tm\_map(corpusBC3, tolower)

## Warning in tm\_map.SimpleCorpus(corpusBC3, tolower): transformation drops  
## documents

corpusBC3 <- tm\_map(corpusBC3, removeNumbers)

## Warning in tm\_map.SimpleCorpus(corpusBC3, removeNumbers): transformation drops  
## documents

corpusBC3 <- tm\_map(corpusBC3, removePunctuation)

## Warning in tm\_map.SimpleCorpus(corpusBC3, removePunctuation): transformation  
## drops documents

# Stopwords  
  
selfstopwordsBC3 <- c("crypto", "cryptocurrency", "blockchain", "technology")  
  
# Strip Whitespace  
corpusBC3 <- tm\_map(corpusBC3, stripWhitespace)

## Warning in tm\_map.SimpleCorpus(corpusBC3, stripWhitespace): transformation drops  
## documents

# Stemming  
corpusBC3 <- tm\_map(corpusBC3, stemDocument)

## Warning in tm\_map.SimpleCorpus(corpusBC3, stemDocument): transformation drops  
## documents

### Create Document-Term Matrices

Now create a document-term matrix for each corpus and combine it into one master DTM with both corpora.

# Create DS DTM  
  
corpusDS.dtm <- DocumentTermMatrix(corpusDS)  
  
# Create BC DTM  
  
corpusBC.dtm <- DocumentTermMatrix(corpusBC3)  
  
# Combine matrices  
  
corpus.dtm <- c(corpusDS.dtm, corpusBC.dtm)  
corpus.dtm

## <<DocumentTermMatrix (documents: 66, terms: 1034)>>  
## Non-/sparse entries: 2118/66126  
## Sparsity : 97%  
## Maximal term length: 18  
## Weighting : term frequency (tf)

## A. Convert the document-term matrix to create a data frame for decision tree analysis

After a lot of overthinking this issue, the DTM above was successfully converted to a data frame.

df.dtm <- as.data.frame(as.matrix(corpus.dtm))

## B. Text Classification

In the document-term data frame, the first 33 documents are related to “Blockchain” technologies, and the next 33 documents (34th – 66th) are related to “Ebola” disease. For classification analysis, add the “news” to the data frame. For example, if your data frame is named, “dfnew,” your r code may appear as follows:

### Data Frame Conversion

To perform our text classification, the news column was added to our document-term data frame. The printout below shows that the values have been assigned as expected. In this case, the first half of the dataframe documents are the Ebola text.

newsDF.dtm <- df.dtm  
newsDF.dtm[,"news"] <- "Ebola"  
newsDF.dtm$news[34:66] <- "Blockchain"  
  
newsDF.dtm$news <- as.factor(newsDF.dtm$news)  
#head(newsDF.dtm)

Your data frame contains so many terms from Q1. Please consider 5 to 10 of the most frequent terms for the decision tree analysis. One of the possible functions to retrieve frequent terms is findFreqTerms (for example, findFreqTerms(“your document term matrix”, # of terms). You can also manually figure out the appropriate frequent terms using Excel.

### Selecting Terms

To focus our decision tree on a few terms, we’ll use findFreqTerms() with a minimum boundary of 10 appearances. Based on the output below (and noticing some terms should have been stop words), I have selected the following:

* Health
* Record
* Trust
* Media
* Public
* Document

findFreqTerms(corpus.dtm, lowfreq = 5)

## [1] "ebola" "emerg" "govern" "health" "intern"   
## [6] "travel" "world" "case" "earli" "guinea"   
## [11] "histori" "agenc" "mani" "said" "¡°"   
## [16] "data" "exampl" "new" "privat" "worker"   
## [21] "also" "public" "report" "spread" "contain"   
## [26] "countri" "global" "includ" "provid" "respons"   
## [31] "set" "support" "origin" "medic" "nation"   
## [36] "communiti" "time" "cuba" "peopl" "state"   
## [41] "help" "year" "million" "need" "trust"   
## [46] "secur" "can" "get" "media" "use"   
## [51] "without" "now" "number" "one" "will"   
## [56] "about" "and" "blockchain" "but" "for"   
## [61] "like" "point" "technolog" "that" "the"   
## [66] "there" "this" "was" "were" "where"   
## [71] "while" "compani" "have" "all" "are"   
## [76] "document" "it¡¯" "more" "most" "other"   
## [81] "person" "share" "they" "track" "way"   
## [86] "with" "busi" "digit" "reli" "verifi"   
## [91] "record" "when" "babi" "problem" "these"   
## [96] "down" "such" "which" "our" "system"   
## [101] "from" "network" "not" "had" "their"   
## [106] "transpar" "ident" "institut" "ledger"

Conduct the tree-based classification to determine which terms contribute to classifying the news into either “BlockChain” or “Ebola.” Please carefully interpret the outputs with the rules and visualization technique, and verify your classification. (25 pts)

### Partition Data

set.seed(123)  
partition2 <- createDataPartition(y = newsDF.dtm$news, p = 0.7, list = FALSE)  
  
train3<- newsDF.dtm[partition2,]  
test3 <- newsDF.dtm[-partition2,]  
#str(train$news)

### Create the Tree

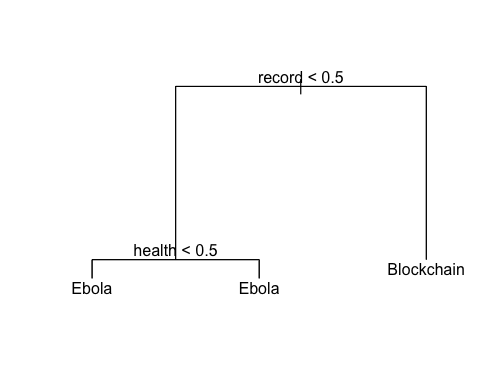
The data partitioning creates an interesting issue where it is difficult to pick terms that are guaranteed to be in one of the partitions. I was hoping to develop a more interesting tree, but unfortunately including less common words caused the model to fail.

In the tree created below, we can see that the root node it split on record, then by health on the left branch.

library(tree)

## Warning: package 'tree' was built under R version 4.0.2

treeterms <- tree(news ~ health + record + trust + media + public + document, data = train3)  
  
plot(treeterms)  
text(treeterms)

 ### Evaluate Model

This model did not perform well, specifically it only predicted a single Blockchain row and missed the other 8. This could be due to the low sample that could be pulled from the data, or the choice of terms in the first place. The accuracy is only 55% with an extremely high p-value.

library(e1071)  
treepred <- predict(treeterms, test3, type = "class")  
confusionMatrix(treepred, test3$news)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Blockchain Ebola  
## Blockchain 1 0  
## Ebola 8 9  
##   
## Accuracy : 0.5556   
## 95% CI : (0.3076, 0.7847)  
## No Information Rate : 0.5   
## P-Value [Acc > NIR] : 0.40726   
##   
## Kappa : 0.1111   
##   
## Mcnemar's Test P-Value : 0.01333   
##   
## Sensitivity : 0.11111   
## Specificity : 1.00000   
## Pos Pred Value : 1.00000   
## Neg Pred Value : 0.52941   
## Prevalence : 0.50000   
## Detection Rate : 0.05556   
## Detection Prevalence : 0.05556   
## Balanced Accuracy : 0.55556   
##   
## 'Positive' Class : Blockchain   
##