## Module 5 Assignment 1 Neural Networks

### Nicholas Barrett, 02/23/19

options(tidyverse.quiet = TRUE)  
library(tidyverse)  
library(caret)

## Warning: package 'caret' was built under R version 3.5.2

## Loading required package: lattice

##   
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':  
##   
## lift

library(nnet)

## Warning: package 'nnet' was built under R version 3.5.2

parole = read\_csv("parole.csv")

## Parsed with column specification:  
## cols(  
## male = col\_integer(),  
## race = col\_integer(),  
## age = col\_double(),  
## state = col\_integer(),  
## time.served = col\_double(),  
## max.sentence = col\_integer(),  
## multiple.offenses = col\_integer(),  
## crime = col\_integer(),  
## violator = col\_integer()  
## )

parole = parole %>% mutate(male = as.factor(male)) %>%   
 mutate(male = fct\_recode(male, "Female" = "0", "Male" = "1" ))  
parole = parole %>%mutate(race = as.factor(race)) %>%  
 mutate(race = fct\_recode(race, "Other" = "2", "White" = "1" ))  
parole = parole %>% mutate(state = as.factor(state))%>%  
 mutate(state = fct\_recode(state, "Kentucky"= "2", "Lousiana"= "3", "Virginia"= "4", "Other"= "1" ))  
parole = parole %>% mutate(crime = as.factor(crime))%>%  
 mutate(crime = fct\_recode(crime, "Larceny"= "2", "Drug-Related"= "3", "Driving-Related"= "4", "Other"= "1" ))  
parole = parole %>% mutate(multiple.offenses = as.factor(multiple.offenses))%>%  
 mutate(multiple.offenses = fct\_recode(multiple.offenses, "Incarcerated" = "1", "Other" = "0" ))  
parole = parole %>% mutate(violator = as.factor(violator))%>%  
 mutate(violator = fct\_recode(violator, "Violated" = "1", "No Violation" = "0" ))

## Task 1 - Split the data into training and testing sets. Training set should have 70% of the data. Use a set seed of 12345

set.seed(12345)  
  
train.rows = createDataPartition(y = parole$violator, p=0.7, list = FALSE)   
train = parole[train.rows,]   
test = parole[-train.rows,]

## Task 2 - Create a neural network to predict parole violation. Use 12 and a decay rate of 0.1. Use caret to implement 10-fold k-fold cross-validation. Use a random number seed of 1234.

start\_time = Sys.time()  
fitControl = trainControl(method = "cv", number = 10)  
  
nnetGrid <- expand.grid(size = 12, decay = 0.1)  
  
set.seed(1234)  
nnetTrain = train(violator ~.,   
 train,  
 method = "nnet",  
 tuneGrid = nnetGrid,  
 trControl = fitControl,  
 verbose = FALSE,  
 trace = FALSE)  
  
end\_time = Sys.time()  
end\_time-start\_time

## Time difference of 2.197165 secs

nnetTrain

## Neural Network   
##   
## 473 samples  
## 8 predictor  
## 2 classes: 'No Violation', 'Violated'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 427, 425, 426, 425, 425, 426, ...   
## Resampling results:  
##   
## Accuracy Kappa   
## 0.8836629 0.3218865  
##   
## Tuning parameter 'size' was held constant at a value of 12  
##   
## Tuning parameter 'decay' was held constant at a value of 0.1

## Task 3 - Use the model from Task 2 to develop predictions on the training set. Use caret’s confusionMatrix function to evaluate the model for quality. Comment on the quality.

predNetTrain = predict(nnetTrain, train)

confusionMatrix(predNetTrain, train$violator, positive = "Violated")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Violation Violated  
## No Violation 412 19  
## Violated 6 36  
##   
## Accuracy : 0.9471   
## 95% CI : (0.923, 0.9655)  
## No Information Rate : 0.8837   
## P-Value [Acc > NIR] : 1.65e-06   
##   
## Kappa : 0.7134   
## Mcnemar's Test P-Value : 0.0164   
##   
## Sensitivity : 0.65455   
## Specificity : 0.98565   
## Pos Pred Value : 0.85714   
## Neg Pred Value : 0.95592   
## Prevalence : 0.11628   
## Detection Rate : 0.07611   
## Detection Prevalence : 0.08879   
## Balanced Accuracy : 0.82010   
##   
## 'Positive' Class : Violated   
##

The nueral network model’s accuracy with the size to be 12 and decay to be 0.1 was 88.3% accurate. After tuning the model and using caret’s confusionnMatric, the accuracy improved to 94.7% when applied to the training set. So, the model is quite good.

## Task 4 - Create a neural network to predict parole violation. use a grid to search sizes 1 through 12 (by 1) and decay rates of 0.1 to 0.5 (by 0.1). use caret to implement 10-fild k-fold cross-validation. Use a random number seed of 1234. Suppress the text describing model convergence using the command trace = FALSE.

start\_time = Sys.time()  
fitControl = trainControl(method = "cv",  
 number = 10)  
  
nnetGrid = expand.grid(size = seq(from = 1, to = 12, by = 1),  
 decay = seq(from = 0.1, to = 0.5, by = 0.1))  
  
set.seed(1234)  
nnetFitTrain = train(violator ~.,   
 train,  
 method = "nnet",  
 tuneGrid = nnetGrid,  
 trControl = fitControl,  
 verbose = FALSE,  
 trace = FALSE)  
  
end\_time = Sys.time()  
end\_time-start\_time

## Time difference of 50.01371 secs

## Task 5 - Use the model from Task 4 to develop predictions on the training set. Use caret’s confusionMatric function to evaluate the quality of the model. Comment on the quality of the model.

nnetFitTrain

## Neural Network   
##   
## 473 samples  
## 8 predictor  
## 2 classes: 'No Violation', 'Violated'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 427, 425, 426, 425, 425, 426, ...   
## Resampling results across tuning parameters:  
##   
## size decay Accuracy Kappa   
## 1 0.1 0.8793170 0.24818573  
## 1 0.2 0.8899129 0.27778686  
## 1 0.3 0.8942607 0.27309967  
## 1 0.4 0.8838460 0.16103920  
## 1 0.5 0.8859293 0.06161616  
## 2 0.1 0.8753758 0.29957131  
## 2 0.2 0.8857462 0.28462246  
## 2 0.3 0.8900035 0.25780072  
## 2 0.4 0.8794519 0.20529774  
## 2 0.5 0.8880126 0.18514694  
## 3 0.1 0.8687673 0.23321579  
## 3 0.2 0.8732481 0.23873110  
## 3 0.3 0.8773223 0.22645350  
## 3 0.4 0.8836648 0.19571623  
## 3 0.5 0.8837091 0.17883252  
## 4 0.1 0.8815333 0.36168936  
## 4 0.2 0.8856113 0.29011294  
## 4 0.3 0.8751947 0.19884275  
## 4 0.4 0.8879201 0.24831511  
## 4 0.5 0.8837091 0.20278074  
## 5 0.1 0.8793613 0.33075682  
## 5 0.2 0.8900921 0.36096556  
## 5 0.3 0.8794962 0.26002484  
## 5 0.4 0.8773686 0.19349603  
## 5 0.5 0.8880107 0.22130408  
## 6 0.1 0.8773686 0.27801065  
## 6 0.2 0.8879645 0.33182442  
## 6 0.3 0.8815796 0.24879397  
## 6 0.4 0.8752852 0.21227657  
## 6 0.5 0.8859274 0.19075981  
## 7 0.1 0.8794056 0.34316189  
## 7 0.2 0.8731576 0.22884413  
## 7 0.3 0.8817145 0.28270775  
## 7 0.4 0.8774129 0.19535080  
## 7 0.5 0.8878739 0.26646403  
## 8 0.1 0.8667303 0.26125774  
## 8 0.2 0.8858368 0.32684983  
## 8 0.3 0.8815796 0.25929902  
## 8 0.4 0.8794962 0.20043016  
## 8 0.5 0.8858368 0.20254870  
## 9 0.1 0.8751079 0.27199523  
## 9 0.2 0.8794095 0.27498025  
## 9 0.3 0.8816239 0.26283372  
## 9 0.4 0.8794962 0.22437838  
## 9 0.5 0.8859274 0.20665206  
## 10 0.1 0.8689466 0.26154555  
## 10 0.2 0.8816258 0.29902102  
## 10 0.3 0.8857462 0.26091518  
## 10 0.4 0.8794519 0.19955663  
## 10 0.5 0.8815796 0.20649077  
## 11 0.1 0.8772780 0.30140762  
## 11 0.2 0.8794981 0.26680630  
## 11 0.3 0.8837091 0.26093614  
## 11 0.4 0.8774129 0.20972637  
## 11 0.5 0.8879201 0.23783490  
## 12 0.1 0.8751966 0.31416906  
## 12 0.2 0.8837091 0.33439321  
## 12 0.3 0.8837535 0.26791763  
## 12 0.4 0.8795405 0.23144289  
## 12 0.5 0.8857462 0.23558168  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were size = 1 and decay = 0.3.

predTrain = predict(nnetFitTrain, train)

confusionMatrix(predTrain, train$violator, positive = "Violated")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Violation Violated  
## No Violation 410 41  
## Violated 8 14  
##   
## Accuracy : 0.8964   
## 95% CI : (0.8654, 0.9224)  
## No Information Rate : 0.8837   
## P-Value [Acc > NIR] : 0.2171   
##   
## Kappa : 0.3183   
## Mcnemar's Test P-Value : 4.844e-06   
##   
## Sensitivity : 0.25455   
## Specificity : 0.98086   
## Pos Pred Value : 0.63636   
## Neg Pred Value : 0.90909   
## Prevalence : 0.11628   
## Detection Rate : 0.02960   
## Detection Prevalence : 0.04651   
## Balanced Accuracy : 0.61770   
##   
## 'Positive' Class : Violated   
##

The optimal value of the tuning parameters for the training set were a size of 1 and a decay of 0.3. After tuning, those parameters are different than the training set parameters of a size of 12 and a decay of 0.1. The confusionMatrix showed an accuracy of 89.6% which isnt quite as good as the model with a 94.7% accuracy.

## Task 6 - Use the model from Task 2 to develop predictions on the testing set. Comment on the model quality.

start\_time = Sys.time()  
fitControl = trainControl(method = "cv", number = 10)  
  
nnetGrid <- expand.grid(size = 12, decay = 0.1)  
  
set.seed(1234)  
nnetTest = train(violator ~.,   
 test,  
 method = "nnet",  
 tuneGrid = nnetGrid,  
 trControl = fitControl,  
 verbose = FALSE,  
 trace = FALSE)  
  
end\_time = Sys.time()  
end\_time-start\_time

## Time difference of 1.155774 secs

nnetTest

## Neural Network   
##   
## 202 samples  
## 8 predictor  
## 2 classes: 'No Violation', 'Violated'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 182, 182, 182, 182, 181, 182, ...   
## Resampling results:  
##   
## Accuracy Kappa   
## 0.8571429 0.1086252  
##   
## Tuning parameter 'size' was held constant at a value of 12  
##   
## Tuning parameter 'decay' was held constant at a value of 0.1

predNetTest= predict(nnetTest, test)

confusionMatrix(predNetTest, test$violator, positive = "Violated")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Violation Violated  
## No Violation 178 9  
## Violated 1 14  
##   
## Accuracy : 0.9505   
## 95% CI : (0.9108, 0.976)  
## No Information Rate : 0.8861   
## P-Value [Acc > NIR] : 0.001222   
##   
## Kappa : 0.7109   
## Mcnemar's Test P-Value : 0.026857   
##   
## Sensitivity : 0.60870   
## Specificity : 0.99441   
## Pos Pred Value : 0.93333   
## Neg Pred Value : 0.95187   
## Prevalence : 0.11386   
## Detection Rate : 0.06931   
## Detection Prevalence : 0.07426   
## Balanced Accuracy : 0.80155   
##   
## 'Positive' Class : Violated   
##

The model accuracy with the size to be 12 and decay to be 0.1 was 85.7% accurate. After tuning the model and using caret’s confusionnMatric, the accuracy improved to 95.05% when applied to the training set. So, the model is quite good.

## Task 7 - Use the model from Task 4 to develop prediction on the testing set. Use caret’s confusionMatrix to assess and comment on the quality of the model.

start\_time = Sys.time()  
fitControl = trainControl(method = "cv",  
 number = 10)  
  
nnetGrid = expand.grid(size = seq(from = 1, to = 12, by = 1),  
 decay = seq(from = 0.1, to = 0.5, by = 0.1))  
  
set.seed(1234)  
nnetFitTest = train(violator ~.,   
 test,  
 method = "nnet",  
 tuneGrid = nnetGrid,  
 trControl = fitControl,  
 verbose = FALSE,  
 trace = FALSE)  
  
end\_time = Sys.time()  
end\_time-start\_time

## Time difference of 23.72536 secs

nnetFitTest

## Neural Network   
##   
## 202 samples  
## 8 predictor  
## 2 classes: 'No Violation', 'Violated'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 182, 182, 182, 182, 181, 182, ...   
## Resampling results across tuning parameters:  
##   
## size decay Accuracy Kappa   
## 1 0.1 0.8866667 0.170879121  
## 1 0.2 0.8816667 0.049450549  
## 1 0.3 0.8864286 0.000000000  
## 1 0.4 0.8864286 0.000000000  
## 1 0.5 0.8864286 0.000000000  
## 2 0.1 0.8671429 0.059318996  
## 2 0.2 0.8864286 0.057142857  
## 2 0.3 0.8864286 0.057142857  
## 2 0.4 0.8864286 0.000000000  
## 2 0.5 0.8864286 0.000000000  
## 3 0.1 0.8719048 0.090792991  
## 3 0.2 0.8714286 -0.018253968  
## 3 0.3 0.8814286 -0.007142857  
## 3 0.4 0.8864286 0.000000000  
## 3 0.5 0.8864286 0.000000000  
## 4 0.1 0.8571429 0.057254394  
## 4 0.2 0.8716667 0.038339438  
## 4 0.3 0.8864286 0.057142857  
## 4 0.4 0.8814286 -0.007142857  
## 4 0.5 0.8814286 -0.007142857  
## 5 0.1 0.8621429 0.069880657  
## 5 0.2 0.8716667 -0.021978022  
## 5 0.3 0.8814286 -0.007142857  
## 5 0.4 0.8814286 -0.007142857  
## 5 0.5 0.8814286 -0.007142857  
## 6 0.1 0.8571429 0.061772549  
## 6 0.2 0.8866667 0.170879121  
## 6 0.3 0.8764286 -0.014285714  
## 6 0.4 0.8814286 -0.007142857  
## 6 0.5 0.8814286 -0.007142857  
## 7 0.1 0.8519048 -0.018009768  
## 7 0.2 0.8666667 -0.025946276  
## 7 0.3 0.8814286 -0.007142857  
## 7 0.4 0.8814286 -0.007142857  
## 7 0.5 0.8814286 -0.007142857  
## 8 0.1 0.8421429 0.001482389  
## 8 0.2 0.8666667 -0.025946276  
## 8 0.3 0.8814286 -0.007142857  
## 8 0.4 0.8814286 -0.007142857  
## 8 0.5 0.8814286 -0.007142857  
## 9 0.1 0.8519048 0.044139194  
## 9 0.2 0.8666667 -0.025946276  
## 9 0.3 0.8816667 0.049450549  
## 9 0.4 0.8814286 -0.007142857  
## 9 0.5 0.8814286 -0.007142857  
## 10 0.1 0.8719048 0.185264735  
## 10 0.2 0.8616667 -0.028471528  
## 10 0.3 0.8766667 -0.014835165  
## 10 0.4 0.8814286 -0.007142857  
## 10 0.5 0.8814286 -0.007142857  
## 11 0.1 0.8619048 0.072566323  
## 11 0.2 0.8619048 -0.033638584  
## 11 0.3 0.8764286 -0.014285714  
## 11 0.4 0.8814286 -0.007142857  
## 11 0.5 0.8814286 -0.007142857  
## 12 0.1 0.8619048 0.055250305  
## 12 0.2 0.8619048 -0.033638584  
## 12 0.3 0.8814286 0.050000000  
## 12 0.4 0.8814286 -0.007142857  
## 12 0.5 0.8814286 -0.007142857  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were size = 1 and decay = 0.1.

predTest = predict(nnetFitTest, test)

confusionMatrix(predTest, test$violator, positive = "Violated")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Violation Violated  
## No Violation 177 19  
## Violated 2 4  
##   
## Accuracy : 0.896   
## 95% CI : (0.8455, 0.9345)  
## No Information Rate : 0.8861   
## P-Value [Acc > NIR] : 0.3796524   
##   
## Kappa : 0.2401   
## Mcnemar's Test P-Value : 0.0004803   
##   
## Sensitivity : 0.1739   
## Specificity : 0.9888   
## Pos Pred Value : 0.6667   
## Neg Pred Value : 0.9031   
## Prevalence : 0.1139   
## Detection Rate : 0.0198   
## Detection Prevalence : 0.0297   
## Balanced Accuracy : 0.5814   
##   
## 'Positive' Class : Violated   
##

The optimal value of the tuning parameters for the training set were a size of 1 and a decay of 0.1. Those parameters are different than the training set parameters of a size of 12 and a decay of 0.1. The confusionMatrix showed an accuracy of the tuned testing set model of 89.6% which isnt quite as good as the neural network training model with a 94.7% accuracy.

## Task 8 - Comment on whether there appears to be overfitting in one or both of your models from Task 2 and 4.

Both models have a very high specificity and accuracy which could potentially mean they are both overfitted.