# Neural Networks

## Kinser, Shannon

### BAN502 - Module 5, Assignment 1

#install.packages("nnet")

library(tidyverse)

## -- Attaching packages ---------------------------------------------------------------------------------- tidyverse 1.2.1 --

## v ggplot2 3.1.0 v purrr 0.2.5  
## v tibble 1.4.2 v dplyr 0.7.7  
## v tidyr 0.8.2 v stringr 1.3.1  
## v readr 1.1.1 v forcats 0.3.0

## -- Conflicts ------------------------------------------------------------------------------------- tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

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" )) %>%  
 mutate(race = as.factor(race)) %>%  
 mutate(race = fct\_recode(race, "White" = "1", "Other" = "2")) %>%  
 mutate(state = as.factor(state)) %>%  
 mutate(state = fct\_recode(state, "Other" = "1", "Kentucky" = "2", "Louisiana" = "3", "Virginia" = "4")) %>%  
 mutate(crime = as.factor(crime)) %>%  
 mutate(crime = fct\_recode(crime, "Larceny" = "2", "Drug-Related" = "3", "Driving-Related" = "4", "Other" = "1")) %>%  
 mutate(multiple.offenses = as.factor(multiple.offenses)) %>%  
 mutate(multiple.offenses = fct\_recode(multiple.offenses, "Yes" = "1", "No" = "0")) %>%  
 mutate(violator = as.factor(violator)) %>%  
 mutate(violator = fct\_recode(violator, "Violated" = "1", "Did Not Violate" = "0"))

#### Task 1

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

#### Task 2

fitControl = trainControl(method = "cv",   
 number = 10)  
  
nnetGrid <- expand.grid(size = 12, decay = 0.1)  
  
set.seed(1234)  
nnetBasic = train(violator ~ .,   
 parole,  
 method = "nnet",  
 tuneGrid = nnetGrid,  
 trControl = fitControl,  
 verbose = FALSE,  
 trace = FALSE)

nnetBasic

## Neural Network   
##   
## 675 samples  
## 8 predictor  
## 2 classes: 'Did Not Violate', 'Violated'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 608, 607, 609, 607, 607, 607, ...   
## Resampling results:  
##   
## Accuracy Kappa   
## 0.8799505 0.309989  
##   
## 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

predNetBasic = predict(nnetBasic, train)

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Did Not Violate Violated  
## Did Not Violate 411 28  
## Violated 7 27  
##   
## Accuracy : 0.926   
## 95% CI : (0.8986, 0.9479)  
## No Information Rate : 0.8837   
## P-Value [Acc > NIR] : 0.0015972   
##   
## Kappa : 0.5684   
## Mcnemar's Test P-Value : 0.0007232   
##   
## Sensitivity : 0.49091   
## Specificity : 0.98325   
## Pos Pred Value : 0.79412   
## Neg Pred Value : 0.93622   
## Prevalence : 0.11628   
## Detection Rate : 0.05708   
## Detection Prevalence : 0.07188   
## Balanced Accuracy : 0.73708   
##   
## 'Positive' Class : Violated   
##

With an accuracy of 92.81%, and a p-value of well below .05, the variables are statistically significant, and this appears to be a very good model.

#### Task 4

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)  
nnetFit = train(violator ~ .,   
 parole,  
 method = "nnet",  
 trControl = fitControl,  
 tuneGrid = nnetGrid,  
 verbose = FALSE,  
 trace = FALSE)

nnetFit

## Neural Network   
##   
## 675 samples  
## 8 predictor  
## 2 classes: 'Did Not Violate', 'Violated'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 608, 607, 609, 607, 607, 607, ...   
## Resampling results across tuning parameters:  
##   
## size decay Accuracy Kappa   
## 1 0.1 0.8860996 0.2753963  
## 1 0.2 0.8846064 0.2319070  
## 1 0.3 0.8845844 0.2253438  
## 1 0.4 0.8816206 0.1482602  
## 1 0.5 0.8875023 0.1391544  
## 2 0.1 0.8846297 0.2694282  
## 2 0.2 0.8846070 0.2608178  
## 2 0.3 0.8830028 0.2431773  
## 2 0.4 0.8934306 0.2779410  
## 2 0.5 0.8800835 0.1208822  
## 3 0.1 0.8681186 0.1459753  
## 3 0.2 0.8755826 0.1845101  
## 3 0.3 0.8741340 0.1649251  
## 3 0.4 0.8801055 0.2168256  
## 3 0.5 0.8889736 0.2200998  
## 4 0.1 0.8859439 0.3533178  
## 4 0.2 0.8741127 0.2168751  
## 4 0.3 0.8815761 0.2380312  
## 4 0.4 0.8875030 0.2591706  
## 4 0.5 0.8815096 0.1925124  
## 5 0.1 0.8830021 0.3006922  
## 5 0.2 0.8815980 0.2820099  
## 5 0.3 0.8889736 0.2694574  
## 5 0.4 0.8785238 0.1680444  
## 5 0.5 0.8830247 0.2222113  
## 6 0.1 0.8814437 0.3444250  
## 6 0.2 0.8815315 0.2853840  
## 6 0.3 0.8844507 0.2308458  
## 6 0.4 0.8859878 0.2547509  
## 6 0.5 0.8904887 0.2505255  
## 7 0.1 0.8726182 0.2714744  
## 7 0.2 0.8861441 0.2815389  
## 7 0.3 0.8875476 0.2729190  
## 7 0.4 0.8800835 0.1814272  
## 7 0.5 0.8844727 0.1920932  
## 8 0.1 0.8726415 0.2541996  
## 8 0.2 0.8830241 0.2587072  
## 8 0.3 0.8963485 0.3442124  
## 8 0.4 0.8844507 0.2248290  
## 8 0.5 0.8859433 0.2167586  
## 9 0.1 0.8695227 0.2237128  
## 9 0.2 0.8875249 0.2483789  
## 9 0.3 0.8859433 0.2578335  
## 9 0.4 0.8844062 0.2255860  
## 9 0.5 0.8888845 0.2697893  
## 10 0.1 0.8784560 0.2997999  
## 10 0.2 0.8785019 0.2079413  
## 10 0.3 0.8859878 0.2973794  
## 10 0.4 0.8859433 0.2361274  
## 10 0.5 0.8858987 0.2210937  
## 11 0.1 0.8814424 0.3401194  
## 11 0.2 0.8829795 0.2924382  
## 11 0.3 0.8785684 0.2038563  
## 11 0.4 0.8859433 0.2042523  
## 11 0.5 0.8874584 0.2413304  
## 12 0.1 0.8814650 0.3571157  
## 12 0.2 0.8828245 0.2942325  
## 12 0.3 0.8920265 0.3503324  
## 12 0.4 0.8875030 0.2473338  
## 12 0.5 0.8844281 0.2198215  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were size = 8 and decay = 0.3.

#### Task 5

predNet = predict(nnetFit, train)

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Did Not Violate Violated  
## Did Not Violate 412 37  
## Violated 6 18  
##   
## Accuracy : 0.9091   
## 95% CI : (0.8795, 0.9334)  
## No Information Rate : 0.8837   
## P-Value [Acc > NIR] : 0.04597   
##   
## Kappa : 0.4143   
## Mcnemar's Test P-Value : 4.763e-06   
##   
## Sensitivity : 0.32727   
## Specificity : 0.98565   
## Pos Pred Value : 0.75000   
## Neg Pred Value : 0.91759   
## Prevalence : 0.11628   
## Detection Rate : 0.03805   
## Detection Prevalence : 0.05074   
## Balanced Accuracy : 0.65646   
##   
## 'Positive' Class : Violated   
##

This model appears to be of good quality, though the accuracy is not quite as high as the previous model, at 91.12%

#### Task 6

fitControl = trainControl(method = "cv",   
 number = 10)  
  
nnetGrid <- expand.grid(size = 12, decay = 0.1)  
  
set.seed(1234)  
nnetBasic = train(violator ~ .,   
 parole,  
 method = "nnet",  
 tuneGrid = nnetGrid,  
 trControl = fitControl,  
 verbose = FALSE,  
 trace = FALSE)  
predNetBasic = predict(nnetBasic, test)  
  
confusionMatrix(predNetBasic, test$violator, positive = "Violated")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Did Not Violate Violated  
## Did Not Violate 176 7  
## Violated 3 16  
##   
## Accuracy : 0.9505   
## 95% CI : (0.9108, 0.976)  
## No Information Rate : 0.8861   
## P-Value [Acc > NIR] : 0.001222   
##   
## Kappa : 0.7346   
## Mcnemar's Test P-Value : 0.342782   
##   
## Sensitivity : 0.69565   
## Specificity : 0.98324   
## Pos Pred Value : 0.84211   
## Neg Pred Value : 0.96175   
## Prevalence : 0.11386   
## Detection Rate : 0.07921   
## Detection Prevalence : 0.09406   
## Balanced Accuracy : 0.83945   
##   
## 'Positive' Class : Violated   
##

The quality of this model is very good, with an accuracy of 94.55%.

#### Task 7

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)  
nnetFit = train(violator ~ .,   
 parole,  
 method = "nnet",  
 trControl = fitControl,  
 tuneGrid = nnetGrid,  
 verbose = FALSE,  
 trace = FALSE)  
predNet = predict(nnetFit, test)  
  
confusionMatrix(predNet, test$violator, positive = "Violated")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Did Not Violate Violated  
## Did Not Violate 178 11  
## Violated 1 12  
##   
## Accuracy : 0.9406   
## 95% CI : (0.8985, 0.9689)  
## No Information Rate : 0.8861   
## P-Value [Acc > NIR] : 0.006397   
##   
## Kappa : 0.6368   
## Mcnemar's Test P-Value : 0.009375   
##   
## Sensitivity : 0.52174   
## Specificity : 0.99441   
## Pos Pred Value : 0.92308   
## Neg Pred Value : 0.94180   
## Prevalence : 0.11386   
## Detection Rate : 0.05941   
## Detection Prevalence : 0.06436   
## Balanced Accuracy : 0.75808   
##   
## 'Positive' Class : Violated   
##

The accuracy of this model also appears to be pretty good, though lower that the training set at 93.56%.

#### Task 8

I don’t believe either of these models appear to overfit. The test data does not perform much lower than the training data.