# Module 5: Assignment 1 – Classification Trees

## Kellie McLiverty

### Assignment Needs & Data Importation

Libraries needed for Assignemnt

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 0.7.8  
## v tidyr 0.8.2 v stringr 1.3.1  
## v readr 1.2.1 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\_double(),  
## race = col\_double(),  
## age = col\_double(),  
## state = col\_double(),  
## time.served = col\_double(),  
## max.sentence = col\_double(),  
## multiple.offenses = col\_double(),  
## crime = col\_double(),  
## violator = col\_double()  
## )

### Converting data to factors & setting training/testing data

#converting data into male or female  
parole = parole %>% mutate(male = as\_factor(as.character(male))) %>%  
mutate(male = fct\_recode(male,  
"Female" = "0",  
"Male" = "1"  
))  
  
#converting race into White or otherwise  
parole = parole %>% mutate(race = as\_factor(as.character(race))) %>%  
mutate(race = fct\_recode(race,  
"White" = "1",  
"Otherwise" = "2"  
))  
  
#converting states  
parole = parole %>% mutate(state = as\_factor(as.character(state))) %>%  
mutate(state = fct\_recode(state,  
"Any Other State" = "1",  
"Kentucky" = "2",  
"Louisiana" = "3",  
"Virginia" = "4"  
))  
  
#converting Crimes  
parole = parole %>% mutate(crime = as\_factor(as.character(crime))) %>%  
mutate(crime = fct\_recode(crime,  
"Any Other Crime" = "1",  
"Larceny" = "2",  
"Drug-related crime" = "3",  
"Driving-related crime" = "4"  
))  
  
#converting Multiple Offenses  
parole = parole %>% mutate(multiple.offenses = as\_factor(as.character(multiple.offenses))) %>%  
mutate(multiple.offenses = fct\_recode(multiple.offenses,  
"Otherwise" = "0",  
"Incarcerated for multiple offenses" = "1"  
))  
  
#converting parole  
parole = parole %>% mutate(violator = as\_factor(as.character(violator))) %>%  
mutate(violator = fct\_recode(violator,  
"Completed the parole without violation" = "0",  
"Violated the parole" = "1"  
))

For this assignment, we’ll start by splitting the data into training and testing, using a random number seed of 12345.

set.seed(12345) #sets random number seed for cross validation  
train.rows = createDataPartition(y = parole$violator, p=0.7, list= FALSE)  
train = parole[train.rows,]  
test = parole[-train.rows,]

Now we will be creating a neural network to predict parole violation.

start\_time = Sys.time() #for timing  
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)  
  
end\_time = Sys.time()  
end\_time-start\_time

## Time difference of 3.145116 secs

nnetBasic

## Neural Network   
##   
## 675 samples  
## 8 predictor  
## 2 classes: 'Completed the parole without violation', 'Violated the parole'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 608, 607, 609, 607, 607, 607, ...   
## Resampling results:  
##   
## Accuracy Kappa   
## 0.8799266 0.3222643  
##   
## Tuning parameter 'size' was held constant at a value of 12  
##   
## Tuning parameter 'decay' was held constant at a value of 0.1

This model returned final size of 12 and decay of 0.1 with an accuracy of 88%. We will next take this model and make predictions on who will violate parole, and take a look at the quality of this model.

#Predictions   
predNetBasic = predict(nnetBasic, train)  
  
  
#Confusion matrix  
confusionMatrix(predNetBasic, train$violator, positive = "Completed the parole without violation")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Completed the parole without violation  
## Completed the parole without violation 409  
## Violated the parole 9  
## Reference  
## Prediction Violated the parole  
## Completed the parole without violation 24  
## Violated the parole 31  
##   
## Accuracy : 0.9302   
## 95% CI : (0.9034, 0.9515)   
## No Information Rate : 0.8837   
## P-Value [Acc > NIR] : 0.0005254   
##   
## Kappa : 0.6149   
## Mcnemar's Test P-Value : 0.0148061   
##   
## Sensitivity : 0.9785   
## Specificity : 0.5636   
## Pos Pred Value : 0.9446   
## Neg Pred Value : 0.7750   
## Prevalence : 0.8837   
## Detection Rate : 0.8647   
## Detection Prevalence : 0.9154   
## Balanced Accuracy : 0.7711   
##   
## 'Positive' Class : Completed the parole without violation  
##

As we can see from the confusion matrix, this model reports back an accuracy of roughly 93% with a P-Value that is less than 0.05. This model has high sensitivity of 0.9785 and moderate specificity of 0.5636, which does indicate there is a bit of quality lacking in this model. The tradeoff would be to change the threshold to see if that would improve the model’s quality, but we will leave it like this for now.

Considering that our naive accuracy and the accuracy of our model were about the same, we may see some overfitting of the data that we will look at later with the predictions of the testing data.

Now, let’s take a look at another neural network with different parameters to see what the models return. Give this a moment to run.

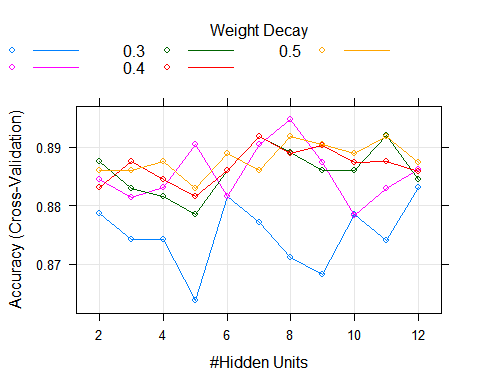
start\_time = Sys.time() #for timing  
fitControl = trainControl(method = "cv",   
 number = 10)  
  
nnetGrid = expand.grid(size = seq(from = 2, to = 12, by = 1), #rule of thumb --> between # of input and # of output layers  
 decay = seq(from = 0.1, to = 0.5, by = 0.1))  
set.seed(1234)  
nnetFit = train(violator ~ .,   
 parole,  
 method = "nnet",  
 trControl = fitControl,  
 tuneGrid = nnetGrid,  
 verbose = FALSE,  
 trace = FALSE)  
  
end\_time = Sys.time()  
end\_time-start\_time

## Time difference of 1.192369 mins

nnetFit

## Neural Network   
##   
## 675 samples  
## 8 predictor  
## 2 classes: 'Completed the parole without violation', 'Violated the parole'   
##   
## 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   
## 2 0.1 0.8786362 0.2808107  
## 2 0.2 0.8844734 0.2505248  
## 2 0.3 0.8875482 0.2893015  
## 2 0.4 0.8830473 0.1908547  
## 2 0.5 0.8860770 0.2246442  
## 3 0.1 0.8741333 0.2253701  
## 3 0.2 0.8814663 0.2499874  
## 3 0.3 0.8830021 0.2309008  
## 3 0.4 0.8875030 0.2722987  
## 3 0.5 0.8860544 0.2145919  
## 4 0.1 0.8742019 0.2642470  
## 4 0.2 0.8831577 0.2712408  
## 4 0.3 0.8815315 0.2371564  
## 4 0.4 0.8845173 0.2270156  
## 4 0.5 0.8875249 0.2365337  
## 5 0.1 0.8637295 0.2296724  
## 5 0.2 0.8904661 0.3322049  
## 5 0.3 0.8785677 0.2017872  
## 5 0.4 0.8815315 0.2030407  
## 5 0.5 0.8830021 0.2072619  
## 6 0.1 0.8815767 0.3131278  
## 6 0.2 0.8815980 0.2815706  
## 6 0.3 0.8859433 0.2340951  
## 6 0.4 0.8859659 0.2383317  
## 6 0.5 0.8889064 0.2211595  
## 7 0.1 0.8771197 0.2523629  
## 7 0.2 0.8904661 0.2863545  
## 7 0.3 0.8918476 0.2948277  
## 7 0.4 0.8918921 0.2729689  
## 7 0.5 0.8859878 0.2299659  
## 8 0.1 0.8711030 0.2735820  
## 8 0.2 0.8948327 0.3454710  
## 8 0.3 0.8890401 0.2876919  
## 8 0.4 0.8889955 0.2593342  
## 8 0.5 0.8918921 0.2740002  
## 9 0.1 0.8681186 0.2856987  
## 9 0.2 0.8874132 0.3232810  
## 9 0.3 0.8859878 0.2602482  
## 9 0.4 0.8903770 0.2491088  
## 9 0.5 0.8904216 0.2512283  
## 10 0.1 0.8784354 0.3540449  
## 10 0.2 0.8783895 0.2688093  
## 10 0.3 0.8859872 0.2796955  
## 10 0.4 0.8874584 0.2508432  
## 10 0.5 0.8889510 0.2588275  
## 11 0.1 0.8740449 0.2999131  
## 11 0.2 0.8829123 0.3270906  
## 11 0.3 0.8919813 0.2982609  
## 11 0.4 0.8874804 0.2533419  
## 11 0.5 0.8918921 0.2640208  
## 12 0.1 0.8830686 0.3401095  
## 12 0.2 0.8861209 0.2886209  
## 12 0.3 0.8845392 0.2740419  
## 12 0.4 0.8859207 0.2373754  
## 12 0.5 0.8873913 0.2138910  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were size = 8 and decay = 0.2.

plot(nnetFit)



This new model returned with a final size of 8 and decay of 0.2. We now want to see how this model will make predictions on the training and testing data.

predNet = predict(nnetFit, train)  
  
confusionMatrix(predNet, train$violator, positive = "Completed the parole without violation")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Completed the parole without violation  
## Completed the parole without violation 410  
## Violated the parole 8  
## Reference  
## Prediction Violated the parole  
## Completed the parole without violation 30  
## Violated the parole 25  
##   
## Accuracy : 0.9197   
## 95% CI : (0.8914, 0.9425)   
## No Information Rate : 0.8837   
## P-Value [Acc > NIR] : 0.0068328   
##   
## Kappa : 0.5269   
## Mcnemar's Test P-Value : 0.0006577   
##   
## Sensitivity : 0.9809   
## Specificity : 0.4545   
## Pos Pred Value : 0.9318   
## Neg Pred Value : 0.7576   
## Prevalence : 0.8837   
## Detection Rate : 0.8668   
## Detection Prevalence : 0.9302   
## Balanced Accuracy : 0.7177   
##   
## 'Positive' Class : Completed the parole without violation  
##

Looking at the confusion matrix for the nnetfit model, we can see we have a slight decrease in accuracy between the two models, in which nnetfit has an accuracy of 92%. We also see a slight decrease in the sensitivity 0.9809 and specificty 0.4545 of the model.

Predictions on testing set nnetBasic

predTest1 = predict(nnetBasic, newdata = test)  
head(predTest1)

## [1] Completed the parole without violation  
## [2] Completed the parole without violation  
## [3] Completed the parole without violation  
## [4] Completed the parole without violation  
## [5] Completed the parole without violation  
## [6] Completed the parole without violation  
## Levels: Completed the parole without violation Violated the parole

confusionMatrix(predTest1,test$violator, positive = "Completed the parole without violation") #predictions first then actual

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Completed the parole without violation  
## Completed the parole without violation 178  
## Violated the parole 1  
## Reference  
## Prediction Violated the parole  
## Completed the parole without violation 16  
## Violated the parole 7  
##   
## Accuracy : 0.9158   
## 95% CI : (0.8687, 0.9502)   
## No Information Rate : 0.8861   
## P-Value [Acc > NIR] : 0.108358   
##   
## Kappa : 0.4174   
## Mcnemar's Test P-Value : 0.000685   
##   
## Sensitivity : 0.9944   
## Specificity : 0.3043   
## Pos Pred Value : 0.9175   
## Neg Pred Value : 0.8750   
## Prevalence : 0.8861   
## Detection Rate : 0.8812   
## Detection Prevalence : 0.9604   
## Balanced Accuracy : 0.6494   
##   
## 'Positive' Class : Completed the parole without violation  
##

From this confusion matrix, we can see an accuracy of 92% of this model with a Sensitivity of 0.9944 and Specificity of 0.3043. It is interesting to note that the p value for the testing data in this model is above 0.05 and we have a naive accuracy of 89%. The accuracy of the testing data is slightly lower than the training data model.

Predictions on testing set For Model nnetFit

predNettest = predict(nnetFit, newdata = test)  
head(predNettest)

## [1] Completed the parole without violation  
## [2] Violated the parole   
## [3] Completed the parole without violation  
## [4] Completed the parole without violation  
## [5] Completed the parole without violation  
## [6] Completed the parole without violation  
## Levels: Completed the parole without violation Violated the parole

confusionMatrix(predNettest,test$violator, positive = "Completed the parole without violation") #predictions first then actual

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Completed the parole without violation  
## Completed the parole without violation 176  
## Violated the parole 3  
## Reference  
## Prediction Violated the parole  
## Completed the parole without violation 16  
## Violated the parole 7  
##   
## Accuracy : 0.9059   
## 95% CI : (0.857, 0.9424)   
## No Information Rate : 0.8861   
## P-Value [Acc > NIR] : 0.222456   
##   
## Kappa : 0.3816   
## Mcnemar's Test P-Value : 0.005905   
##   
## Sensitivity : 0.9832   
## Specificity : 0.3043   
## Pos Pred Value : 0.9167   
## Neg Pred Value : 0.7000   
## Prevalence : 0.8861   
## Detection Rate : 0.8713   
## Detection Prevalence : 0.9505   
## Balanced Accuracy : 0.6438   
##   
## 'Positive' Class : Completed the parole without violation  
##

The testing data for this model returned an Accuracy of 91%, Sensitivity of 0.9832 and Specificity of 0.3043. This model also shows the naive model being around 88% accurate and a p value of over 0.05. The testing data of this model shows a 2% decrease in accuracy of the model.

Looking over these two models, the nnetFit model appears to be the better model to avoid overfitting the data. Without adjusting the threshold for the nnetBasic dataset, I feel that this model may be closer to overfitting the data, especially with such a high sensitivity score.