## Module 5, Assignment 1

### Hackett, Evan

library(tidyverse)

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

## -- 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("C:/Users/Evan/Desktop/BAN 502/Module 3/Assignment 2/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()  
## )

View(parole)  
  
parole = parole %>% mutate(male = as\_factor(as.character(male))) %>%  
mutate(male = fct\_recode(male,  
"male" = "1",  
"female" = "0"))  
  
parole = parole %>% mutate(race = as\_factor(as.character(race))) %>%  
mutate(race = fct\_recode(race,  
"white" = "1",  
"otherwise" = "2"))  
  
parole = parole %>% mutate(state = as\_factor(as.character(state))) %>%  
mutate(state = fct\_recode(state,  
"Other" = "1",  
"Kentucky" = "2",  
"Louisana" = "3",  
"Virginia" = "4"))  
  
parole = parole %>% mutate(crime = as\_factor(as.character(crime))) %>%  
mutate(crime = fct\_recode(crime,  
"Other" = "1",  
"larceny" = "2",  
"drug-related" = "3",  
"driving-related" = "4"))  
  
parole = parole %>% mutate(multiple.offenses = as\_factor(as.character(multiple.offenses))) %>%  
mutate(multiple.offenses = fct\_recode(multiple.offenses,  
"multiple offenses" = "1",  
"otherwise" = "0"))  
  
parole = parole %>% mutate(violator = as\_factor(as.character(violator))) %>%  
mutate(violator = fct\_recode(violator,  
"violated parole" = "1",  
"completed parole" = "0"))

set.seed(12345)  
train.rows = createDataPartition(y = parole$violator, p=0.7, list = FALSE) #70% in training  
train = parole[train.rows,]   
test = parole[-train.rows,]

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 3.076176 secs

nnetBasic

## Neural Network   
##   
## 675 samples  
## 8 predictor  
## 2 classes: 'completed parole', 'violated 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

predNetBasic = predict(nnetBasic, train)

Confusion matrix

confusionMatrix(predNetBasic, train$violator, positive = "completed parole")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction completed parole violated parole  
## completed parole 409 24  
## violated parole 9 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 parole  
##

The model has a 93.02% accuracy rate on the train data set, which is greater than the naive accuracy rate of 88.37%, the difference between the two is statistically significant as the p-value is less than .05.

start\_time = Sys.time() #for timing  
fitControl = trainControl(method = "cv",   
 number = 10)  
  
nnetGrid = expand.grid(size = seq(from = 1, 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 1.298158 mins

nnetFit

## Neural Network   
##   
## 675 samples  
## 8 predictor  
## 2 classes: 'completed parole', 'violated 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   
## 1 0.1 0.8860996 0.27539625  
## 1 0.2 0.8831358 0.22036659  
## 1 0.3 0.8860770 0.22879320  
## 1 0.4 0.8831132 0.15212746  
## 1 0.5 0.8845166 0.09792188  
## 2 0.1 0.8802851 0.27447784  
## 2 0.2 0.8785245 0.22152192  
## 2 0.3 0.8785464 0.19928703  
## 2 0.4 0.8830686 0.19772626  
## 2 0.5 0.8786130 0.12512155  
## 3 0.1 0.8784799 0.29081727  
## 3 0.2 0.8844507 0.25013234  
## 3 0.3 0.8799725 0.19589131  
## 3 0.4 0.8860996 0.26138337  
## 3 0.5 0.8831358 0.22640360  
## 4 0.1 0.8667591 0.24486223  
## 4 0.2 0.8845399 0.28477941  
## 4 0.3 0.8859433 0.25528365  
## 4 0.4 0.8800609 0.17015102  
## 4 0.5 0.8829575 0.20886228  
## 5 0.1 0.8725523 0.28022760  
## 5 0.2 0.8859439 0.29679295  
## 5 0.3 0.8844953 0.23929651  
## 5 0.4 0.8844281 0.21982147  
## 5 0.5 0.8800164 0.18428469  
## 6 0.1 0.8695892 0.24255063  
## 6 0.2 0.8844727 0.31579548  
## 6 0.3 0.8888838 0.24521616  
## 6 0.4 0.8844288 0.22369549  
## 6 0.5 0.8889729 0.24899322  
## 7 0.1 0.8904435 0.35813188  
## 7 0.2 0.8741566 0.25430037  
## 7 0.3 0.8845173 0.25597072  
## 7 0.4 0.8918030 0.23402163  
## 7 0.5 0.8860098 0.22798699  
## 8 0.1 0.8844501 0.34281531  
## 8 0.2 0.8845618 0.33902028  
## 8 0.3 0.8874804 0.27575046  
## 8 0.4 0.8830686 0.19983020  
## 8 0.5 0.8859878 0.24128266  
## 9 0.1 0.8784793 0.30099362  
## 9 0.2 0.8784566 0.28616746  
## 9 0.3 0.8875695 0.30428209  
## 9 0.4 0.8889510 0.23204690  
## 9 0.5 0.8874804 0.23375539  
## 10 0.1 0.8710824 0.27885801  
## 10 0.2 0.8814198 0.32244321  
## 10 0.3 0.8860544 0.26395559  
## 10 0.4 0.8889290 0.26397964  
## 10 0.5 0.8873913 0.24020504  
## 11 0.1 0.8816200 0.35642251  
## 11 0.2 0.8828677 0.28647690  
## 11 0.3 0.8889510 0.30462243  
## 11 0.4 0.8904442 0.28791900  
## 11 0.5 0.8889064 0.22917786  
## 12 0.1 0.8696112 0.28600455  
## 12 0.2 0.8844946 0.31973627  
## 12 0.3 0.8860098 0.27921784  
## 12 0.4 0.8859878 0.24050575  
## 12 0.5 0.8859213 0.22716343  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were size = 7 and decay = 0.4.

predNet = predict(nnetFit, train)

confusionMatrix(predNet, train$violator, positive = "completed parole")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction completed parole violated parole  
## completed parole 411 38  
## violated parole 7 17  
##   
## Accuracy : 0.9049   
## 95% CI : (0.8748, 0.9298)  
## No Information Rate : 0.8837   
## P-Value [Acc > NIR] : 0.08373   
##   
## Kappa : 0.3871   
## Mcnemar's Test P-Value : 7.744e-06   
##   
## Sensitivity : 0.9833   
## Specificity : 0.3091   
## Pos Pred Value : 0.9154   
## Neg Pred Value : 0.7083   
## Prevalence : 0.8837   
## Detection Rate : 0.8689   
## Detection Prevalence : 0.9493   
## Balanced Accuracy : 0.6462   
##   
## 'Positive' Class : completed parole  
##

This model is not as accurate as the other model, but is still fairly accurate at 90.49%, the naive model is at 88.37% however, the difference between the two models is not statistically significant.

predNetBasic\_test = predict(nnetBasic, test)

confusionMatrix(predNetBasic\_test, test$violator, positive = "completed parole")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction completed parole violated parole  
## completed parole 178 16  
## violated parole 1 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 parole  
##

when looking at this model through the test data set our accuracy is now 91.58% and the naive accruacy rate is 88.61%, it should be noted that the difference is not statistically significant.

predNet\_test = predict(nnetFit, test)

confusionMatrix(predNet\_test, test$violator, positive = "completed parole")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction completed parole violated parole  
## completed parole 176 20  
## violated parole 3 3  
##   
## Accuracy : 0.8861   
## 95% CI : (0.8341, 0.9264)  
## No Information Rate : 0.8861   
## P-Value [Acc > NIR] : 0.5552509   
##   
## Kappa : 0.1677   
## Mcnemar's Test P-Value : 0.0008492   
##   
## Sensitivity : 0.9832   
## Specificity : 0.1304   
## Pos Pred Value : 0.8980   
## Neg Pred Value : 0.5000   
## Prevalence : 0.8861   
## Detection Rate : 0.8713   
## Detection Prevalence : 0.9703   
## Balanced Accuracy : 0.5568   
##   
## 'Positive' Class : completed parole  
##

When looking at this model, utilizing the test dataste, the accuracy rate for the model and naive model are identical at 88.61%.

When looking at the models that were created in Task 2 and 4 I can confidently say that we are not overfitting with the model in task 2 as there is not a large change between the accuracy rates when we switch from the training dataset to the testing dataset, the accuracy changes from ~93% to ~91%. I do not that think that we are experiencing overfit in the task 4 model either as we are seeing a roughly similar decrease in accruacy as we saw in task 2, which was around 2%.