## Module 5 - Assignment 1

#install.packages("nnet")

library(tidyverse)

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

## v ggplot2 3.1.0 v purrr 0.3.0  
## v tibble 2.0.1 v dplyr 0.7.8  
## v tidyr 0.8.2 v stringr 1.3.1  
## v readr 1.3.1 v forcats 0.3.0

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

library(caret)

## Loading required package: lattice

##   
## Attaching package: 'caret'

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

library(nnet)

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()  
## )

summary(parole)

## male race age state   
## Min. :0.0000 Min. :1.000 Min. :18.40 Min. :1.000   
## 1st Qu.:1.0000 1st Qu.:1.000 1st Qu.:25.35 1st Qu.:2.000   
## Median :1.0000 Median :1.000 Median :33.70 Median :3.000   
## Mean :0.8074 Mean :1.424 Mean :34.51 Mean :2.887   
## 3rd Qu.:1.0000 3rd Qu.:2.000 3rd Qu.:42.55 3rd Qu.:4.000   
## Max. :1.0000 Max. :2.000 Max. :67.00 Max. :4.000   
## time.served max.sentence multiple.offenses crime   
## Min. :0.000 Min. : 1.00 Min. :0.0000 Min. :1.000   
## 1st Qu.:3.250 1st Qu.:12.00 1st Qu.:0.0000 1st Qu.:1.000   
## Median :4.400 Median :12.00 Median :1.0000 Median :2.000   
## Mean :4.198 Mean :13.06 Mean :0.5363 Mean :2.059   
## 3rd Qu.:5.200 3rd Qu.:15.00 3rd Qu.:1.0000 3rd Qu.:3.000   
## Max. :6.000 Max. :18.00 Max. :1.0000 Max. :4.000   
## violator   
## Min. :0.0000   
## 1st Qu.:0.0000   
## Median :0.0000   
## Mean :0.1156   
## 3rd Qu.:0.0000   
## Max. :1.0000

str(parole)

## Classes 'spec\_tbl\_df', 'tbl\_df', 'tbl' and 'data.frame': 675 obs. of 9 variables:  
## $ male : num 1 0 1 1 1 1 1 0 0 1 ...  
## $ race : num 1 1 2 1 2 2 1 1 1 2 ...  
## $ age : num 33.2 39.7 29.5 22.4 21.6 46.7 31 24.6 32.6 29.1 ...  
## $ state : num 1 1 1 1 1 1 1 1 1 1 ...  
## $ time.served : num 5.5 5.4 5.6 5.7 5.4 6 6 4.8 4.5 4.7 ...  
## $ max.sentence : num 18 12 12 18 12 18 18 12 13 12 ...  
## $ multiple.offenses: num 0 0 0 0 0 0 0 0 0 0 ...  
## $ crime : num 4 3 3 1 1 4 3 1 3 2 ...  
## $ violator : num 0 0 0 0 0 0 0 0 0 0 ...  
## - attr(\*, "spec")=  
## .. 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()  
## .. )

parole = parole %>% mutate(male = as\_factor(as.character(male))) %>%  
mutate(male = fct\_recode(male,  
"female" = "0",  
"male" = "1"))%>%  
   
 mutate(age = as.numeric(as.character(age)))%>%  
   
mutate(time.served = as.numeric(as.character(time.served)))%>%   
   
 mutate(max.sentence = as\_factor(as.character(max.sentence)))%>%   
   
 mutate(race = as\_factor(as.character(race)))%>%  
   
mutate(race = fct\_recode(race,  
"white" = "1",  
"otherwise" = "2"))%>%   
   
 mutate(multiple.offenses = as\_factor(as.character(multiple.offenses)))%>%  
   
mutate(multiple.offenses = fct\_recode(multiple.offenses,  
"otherwise" = "0",  
"multipleoffenses" = "1"))%>%  
   
 mutate(state = as\_factor(as.character(state))) %>%  
   
mutate(state = fct\_recode(state,  
"anyotherstate" = "1",  
"Kentucky" = "2",  
"Louisiana" = "3",  
"Virginia" = "4"))%>%  
   
 mutate(crime = as\_factor(as.character(crime))) %>%  
   
   
mutate(crime = fct\_recode(crime,  
"othercrime" = "1",  
"larceny" = "2",  
"drug-relatedcrime" = "3",  
"driving-relatedcrime" = "4"))%>%   
   
 mutate(violator = as\_factor(as.character(violator)))%>%  
   
mutate(violator = fct\_recode(violator,  
"violatedparole" = "1",  
"noparoleviolation" = "0"))  
str(parole)

## Classes 'tbl\_df', 'tbl' and 'data.frame': 675 obs. of 9 variables:  
## $ male : Factor w/ 2 levels "male","female": 1 2 1 1 1 1 1 2 2 1 ...  
## $ race : Factor w/ 2 levels "white","otherwise": 1 1 2 1 2 2 1 1 1 2 ...  
## $ age : num 33.2 39.7 29.5 22.4 21.6 46.7 31 24.6 32.6 29.1 ...  
## $ state : Factor w/ 4 levels "anyotherstate",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ time.served : num 5.5 5.4 5.6 5.7 5.4 6 6 4.8 4.5 4.7 ...  
## $ max.sentence : Factor w/ 17 levels "18","12","13",..: 1 2 2 1 2 1 1 2 3 2 ...  
## $ multiple.offenses: Factor w/ 2 levels "otherwise","multipleoffenses": 1 1 1 1 1 1 1 1 1 1 ...  
## $ crime : Factor w/ 4 levels "driving-relatedcrime",..: 1 2 2 3 3 1 2 3 2 4 ...  
## $ violator : Factor w/ 2 levels "noparoleviolation",..: 1 1 1 1 1 1 1 1 1 1 ...

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

## Time difference of 3.04852 secs

nnetBasic

## Neural Network   
##   
## 473 samples  
## 8 predictor  
## 2 classes: 'noparoleviolation', 'violatedparole'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 427, 425, 426, 425, 425, 426, ...   
## Resampling results:  
##   
## Accuracy Kappa   
## 0.8795849 0.3316793  
##   
## 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)

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction noparoleviolation violatedparole  
## noparoleviolation 417 11  
## violatedparole 1 44  
##   
## Accuracy : 0.9746   
## 95% CI : (0.9561, 0.9868)  
## No Information Rate : 0.8837   
## P-Value [Acc > NIR] : 3.073e-13   
##   
## Kappa : 0.866   
## Mcnemar's Test P-Value : 0.009375   
##   
## Sensitivity : 0.80000   
## Specificity : 0.99761   
## Pos Pred Value : 0.97778   
## Neg Pred Value : 0.97430   
## Prevalence : 0.11628   
## Detection Rate : 0.09302   
## Detection Prevalence : 0.09514   
## Balanced Accuracy : 0.89880   
##   
## 'Positive' Class : violatedparole   
##

This is a great model to use. The accuracy is extremly high at .97 and we see that the P-value shows strong significance.

start\_time = Sys.time()   
fitControl = trainControl(method = "cv",   
 number = 10)  
  
nnetGrid = expand.grid(size = seq(from = 2, to = 12, by = 1),   
 decay = seq(from = 0.1, to = 0.5, by = 0.1))  
set.seed(1234)  
nnetFit = train(violator ~ .,   
 train,  
 method = "nnet",  
 trControl = fitControl,  
 tuneGrid = nnetGrid,  
 trace = FALSE,  
 verbose = FALSE)  
  
end\_time = Sys.time()  
end\_time-start\_time

## Time difference of 1.041027 mins

nnetFit

## Neural Network   
##   
## 473 samples  
## 8 predictor  
## 2 classes: 'noparoleviolation', 'violatedparole'   
##   
## 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   
## 2 0.1 0.8795425 0.3376303  
## 2 0.2 0.8752852 0.2063142  
## 2 0.3 0.8731576 0.1884573  
## 2 0.4 0.8774591 0.1792752  
## 2 0.5 0.8859736 0.2141438  
## 3 0.1 0.8794037 0.3196840  
## 3 0.2 0.8858792 0.3374440  
## 3 0.3 0.8817626 0.2215886  
## 3 0.4 0.8816701 0.2052553  
## 3 0.5 0.8858368 0.2326297  
## 4 0.1 0.8945286 0.4432679  
## 4 0.2 0.8774591 0.2906431  
## 4 0.3 0.8754664 0.2132046  
## 4 0.4 0.8754683 0.1937099  
## 4 0.5 0.8816701 0.2209436  
## 5 0.1 0.8817164 0.3784792  
## 5 0.2 0.8775960 0.2697419  
## 5 0.3 0.8711205 0.1788997  
## 5 0.4 0.8732038 0.1954834  
## 5 0.5 0.8837535 0.2260718  
## 6 0.1 0.8856113 0.3904522  
## 6 0.2 0.8838884 0.3185759  
## 6 0.3 0.8752409 0.2025969  
## 6 0.4 0.8775054 0.2148786  
## 6 0.5 0.8795868 0.2165480  
## 7 0.1 0.8814427 0.3495036  
## 7 0.2 0.8774591 0.2658248  
## 7 0.3 0.8774148 0.2314184  
## 7 0.4 0.8775516 0.2155281  
## 7 0.5 0.8816701 0.2209436  
## 8 0.1 0.8774129 0.3205081  
## 8 0.2 0.8837535 0.3017489  
## 8 0.3 0.8711205 0.1827142  
## 8 0.4 0.8732038 0.1814282  
## 8 0.5 0.8837997 0.2251932  
## 9 0.1 0.8753276 0.3233297  
## 9 0.2 0.8921755 0.3653160  
## 9 0.3 0.8732944 0.1882121  
## 9 0.4 0.8732038 0.1850645  
## 9 0.5 0.8816701 0.2209436  
## 10 0.1 0.8732019 0.3333153  
## 10 0.2 0.8774591 0.2955383  
## 10 0.3 0.8689023 0.1574584  
## 10 0.4 0.8753777 0.1740181  
## 10 0.5 0.8858368 0.2326297  
## 11 0.1 0.8797217 0.3609670  
## 11 0.2 0.8732019 0.2905250  
## 11 0.3 0.8753315 0.1862766  
## 11 0.4 0.8732038 0.1850645  
## 11 0.5 0.8858368 0.2431658  
## 12 0.1 0.8857905 0.3567574  
## 12 0.2 0.8753296 0.2920849  
## 12 0.3 0.8689466 0.1782747  
## 12 0.4 0.8732038 0.1850645  
## 12 0.5 0.8795444 0.2120911  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were size = 4 and decay = 0.1.

predNetFit = predict(nnetFit, train)

confusionMatrix(predNetFit, train$violator, positive = "violatedparole")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction noparoleviolation violatedparole  
## noparoleviolation 413 23  
## violatedparole 5 32  
##   
## Accuracy : 0.9408   
## 95% CI : (0.9156, 0.9603)  
## No Information Rate : 0.8837   
## P-Value [Acc > NIR] : 1.87e-05   
##   
## Kappa : 0.6642   
## Mcnemar's Test P-Value : 0.001315   
##   
## Sensitivity : 0.58182   
## Specificity : 0.98804   
## Pos Pred Value : 0.86486   
## Neg Pred Value : 0.94725   
## Prevalence : 0.11628   
## Detection Rate : 0.06765   
## Detection Prevalence : 0.07822   
## Balanced Accuracy : 0.78493   
##   
## 'Positive' Class : violatedparole   
##

Stiil a great model to use but not quite as accurate as the previous one. We also see the P-value fall but regardless it is so much less than .05 it isn’t really worth note.

predNetBasicTest = predict(nnetBasic, test)

confusionMatrix(predNetBasicTest, test$violator, positive = "violatedparole")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction noparoleviolation violatedparole  
## noparoleviolation 171 16  
## violatedparole 8 7  
##   
## Accuracy : 0.8812   
## 95% CI : (0.8284, 0.9224)  
## No Information Rate : 0.8861   
## P-Value [Acc > NIR] : 0.6396   
##   
## Kappa : 0.306   
## Mcnemar's Test P-Value : 0.1530   
##   
## Sensitivity : 0.30435   
## Specificity : 0.95531   
## Pos Pred Value : 0.46667   
## Neg Pred Value : 0.91444   
## Prevalence : 0.11386   
## Detection Rate : 0.03465   
## Detection Prevalence : 0.07426   
## Balanced Accuracy : 0.62983   
##   
## 'Positive' Class : violatedparole   
##

This model begins to slide a bit. Again, it is a relatively high accuracy but it falls below the naive rate so you are better off just guessing the majority.

predNetFitTest = predict(nnetFit, test)

confusionMatrix(predNetFitTest, test$violator, positive = "violatedparole")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction noparoleviolation violatedparole  
## noparoleviolation 173 16  
## violatedparole 6 7  
##   
## Accuracy : 0.8911   
## 95% CI : (0.8398, 0.9305)  
## No Information Rate : 0.8861   
## P-Value [Acc > NIR] : 0.46721   
##   
## Kappa : 0.3341   
## Mcnemar's Test P-Value : 0.05501   
##   
## Sensitivity : 0.30435   
## Specificity : 0.96648   
## Pos Pred Value : 0.53846   
## Neg Pred Value : 0.91534   
## Prevalence : 0.11386   
## Detection Rate : 0.03465   
## Detection Prevalence : 0.06436   
## Balanced Accuracy : 0.63541   
##   
## 'Positive' Class : violatedparole   
##

This model is slightly better than the one before. Though, I would still rather use the models surrounding the training data set.

All this said we may see that the model that is used from task 2 is overfitting. The model works well with the first data set of training but then strongly declines in performance when used against the test data set.For the model from task 4 we still see where there may be signs of overfitting but it is not nearly as bad as the first model. The accuracies between the test and training data sets begin to close a bit and also the P-values are closer to each other. In both cases I feel as if there are signs of overfitting but the second model doesn’t cause quite as many red flags.