# Neural Network Assignment

## BAN502 Predictive Analytics

### Susan Wiggins

The following libraries are needed for this assignment:

library(tidyverse)

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

## v ggplot2 3.1.1 v purrr 0.3.2  
## v tibble 2.1.1 v dplyr 0.8.1  
## v tidyr 0.8.3 v stringr 1.4.0  
## v readr 1.3.1 v forcats 0.4.0

## -- 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)

Reading the data set parole and recoding:

library(readr)  
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()  
## )

View(parole)

parole = parole %>% mutate(male = as\_factor(as.character(male)))%>%   
mutate(male = fct\_recode(male, "male" = "1", "female" = "0"))  
  
glimpse(parole)

## Observations: 675  
## Variables: 9  
## $ male <fct> male, female, male, male, male, male, male, ...  
## $ race <dbl> 1, 1, 2, 1, 2, 2, 1, 1, 1, 2, 2, 1, 1, 1, 1,...  
## $ age <dbl> 33.2, 39.7, 29.5, 22.4, 21.6, 46.7, 31.0, 24...  
## $ state <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,...  
## $ time.served <dbl> 5.5, 5.4, 5.6, 5.7, 5.4, 6.0, 6.0, 4.8, 4.5,...  
## $ max.sentence <dbl> 18, 12, 12, 18, 12, 18, 18, 12, 13, 12, 12, ...  
## $ multiple.offenses <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,...  
## $ crime <dbl> 4, 3, 3, 1, 1, 4, 3, 1, 3, 2, 1, 1, 3, 3, 3,...  
## $ violator <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...

parole = parole %>% mutate(race = as\_factor(as.character(race)))%>%   
mutate(race = fct\_recode(race, "white" = "1", "otherwise" = "2"))  
  
glimpse(parole)

## Observations: 675  
## Variables: 9  
## $ male <fct> male, female, male, male, male, male, male, ...  
## $ race <fct> white, white, otherwise, white, otherwise, o...  
## $ age <dbl> 33.2, 39.7, 29.5, 22.4, 21.6, 46.7, 31.0, 24...  
## $ state <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,...  
## $ time.served <dbl> 5.5, 5.4, 5.6, 5.7, 5.4, 6.0, 6.0, 4.8, 4.5,...  
## $ max.sentence <dbl> 18, 12, 12, 18, 12, 18, 18, 12, 13, 12, 12, ...  
## $ multiple.offenses <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,...  
## $ crime <dbl> 4, 3, 3, 1, 1, 4, 3, 1, 3, 2, 1, 1, 3, 3, 3,...  
## $ violator <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...

parole = parole %>% mutate(state = as\_factor(as.character(state)))%>%   
mutate(state = fct\_recode(state, "Kentucky" = "2", "Louisiana" = "3", "Virginia" = "4", "other state" = "1"))  
  
glimpse(parole)

## Observations: 675  
## Variables: 9  
## $ male <fct> male, female, male, male, male, male, male, ...  
## $ race <fct> white, white, otherwise, white, otherwise, o...  
## $ age <dbl> 33.2, 39.7, 29.5, 22.4, 21.6, 46.7, 31.0, 24...  
## $ state <fct> other state, other state, other state, other...  
## $ time.served <dbl> 5.5, 5.4, 5.6, 5.7, 5.4, 6.0, 6.0, 4.8, 4.5,...  
## $ max.sentence <dbl> 18, 12, 12, 18, 12, 18, 18, 12, 13, 12, 12, ...  
## $ multiple.offenses <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,...  
## $ crime <dbl> 4, 3, 3, 1, 1, 4, 3, 1, 3, 2, 1, 1, 3, 3, 3,...  
## $ violator <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...

parole = parole %>% mutate(crime = as\_factor(as.character(crime)))%>%   
mutate(crime = fct\_recode(crime, "larceny" = "2", "drug-related crime" = "3", "driving-related crime" = "4", "other crime" = "1"))  
  
glimpse(parole)

## Observations: 675  
## Variables: 9  
## $ male <fct> male, female, male, male, male, male, male, ...  
## $ race <fct> white, white, otherwise, white, otherwise, o...  
## $ age <dbl> 33.2, 39.7, 29.5, 22.4, 21.6, 46.7, 31.0, 24...  
## $ state <fct> other state, other state, other state, other...  
## $ time.served <dbl> 5.5, 5.4, 5.6, 5.7, 5.4, 6.0, 6.0, 4.8, 4.5,...  
## $ max.sentence <dbl> 18, 12, 12, 18, 12, 18, 18, 12, 13, 12, 12, ...  
## $ multiple.offenses <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,...  
## $ crime <fct> driving-related crime, drug-related crime, d...  
## $ violator <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...

parole = parole %>% mutate(violator = as\_factor(as.character(violator)))%>%   
mutate(violator = fct\_recode(violator, "Yes" = "1", "No" = "0"))  
  
glimpse(parole)

## Observations: 675  
## Variables: 9  
## $ male <fct> male, female, male, male, male, male, male, ...  
## $ race <fct> white, white, otherwise, white, otherwise, o...  
## $ age <dbl> 33.2, 39.7, 29.5, 22.4, 21.6, 46.7, 31.0, 24...  
## $ state <fct> other state, other state, other state, other...  
## $ time.served <dbl> 5.5, 5.4, 5.6, 5.7, 5.4, 6.0, 6.0, 4.8, 4.5,...  
## $ max.sentence <dbl> 18, 12, 12, 18, 12, 18, 18, 12, 13, 12, 12, ...  
## $ multiple.offenses <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,...  
## $ crime <fct> driving-related crime, drug-related crime, d...  
## $ violator <fct> No, No, No, No, No, No, No, No, No, No, No, ...

parole = parole %>% mutate(multiple.offenses = as\_factor(as.character(multiple.offenses)))%>%   
mutate(multiple.offenses = fct\_recode(multiple.offenses, "Multiple Offenses" = "1", "Otherwise" = "0"))  
  
  
glimpse(parole)

## Observations: 675  
## Variables: 9  
## $ male <fct> male, female, male, male, male, male, male, ...  
## $ race <fct> white, white, otherwise, white, otherwise, o...  
## $ age <dbl> 33.2, 39.7, 29.5, 22.4, 21.6, 46.7, 31.0, 24...  
## $ state <fct> other state, other state, other state, other...  
## $ time.served <dbl> 5.5, 5.4, 5.6, 5.7, 5.4, 6.0, 6.0, 4.8, 4.5,...  
## $ max.sentence <dbl> 18, 12, 12, 18, 12, 18, 18, 12, 13, 12, 12, ...  
## $ multiple.offenses <fct> Otherwise, Otherwise, Otherwise, Otherwise, ...  
## $ crime <fct> driving-related crime, drug-related crime, d...  
## $ violator <fct> No, No, No, No, No, No, No, No, No, No, No, ...

summary(parole)

## male race age state   
## male :545 white :389 Min. :18.40 other state:143   
## female:130 otherwise:286 1st Qu.:25.35 Kentucky :120   
## Median :33.70 Louisiana : 82   
## Mean :34.51 Virginia :330   
## 3rd Qu.:42.55   
## Max. :67.00   
## time.served max.sentence multiple.offenses  
## Min. :0.000 Min. : 1.00 Otherwise :313   
## 1st Qu.:3.250 1st Qu.:12.00 Multiple Offenses:362   
## Median :4.400 Median :12.00   
## Mean :4.198 Mean :13.06   
## 3rd Qu.:5.200 3rd Qu.:15.00   
## Max. :6.000 Max. :18.00   
## crime violator   
## driving-related crime:101 No :597   
## drug-related crime :153 Yes: 78   
## other crime :315   
## larceny :106   
##   
##

### Task 1

Split the data into training and testing sets. Your training set should have 70% of the data. Use a random number (set.seed) of 12345.

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,]

### Task 2

Create a neural network to predict parole violation. Use a size of 12 (corresponding roughly to the number of variables, including dummy variables) and a decay rate of 0.1. Use caret to implement 10-fold k-fold cross-validation. Use a random number seed of 1234. To suppress all of the text describing model convergence, add the command: trace = FALSE after verbose = FALSE.

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 2.597318 secs

nnetBasic

## Neural Network   
##   
## 675 samples  
## 8 predictor  
## 2 classes: 'No', 'Yes'   
##   
## 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

### Task 3

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

The model has a 93.02% accuracy, whereas the naive model is 88.37% accurate in model accuracy. The p-value also indicates that the model is significant due to listed as 0.0005254 which is less than .05. Specificity is at 97.85% and sensitivity is 56.36%. The model is indicated as good for the quality of these values.

predNetBasic = predict(nnetBasic, train)  
  
confusionMatrix(predNetBasic, train$violator, positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 409 24  
## Yes 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 : 0.56364   
## Specificity : 0.97847   
## Pos Pred Value : 0.77500   
## Neg Pred Value : 0.94457   
## Prevalence : 0.11628   
## Detection Rate : 0.06554   
## Detection Prevalence : 0.08457   
## Balanced Accuracy : 0.77105   
##   
## 'Positive' Class : Yes   
##

### 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-fold k-fold cross-validation. Use a random number seed of 1234. To suppress all of the text describing model convergence, add the command: trace = FALSE after verbose = FALSE. Note: This model make take some time to run! Be patient, particularly if you are using an older computer.

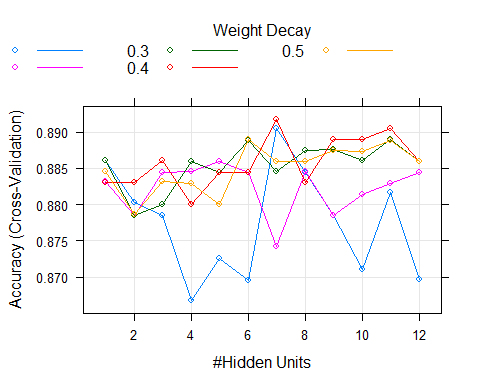
start\_time = Sys.time() #for timing  
fitControl = trainControl(method = "cv",   
 number = 10)  
  
nnetGrid = expand.grid(size = seq(from = 1, to = 12, by = 1), #rule of thumb --> between # of input and # of output layers  
 decay = seq(from = 0.1, to = 0.5, by = 0.1))  
set.seed(1234)  
nnetFit =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

## Time difference of 1.075456 mins

nnetFit

## Neural Network   
##   
## 675 samples  
## 8 predictor  
## 2 classes: 'No', 'Yes'   
##   
## 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.

plot(nnetFit)



### Task 5

Use your model from Task 4 to develop predictions on the training set. Use caret’s confusionMatrix function to evaluate the model quality. Comment on the model quality.

The accuracy of this model has decreased slightky to 90.49%. A difference of 3% from the previous model. The p-value change to a greater degree above .05 listed as 0.08373. The naive model stayed the same in performance. The sensitivity decreased also in this model to 30.91% and specificity remained around the same in value.

predNet = predict(nnetFit, train)  
  
  
confusionMatrix(predNet, train$violator, positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 411 38  
## Yes 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 : 0.30909   
## Specificity : 0.98325   
## Pos Pred Value : 0.70833   
## Neg Pred Value : 0.91537   
## Prevalence : 0.11628   
## Detection Rate : 0.03594   
## Detection Prevalence : 0.05074   
## Balanced Accuracy : 0.64617   
##   
## 'Positive' Class : Yes   
##

### Task 6

Use your model from Task 2 to develop predictions on the testing set. Use the confusionMatrix command to assess and comment on the quality of the model.

The model for the testing set for task 2 is 91.58% accurate indicating a increase in from the naive model of 88.61%. Therefore, the model is slightly better. The model is not less than .05 for p-value. The sensitivity is only at 30.44% and the specificity is at 99.44%.

predNetBasic = predict(nnetBasic, test)  
  
confusionMatrix(predNetBasic, test$violator, positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 178 16  
## Yes 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 : 0.30435   
## Specificity : 0.99441   
## Pos Pred Value : 0.87500   
## Neg Pred Value : 0.91753   
## Prevalence : 0.11386   
## Detection Rate : 0.03465   
## Detection Prevalence : 0.03960   
## Balanced Accuracy : 0.64938   
##   
## 'Positive' Class : Yes   
##

### Task 7

Use your model from Task 4 to develop predictions on the testing set. Use the confusionMatrix command to assess and comment on the quality of the model.

The model for the testing set for task 4 is 88.61% accurate and the naive model is around the same 88.61%. Therefore, the model is the same. The model is not less than .05 for p-value. The sensitivity is only at 13.04% and the specificity is at 98.32%.

predNet = predict(nnetFit, test)  
  
  
confusionMatrix(predNet, test$violator, positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 176 20  
## Yes 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 : 0.13043   
## Specificity : 0.98324   
## Pos Pred Value : 0.50000   
## Neg Pred Value : 0.89796   
## Prevalence : 0.11386   
## Detection Rate : 0.01485   
## Detection Prevalence : 0.02970   
## Balanced Accuracy : 0.55684   
##   
## 'Positive' Class : Yes   
##

### Task 8

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

Task 2

The accuracy for the training set is 93.02% and the accuracy for the testing set is 91.58% only a difference between values of 1.44%. There seems to be about the same accuracy in training and testing set meaning that there is not overfitting in the model.

Task 4

The accuracy for the training set is 90.49% and the accuracy for the testing set is 88.61% only a difference between values of 1.88%. There seems to be about the same accuracy in training and testing set meaning that there is not overfitting in the model.