# Neural Networks

By Katherine Mobley

library(titanic)  
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(mice) #package for imputation

## Loading required package: lattice

##   
## Attaching package: 'mice'

## The following object is masked from 'package:tidyr':  
##   
## complete

## The following objects are masked from 'package:base':  
##   
## cbind, rbind

library(VIM) #visualizing missingness

## Loading required package: colorspace

## Loading required package: grid

## Loading required package: data.table

##   
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':  
##   
## between, first, last

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

## VIM is ready to use.   
## Since version 4.0.0 the GUI is in its own package VIMGUI.  
##   
## Please use the package to use the new (and old) GUI.

## Suggestions and bug-reports can be submitted at: https://github.com/alexkowa/VIM/issues

##   
## Attaching package: 'VIM'

## The following object is masked from 'package:datasets':  
##   
## sleep

library(randomForest) #random forests, using rather than ranger, bug as of Feb 2019 in ranger parameter tuning

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':  
##   
## combine

## The following object is masked from 'package:ggplot2':  
##   
## margin

library(caret) #control model building

##   
## Attaching package: 'caret'

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

library(e1071)  
library(nnet)

Parole = read\_csv("parole (2).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(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,  
"any other state" = "1",  
"Kentucky" = "2",  
"Louisiana" = "3",  
"Virginia" = "4"))  
  
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"))  
  
Parole = Parole %>% mutate(multiple.offenses = as\_factor(as.character(multiple.offenses))) %>%  
mutate(multiple.offenses = fct\_recode(multiple.offenses,  
"incarcerated" = "1",  
"otherwise" = "0"))  
  
Parole = Parole %>% mutate(violator = as\_factor(as.character(violator))) %>%  
mutate(violator = fct\_recode(violator,  
"parolee violated parole" = "1",  
"no violation" = "0"))

# Task 1 - Split Train & Test

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

# Task 2

start\_time = Sys.time() #for timing  
fitControl = trainControl(method = "cv",   
 number = 10)  
  
nnetGrid <- expand.grid(size = 12, decay = 0.1)  
  
set.seed(1234)  
nnetTask2 = train(violator ~ .,   
 train,  
 method = "nnet",  
 tuneGrid = nnetGrid,  
 trControl = fitControl,  
 verbose = FALSE,  
 trace= FALSE)  
  
end\_time = Sys.time()  
end\_time-start\_time

## Time difference of 2.130322 secs

nnetTask2

## Neural Network   
##   
## 473 samples  
## 8 predictor  
## 2 classes: 'no violation', 'parolee violated parole'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 427, 425, 426, 425, 425, 426, ...   
## Resampling results:  
##   
## Accuracy Kappa   
## 0.8729321 0.2839906  
##   
## 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- Predicitions on training Set Task 2

predNetTask2 = predict(nnetTask2, train)  
  
confusionMatrix(predNetTask2, train$violator)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction no violation parolee violated parole  
## no violation 414 16  
## parolee violated parole 4 39  
##   
## Accuracy : 0.9577   
## 95% CI : (0.9355, 0.974)  
## No Information Rate : 0.8837   
## P-Value [Acc > NIR] : 1.25e-08   
##   
## Kappa : 0.7727   
## Mcnemar's Test P-Value : 0.01391   
##   
## Sensitivity : 0.9904   
## Specificity : 0.7091   
## Pos Pred Value : 0.9628   
## Neg Pred Value : 0.9070   
## Prevalence : 0.8837   
## Detection Rate : 0.8753   
## Detection Prevalence : 0.9091   
## Balanced Accuracy : 0.8498   
##   
## 'Positive' Class : no violation   
##

In the above confusion matrix for task 3 the accuracy of the datset is 95.77%. The Naive of the dataset is 88.37%. The senstivity is 99.04% of the dataset. The specificity is 70.91% of the dataset. Compared to the matrix from Task 6 on the testing dataset the specficity numbers are way different. The specificity of testing set is 26.09%.

# Task 4

start\_time = Sys.time() #for timing  
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)  
nnetTask4 = train(violator ~ .,   
 train,  
 method = "nnet",  
 tuneGrid = nnetGrid,  
 trControl = fitControl,  
 verbose = FALSE,  
 trace = FALSE)  
end\_time = Sys.time()  
end\_time-start\_time

## Time difference of 51.36065 secs

nnetTask4

## Neural Network   
##   
## 473 samples  
## 8 predictor  
## 2 classes: 'no violation', 'parolee violated parole'   
##   
## 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 0.8667765 0.2301620  
## 2 0.2 0.8795868 0.2237456  
## 2 0.3 0.8836186 0.2463276  
## 2 0.4 0.8816701 0.2219948  
## 2 0.5 0.8837554 0.1784034  
## 3 0.1 0.8666378 0.2248356  
## 3 0.2 0.8731576 0.2233492  
## 3 0.3 0.8816239 0.2498214  
## 3 0.4 0.8774129 0.2094674  
## 3 0.5 0.8794538 0.1516649  
## 4 0.1 0.8858831 0.3266695  
## 4 0.2 0.8900921 0.3332197  
## 4 0.3 0.8795405 0.2567731  
## 4 0.4 0.8815796 0.2404823  
## 4 0.5 0.8859717 0.2331388  
## 5 0.1 0.8730246 0.2767404  
## 5 0.2 0.8837978 0.2845555  
## 5 0.3 0.8837978 0.2528040  
## 5 0.4 0.8815352 0.2243874  
## 5 0.5 0.8858811 0.2322471  
## 6 0.1 0.8813560 0.3155284  
## 6 0.2 0.8753758 0.2509246  
## 6 0.3 0.8837072 0.2810155  
## 6 0.4 0.8836629 0.2534693  
## 6 0.5 0.8816701 0.2228004  
## 7 0.1 0.8836186 0.3477573  
## 7 0.2 0.8753315 0.2495138  
## 7 0.3 0.8794962 0.2211666  
## 7 0.4 0.8753296 0.1903324  
## 7 0.5 0.8857462 0.2595299  
## 8 0.1 0.8773242 0.3079317  
## 8 0.2 0.8922198 0.3816467  
## 8 0.3 0.8773686 0.2249289  
## 8 0.4 0.8773686 0.2340948  
## 8 0.5 0.8816258 0.2101881  
## 9 0.1 0.8817145 0.3056663  
## 9 0.2 0.8732019 0.2467428  
## 9 0.3 0.8858349 0.2949295  
## 9 0.4 0.8774129 0.2258908  
## 9 0.5 0.8858831 0.2191119  
## 10 0.1 0.8731132 0.2844643  
## 10 0.2 0.8752852 0.2494768  
## 10 0.3 0.8857905 0.2644499  
## 10 0.4 0.8795405 0.2410155  
## 10 0.5 0.8794519 0.1921773  
## 11 0.1 0.8834837 0.3783953  
## 11 0.2 0.8816239 0.3114281  
## 11 0.3 0.8816239 0.2523287  
## 11 0.4 0.8773686 0.2006622  
## 11 0.5 0.8837535 0.2339043  
## 12 0.1 0.8752833 0.3198863  
## 12 0.2 0.8711629 0.2739253  
## 12 0.3 0.8816701 0.2485802  
## 12 0.4 0.8773686 0.1934960  
## 12 0.5 0.8794519 0.2153142  
##   
## 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.

# Task 5

Use model from task 4 to develop predicitions on the training set. Use Confusion Matrix.

predNetTask5 = predict(nnetTask4, train)  
  
confusionMatrix(predNetTask5, train$violator)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction no violation parolee violated parole  
## no violation 408 33  
## parolee violated parole 10 22  
##   
## Accuracy : 0.9091   
## 95% CI : (0.8795, 0.9334)  
## No Information Rate : 0.8837   
## P-Value [Acc > NIR] : 0.0459665   
##   
## Kappa : 0.4595   
## Mcnemar's Test P-Value : 0.0007937   
##   
## Sensitivity : 0.9761   
## Specificity : 0.4000   
## Pos Pred Value : 0.9252   
## Neg Pred Value : 0.6875   
## Prevalence : 0.8837   
## Detection Rate : 0.8626   
## Detection Prevalence : 0.9323   
## Balanced Accuracy : 0.6880   
##   
## 'Positive' Class : no violation   
##

In the above confusion matrix for task 5 the accuracy of the datset is 91.97%. The Naive of the dataset is 88.37%. The senstivity is 98.09% of the dataset. The specificity is 45.45% of the dataset. Compared to the matrix from Task 7 on the testing dataset, the accuracy is 90.59% which is about the same.

# Task 6

Use model from Task 2 to predict on testing set & confusion Matrix.

predNetTask6 = predict(nnetTask2, test)  
  
confusionMatrix(predNetTask6, test$violator)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction no violation parolee violated parole  
## no violation 174 17  
## parolee violated parole 5 6  
##   
## Accuracy : 0.8911   
## 95% CI : (0.8398, 0.9305)  
## No Information Rate : 0.8861   
## P-Value [Acc > NIR] : 0.46721   
##   
## Kappa : 0.3015   
## Mcnemar's Test P-Value : 0.01902   
##   
## Sensitivity : 0.9721   
## Specificity : 0.2609   
## Pos Pred Value : 0.9110   
## Neg Pred Value : 0.5455   
## Prevalence : 0.8861   
## Detection Rate : 0.8614   
## Detection Prevalence : 0.9455   
## Balanced Accuracy : 0.6165   
##   
## 'Positive' Class : no violation   
##

In the above confusion matrix for task 6 the accuracy of the datset is 89.11%. The Naive of the dataset is 88.61%. The senstivity is 97.21% of the dataset. The specificity is 26.09% of the dataset. Compared to the matrix from Task 3 on the training dataset, the accuracy is 95.77% which is about 6% more than the accuracy of testing dataset.

# Task 7

Use model from task 4 to develop predicitions on testing set. Confusion Matrix.

predNetTask7 = predict(nnetTask4, test)  
  
confusionMatrix(predNetTask7, test$violator)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction no violation parolee violated parole  
## no violation 175 19  
## parolee violated parole 4 4  
##   
## Accuracy : 0.8861   
## 95% CI : (0.8341, 0.9264)  
## No Information Rate : 0.8861   
## P-Value [Acc > NIR] : 0.555251   
##   
## Kappa : 0.2117   
## Mcnemar's Test P-Value : 0.003509   
##   
## Sensitivity : 0.9777   
## Specificity : 0.1739   
## Pos Pred Value : 0.9021   
## Neg Pred Value : 0.5000   
## Prevalence : 0.8861   
## Detection Rate : 0.8663   
## Detection Prevalence : 0.9604   
## Balanced Accuracy : 0.5758   
##   
## 'Positive' Class : no violation   
##

In the above test confusion Matric for Task 7 the accuracy of the matrix is 90.59%. The Navie of the dataset is 88.61%. The sensitivity for the dataset is 98.32%. The specificity is 30.43%. Compare this to the training set matrix the sensitivity and naive were almost the same numbers.

# Task 8

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

Both models from Task 2 and Task 4 appear to be overfitting the data. While Task 4 had a lot of extra noise in the dataset it wasn’t an overbearing amount and the data fit well. None of the data seemed to be underfitting because there were more than enough points that fell in line or around the main points in the datasets. If the model was underfitted it would be considered unsuitable. The data is overfitting because there was good preformance on the training data that was evaulated.

plot(nnetTask4)

