## Parameter Selection, Neural Networks, and Ensemble Models

### Anthony Sumter

**Libraries**

#install.packages("tidyverse","caret", "nnet", "rpart", "caretEnsemble", "ranger" )  
options(tidyverse.quiet = TRUE)  
library(tidyverse)  
library(caret)

## Warning: package 'caret' was built under R version 3.6.2

## Loading required package: lattice

##   
## Attaching package: 'caret'

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

library(nnet)  
library(rpart)  
library(caretEnsemble)

## Warning: package 'caretEnsemble' was built under R version 3.6.2

##   
## Attaching package: 'caretEnsemble'

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

library(ranger)

## Warning: package 'ranger' was built under R version 3.6.2

**Dataset**

parole = read.csv("parole.csv")  
parole = parole %>% mutate(male = as\_factor(as.character(male)), race = as\_factor (as.character (race)), state = as\_factor(as.character (state)), crime = as\_factor(as.character (crime)), multiple.offenses = as\_factor(as.character(multiple.offenses)), violator = as\_factor(as.character(violator))) %>%  
mutate(male = fct\_recode(male,"female" = "0","male" = "1")) %>% mutate(race = fct\_recode(race,"White" = "1","Otherwise" = "2"))%>% mutate(state = fct\_recode(state,"OtherState" = "1","Kentucky" = "2", "Louisiana"= "3", "Virgina" = "4"))%>% mutate(crime = fct\_recode(crime,"OtherCrime" = "1", "Larceny" = "2", "DrugRelated" = "3", "DrivingRelated" = "4")) %>% mutate(multiple.offenses = fct\_recode(multiple.offenses,"Otherwise" = "0", "MultipleOffenses" = "1")) %>% mutate(violator = fct\_recode(violator,"Violated" = "1", "NonViolated" = "0"))  
str(parole)

## '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 "OtherState","Kentucky",..: 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 : int 18 12 12 18 12 18 18 12 13 12 ...  
## $ multiple.offenses: Factor w/ 2 levels "Otherwise","MultipleOffenses": 1 1 1 1 1 1 1 1 1 1 ...  
## $ crime : Factor w/ 4 levels "DrivingRelated",..: 1 2 2 3 3 1 2 3 2 4 ...  
## $ violator : Factor w/ 2 levels "NonViolated",..: 1 1 1 1 1 1 1 1 1 1 ...

summary(parole)

## male race age state   
## male :545 White :389 Min. :18.40 OtherState:143   
## female:130 Otherwise:286 1st Qu.:25.35 Kentucky :120   
## Median :33.70 Louisiana : 82   
## Mean :34.51 Virgina :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 MultipleOffenses: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   
## DrivingRelated:101 NonViolated:597   
## DrugRelated :153 Violated : 78   
## OtherCrime :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.

parole = parole %>% drop\_na()  
str(parole)

## '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 "OtherState","Kentucky",..: 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 : int 18 12 12 18 12 18 18 12 13 12 ...  
## $ multiple.offenses: Factor w/ 2 levels "Otherwise","MultipleOffenses": 1 1 1 1 1 1 1 1 1 1 ...  
## $ crime : Factor w/ 4 levels "DrivingRelated",..: 1 2 2 3 3 1 2 3 2 4 ...  
## $ violator : Factor w/ 2 levels "NonViolated",..: 1 1 1 1 1 1 1 1 1 1 ...

set.seed(12345)  
train.rows = createDataPartition(y = parole$violator, p=0.7, list = FALSE)  
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. Hint: Use matrix notation to define x and y and use as.data.frame to convert your x variable to a data frame. This avoids passing a tibble to nnet package and seeing a warning message.

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))  
nnetBasic = train(x=as.data.frame(train[,-9]), y=as.matrix(train$violator),  
 method = "nnet",  
 tuneGrid = nnetGrid,  
 trControl = fitControl,  
 verbose = FALSE,   
 trace = FALSE)  
  
end\_time = Sys.time()  
end\_time-start\_time

## Time difference of 54.14701 secs

#### 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 predictions would be an significant gain of 5% compared to the naive rate of 88% increasing the accuracy of the predictions to 93%. This could indicate that the data is overfitted.

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction NonViolated Violated  
## NonViolated 414 28  
## Violated 4 27  
##   
## Accuracy : 0.9323   
## 95% CI : (0.9058, 0.9533)  
## No Information Rate : 0.8837   
## P-Value [Acc > NIR] : 0.0002879   
##   
## Kappa : 0.5939   
##   
## Mcnemar's Test P-Value : 4.785e-05   
##   
## Sensitivity : 0.49091   
## Specificity : 0.99043   
## Pos Pred Value : 0.87097   
## Neg Pred Value : 0.93665   
## Prevalence : 0.11628   
## Detection Rate : 0.05708   
## Detection Prevalence : 0.06554   
## Balanced Accuracy : 0.74067   
##   
## 'Positive' Class : Violated   
##

#### 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. Hint: Use matrix notation to define x and y and use as.data.frame to convert your x variable to a data frame. This avoids passing a tibble to nnet package and seeing a warning message.

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(x=as.data.frame(train[,-9]), y=as.matrix(train$violator),  
 method = "nnet",  
 trControl = fitControl,  
 tuneGrid = nnetGrid,  
 verbose = FALSE, trace = FALSE)  
  
end\_time = Sys.time()  
end\_time-start\_time

## Time difference of 53.54509 secs

#### 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 predictions would be an marginal gain of 1% compared to the naive rate of 88% increasing the accuracy of the predictions to 89%. This could indicate that the data is not overfitted.

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction NonViolated Violated  
## NonViolated 414 46  
## Violated 4 9  
##   
## Accuracy : 0.8943   
## 95% CI : (0.863, 0.9205)  
## No Information Rate : 0.8837   
## P-Value [Acc > NIR] : 0.2628   
##   
## Kappa : 0.2305   
##   
## Mcnemar's Test P-Value : 6.7e-09   
##   
## Sensitivity : 0.16364   
## Specificity : 0.99043   
## Pos Pred Value : 0.69231   
## Neg Pred Value : 0.90000   
## Prevalence : 0.11628   
## Detection Rate : 0.01903   
## Detection Prevalence : 0.02748   
## Balanced Accuracy : 0.57703   
##   
## 'Positive' Class : Violated   
##

#### 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 predictions would be an significant gain of 3% compared to the naive rate of 88% increasing the accuracy of the predictions to 91%. This could indicate that the data is overfitted.

predNetBasic\_test = predict(nnetBasic, newdata = test)  
confusionMatrix(predNetBasic\_test, test$violator, positive = "Violated")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction NonViolated Violated  
## NonViolated 176 14  
## Violated 3 9  
##   
## Accuracy : 0.9158   
## 95% CI : (0.8687, 0.9502)  
## No Information Rate : 0.8861   
## P-Value [Acc > NIR] : 0.10836   
##   
## Kappa : 0.4732   
##   
## Mcnemar's Test P-Value : 0.01529   
##   
## Sensitivity : 0.39130   
## Specificity : 0.98324   
## Pos Pred Value : 0.75000   
## Neg Pred Value : 0.92632   
## Prevalence : 0.11386   
## Detection Rate : 0.04455   
## Detection Prevalence : 0.05941   
## Balanced Accuracy : 0.68727   
##   
## 'Positive' Class : Violated   
##

#### 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 predictions would be an marginal gain of 2% compared to the naive rate of 88% increasing the accuracy of the predictions to 90%. This could indicate that the data is not overfitted.

predNet\_test = predict(nnetFit, newdata = test)  
confusionMatrix(predNet\_test, test$violator, positive = "Violated")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction NonViolated Violated  
## NonViolated 178 18  
## Violated 1 5  
##   
## Accuracy : 0.9059   
## 95% CI : (0.857, 0.9424)  
## No Information Rate : 0.8861   
## P-Value [Acc > NIR] : 0.2224557   
##   
## Kappa : 0.3124   
##   
## Mcnemar's Test P-Value : 0.0002419   
##   
## Sensitivity : 0.21739   
## Specificity : 0.99441   
## Pos Pred Value : 0.83333   
## Neg Pred Value : 0.90816   
## Prevalence : 0.11386   
## Detection Rate : 0.02475   
## Detection Prevalence : 0.02970   
## Balanced Accuracy : 0.60590   
##   
## 'Positive' Class : Violated   
##

#### Task 8: Comment on whether there appears to be overfitting in one or both of your models from Tasks 2 and 4.

It would appear that the model from task 2 has some signs of overfitting while the model from task 4 does not.

#### Task 9: Build an ensemble (not stacked) model. To save time, use 5 folds in your k-fold cross-validation. Your random number seed should be set to 111. Use matrix notation to define the x and y variables for your model. When creating your model\_list, use glm, ranger, rpart, and nnet models. Hint: Use matrix notation to define x and y and use as.data.frame to convert your x variable to a data frame. This avoids passing a tibble to nnet package and seeing a warning message.These lines of code do some rough parameter tuning for the random forest, classification tree, and neural network models. How correlated are the models in the ensemble? How does the ensemble perform (with regard to AUC) versus the individual models in the ensemble? Be sure to evaluate ensemble model performance on the training and testing sets.

Wasn’t able to get code to run correctly.

control = trainControl(  
 method = "cv",  
 number = 5,  
 savePredictions = "final",  
 classProbs = TRUE,  
 summaryFunction = twoClassSummary  
 )

start\_time = Sys.time()  
set.seed(111)  
#model\_list = caretList(x=as.data.frame(train[,-9]), y=as.matrix(train$violator),  
#metric = "ROC",  
#data = train,  
#trControl= control,   
#methodList = "glm",  
#tuneList=list(rf = caretModelSpec(method="ranger", tuneLength=6)),  
#rpart = caretModelSpec(method="rpart", tuneLength=6),  
#nn = caretModelSpec(method="nnet", tuneLength=6, trace=FALSE))  
  
end\_time = Sys.time ()  
end\_time-start\_time

## Time difference of 0.002027988 secs

#### Task 10: Build a stacked ensemble model. To save time, use 5 folds in your k-fold cross-validation. Your random number seed should be set to 111. Use matrix notation to define the x and y variables for your model. When creating your model\_list, use glm, ranger, rpart, and nnet models. Hint: Use matrix notation to define x and y and use as.data.frame to convert your x variable to a data frame. This avoids passing a tibble to nnet package and seeing a warning message. How does the ensemble perform (with regard to AUC) versus the individual models in the ensemble? How does the model perform on the training and testing sets?

Wasn’t able to get code to run correctly.

#stack = caretStack(  
 #model\_list,  
 #method ="glm",   
 #metric ="ROC",  
 #trControl = trainControl(  
 #method = "cv", #k-fold cross-validation  
 #number = 5, #5 folds  
 #savePredictions = "final",  
 #classProbs = TRUE, #save probabilities  
 #summaryFunction = twoClassSummary #calculate AUC values)