# Module 5 - Parameter, Neural, Ensemble

## Cooper, Sarah

library(tidyverse)

library(caret)

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

library(ranger)

parole <- read\_csv("C:/Users/Sarah/Downloads/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()  
## )

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,  
"Kentucky" = "2",  
"Louisiana" = "3",  
"Virginia" = "4",  
"Other" = "1"))  
  
parole = parole %>% mutate(crime = as\_factor(as.character(crime))) %>%  
mutate(crime = fct\_recode(crime,  
"Larceny" = "2",  
"Drug-related" = "3",  
"Driving-related" = "4",  
"Other" = "1"))  
  
parole = parole %>% mutate(multiple.offenses = as\_factor(as.character(multiple.offenses))) %>%  
mutate(multiple.offenses = fct\_recode(multiple.offenses,  
"MultipleOffenses" = "1",  
"Otherwise" = "0"))  
  
parole = parole %>% mutate(violator = as\_factor(as.character(violator))) %>%  
mutate(violator = fct\_recode(violator,  
"ViolatedParole" = "1",  
"WithoutViolation" = "0"))

# Task 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

fitControl = trainControl(method = "cv", number = 10)  
nnetGrid <- expand.grid(size = 12, decay = 0.1)  
  
set.seed(1234)  
nnetBasic = train(x=as.data.frame(train[,-9]), y=train$violator,  
 method = "nnet",  
 tuneGrid = nnetGrid,  
 trControl = fitControl,  
 verbose = FALSE)

## # weights: 169  
## initial value 318.381598   
## iter 10 value 149.563206  
## iter 20 value 115.702460  
## iter 30 value 107.032881  
## iter 40 value 103.879737  
## iter 50 value 100.474141  
## iter 60 value 98.961482  
## iter 70 value 95.715548  
## iter 80 value 94.644949  
## iter 90 value 94.153735  
## iter 100 value 93.246545  
## final value 93.246545   
## stopped after 100 iterations  
## # weights: 169  
## initial value 429.257563   
## iter 10 value 162.286275  
## iter 20 value 145.852424  
## iter 30 value 119.172361  
## iter 40 value 108.892614  
## iter 50 value 105.381816  
## iter 60 value 101.469496  
## iter 70 value 97.700591  
## iter 80 value 94.455165  
## iter 90 value 92.805089  
## iter 100 value 90.981925  
## final value 90.981925   
## stopped after 100 iterations  
## # weights: 169  
## initial value 707.799151   
## iter 10 value 153.643944  
## iter 20 value 131.163955  
## iter 30 value 110.306740  
## iter 40 value 106.370630  
## iter 50 value 101.063320  
## iter 60 value 96.199685  
## iter 70 value 94.397390  
## iter 80 value 93.154987  
## iter 90 value 91.960124  
## iter 100 value 90.142826  
## final value 90.142826   
## stopped after 100 iterations  
## # weights: 169  
## initial value 587.486447   
## iter 10 value 140.720802  
## iter 20 value 121.606152  
## iter 30 value 113.562155  
## iter 40 value 108.765878  
## iter 50 value 105.662900  
## iter 60 value 102.739442  
## iter 70 value 100.609054  
## iter 80 value 96.803612  
## iter 90 value 94.563714  
## iter 100 value 89.613192  
## final value 89.613192   
## stopped after 100 iterations  
## # weights: 169  
## initial value 563.879050   
## iter 10 value 144.792069  
## iter 20 value 119.407763  
## iter 30 value 108.669310  
## iter 40 value 102.311539  
## iter 50 value 97.587587  
## iter 60 value 96.340954  
## iter 70 value 95.797629  
## iter 80 value 93.563977  
## iter 90 value 91.712850  
## iter 100 value 90.204297  
## final value 90.204297   
## stopped after 100 iterations  
## # weights: 169  
## initial value 311.708299   
## iter 10 value 157.271630  
## iter 20 value 150.535694  
## iter 30 value 131.196105  
## iter 40 value 116.124982  
## iter 50 value 102.688436  
## iter 60 value 95.253266  
## iter 70 value 91.075599  
## iter 80 value 88.076533  
## iter 90 value 86.009701  
## iter 100 value 84.829548  
## final value 84.829548   
## stopped after 100 iterations  
## # weights: 169  
## initial value 203.929078   
## iter 10 value 150.404390  
## iter 20 value 138.309780  
## iter 30 value 116.380250  
## iter 40 value 105.638724  
## iter 50 value 101.155571  
## iter 60 value 99.406937  
## iter 70 value 97.291986  
## iter 80 value 93.872920  
## iter 90 value 88.386611  
## iter 100 value 84.232078  
## final value 84.232078   
## stopped after 100 iterations  
## # weights: 169  
## initial value 652.330640   
## iter 10 value 159.525631  
## iter 20 value 135.428775  
## iter 30 value 120.280488  
## iter 40 value 108.020108  
## iter 50 value 106.472799  
## iter 60 value 105.356742  
## iter 70 value 101.650213  
## iter 80 value 93.661129  
## iter 90 value 90.245067  
## iter 100 value 88.293133  
## final value 88.293133   
## stopped after 100 iterations  
## # weights: 169  
## initial value 429.974077   
## iter 10 value 150.384382  
## iter 20 value 128.254862  
## iter 30 value 113.542432  
## iter 40 value 107.348392  
## iter 50 value 101.449929  
## iter 60 value 99.887789  
## iter 70 value 99.472452  
## iter 80 value 97.774584  
## iter 90 value 94.630331  
## iter 100 value 89.892954  
## final value 89.892954   
## stopped after 100 iterations  
## # weights: 169  
## initial value 289.112988   
## iter 10 value 146.446410  
## iter 20 value 127.272413  
## iter 30 value 116.961423  
## iter 40 value 109.875456  
## iter 50 value 106.911919  
## iter 60 value 105.908477  
## iter 70 value 104.137481  
## iter 80 value 99.693953  
## iter 90 value 96.441609  
## iter 100 value 94.474981  
## final value 94.474981   
## stopped after 100 iterations  
## # weights: 169  
## initial value 173.334664   
## iter 10 value 161.787268  
## iter 20 value 139.895077  
## iter 30 value 117.875556  
## iter 40 value 113.366291  
## iter 50 value 110.330569  
## iter 60 value 104.832400  
## iter 70 value 99.026889  
## iter 80 value 96.150357  
## iter 90 value 94.841973  
## iter 100 value 93.633065  
## final value 93.633065   
## stopped after 100 iterations

nnetBasic

## Neural Network   
##   
## 473 samples  
## 8 predictor  
## 2 classes: 'WithoutViolation', 'ViolatedParole'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 425, 426, 425, 426, 425, 427, ...   
## Resampling results:  
##   
## Accuracy Kappa   
## 0.8733522 0.2936362  
##   
## 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

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction WithoutViolation ViolatedParole  
## WithoutViolation 416 20  
## ViolatedParole 2 35  
##   
## Accuracy : 0.9535   
## 95% CI : (0.9304, 0.9706)  
## No Information Rate : 0.8837   
## P-Value [Acc > NIR] : 1.009e-07   
##   
## Kappa : 0.7362   
##   
## Mcnemar's Test P-Value : 0.0002896   
##   
## Sensitivity : 0.63636   
## Specificity : 0.99522   
## Pos Pred Value : 0.94595   
## Neg Pred Value : 0.95413   
## Prevalence : 0.11628   
## Detection Rate : 0.07400   
## Detection Prevalence : 0.07822   
## Balanced Accuracy : 0.81579   
##   
## 'Positive' Class : ViolatedParole   
##

*This model has great accuracy at 95% and a relatively close No Information Rate of 88%. These numbers indicate that the model is reliable.*

# Task 4

fitControl = trainControl(method = "cv", number = 10)  
nnetGrid = expand.grid(size = seq(from = 1, 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=train$violator,  
 method = "nnet",  
 trControl = fitControl,  
 tuneGrid = nnetGrid,  
 verbose = FALSE,  
 trace = FALSE)

# Task 5

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction WithoutViolation ViolatedParole  
## WithoutViolation 412 48  
## ViolatedParole 6 7  
##   
## Accuracy : 0.8858   
## 95% CI : (0.8537, 0.9131)  
## No Information Rate : 0.8837   
## P-Value [Acc > NIR] : 0.4787   
##   
## Kappa : 0.1689   
##   
## Mcnemar's Test P-Value : 2.414e-08   
##   
## Sensitivity : 0.12727   
## Specificity : 0.98565   
## Pos Pred Value : 0.53846   
## Neg Pred Value : 0.89565   
## Prevalence : 0.11628   
## Detection Rate : 0.01480   
## Detection Prevalence : 0.02748   
## Balanced Accuracy : 0.55646   
##   
## 'Positive' Class : ViolatedParole   
##

*Unlike our previous model, this representation slipped to 89% in Accuracy.*

# Task 6

predTestBasic = predict(nnetBasic, test)  
confusionMatrix(predTestBasic, test$violator, positive = "ViolatedParole")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction WithoutViolation ViolatedParole  
## WithoutViolation 171 14  
## ViolatedParole 8 9  
##   
## Accuracy : 0.8911   
## 95% CI : (0.8398, 0.9305)  
## No Information Rate : 0.8861   
## P-Value [Acc > NIR] : 0.4672   
##   
## Kappa : 0.3911   
##   
## Mcnemar's Test P-Value : 0.2864   
##   
## Sensitivity : 0.39130   
## Specificity : 0.95531   
## Pos Pred Value : 0.52941   
## Neg Pred Value : 0.92432   
## Prevalence : 0.11386   
## Detection Rate : 0.04455   
## Detection Prevalence : 0.08416   
## Balanced Accuracy : 0.67331   
##   
## 'Positive' Class : ViolatedParole   
##

*Still clinging to a high Accuracy rate and close No Information Rate.*

# Task 7

predTestNet = predict(nnetFit, test)  
confusionMatrix(predTestNet, test$violator, positive = "ViolatedParole")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction WithoutViolation ViolatedParole  
## WithoutViolation 178 18  
## ViolatedParole 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 : ViolatedParole   
##

*Again, the Accuracy and No Information Rate remain high as in the three previous models.*

# Task 8

*I do not suspect that there is overfitting present in the models created in Task 2 and Task 4. The models are hardly distinguishable from one another because they are each performing well. A red flag would raise only if the testing models were vastly different than the training models.*

# Task 9

set.seed(111)  
control = trainControl(method = "cv",number = 5,  
 savePredictions = "final",classProbs = TRUE, summaryFunction = twoClassSummary)

model\_list = caretList(x=as.data.frame(train[,-9]), y=train$violator,  
 metric = "ROC",  
 trControl= control,   
 methodList = c("glm"),  
 tuneList= list(  
 rf = caretModelSpec(method="ranger", tuneLength=6),  
 rpart = caretModelSpec(method="rpart", tuneLength=6),  
 nn = caretModelSpec(method="nnet", tuneLength=6, trace=FALSE)))

modelCor(resamples(model\_list))

## rf rpart nn glm  
## rf 1.00000000 0.03495397 -0.3102612 -0.5757076  
## rpart 0.03495397 1.00000000 -0.1123056 0.2311165  
## nn -0.31026118 -0.11230565 1.0000000 0.8746848  
## glm -0.57570755 0.23111654 0.8746848 1.0000000

*These models do not appear to be correlated.*

ensemble = caretEnsemble(  
 model\_list,   
 metric="ROC",  
 trControl=trainControl(  
 method = "cv",  
 number= 5,  
 summaryFunction=twoClassSummary,  
 classProbs=TRUE  
 ))

summary(ensemble)

## The following models were ensembled: rf, rpart, nn, glm   
## They were weighted:   
## 2.46 -4.3158 3.1407 -1.7002 -2.7489  
## The resulting ROC is: 0.8234  
## The fit for each individual model on the ROC is:   
## method ROC ROCSD  
## rf 0.8167266 0.01613172  
## rpart 0.7507719 0.12634167  
## nn 0.8306994 0.04628532  
## glm 0.8183148 0.02622670

*According to our output, the nn model is preferred over the rf, rpart, and glm due to the .83 ROC.*

pred\_ensemble = predict(ensemble, train, type = "raw")  
confusionMatrix(pred\_ensemble,train$violator)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction WithoutViolation ViolatedParole  
## WithoutViolation 418 10  
## ViolatedParole 0 45  
##   
## Accuracy : 0.9789   
## 95% CI : (0.9615, 0.9898)  
## No Information Rate : 0.8837   
## P-Value [Acc > NIR] : 1.053e-14   
##   
## Kappa : 0.8883   
##   
## Mcnemar's Test P-Value : 0.004427   
##   
## Sensitivity : 1.0000   
## Specificity : 0.8182   
## Pos Pred Value : 0.9766   
## Neg Pred Value : 1.0000   
## Prevalence : 0.8837   
## Detection Rate : 0.8837   
## Detection Prevalence : 0.9049   
## Balanced Accuracy : 0.9091   
##   
## 'Positive' Class : WithoutViolation  
##

pred\_ensemble\_test = predict(ensemble, test, type = "raw")  
confusionMatrix(pred\_ensemble\_test,test$violator)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction WithoutViolation ViolatedParole  
## WithoutViolation 177 18  
## ViolatedParole 2 5  
##   
## Accuracy : 0.901   
## 95% CI : (0.8512, 0.9385)  
## No Information Rate : 0.8861   
## P-Value [Acc > NIR] : 0.2968298   
##   
## Kappa : 0.2959   
##   
## Mcnemar's Test P-Value : 0.0007962   
##   
## Sensitivity : 0.9888   
## Specificity : 0.2174   
## Pos Pred Value : 0.9077   
## Neg Pred Value : 0.7143   
## Prevalence : 0.8861   
## Detection Rate : 0.8762   
## Detection Prevalence : 0.9653   
## Balanced Accuracy : 0.6031   
##   
## 'Positive' Class : WithoutViolation  
##

*Using the train set would be far preferred in this ensemble as it performed at 98% accuracy (versus 90% accuracy in the test set).*

# Task 10

stack = caretStack(  
 model\_list,  
 method ="glm",  
 metric ="ROC",  
 trControl = trainControl(  
 method = "cv",  
 number = 5,  
 savePredictions = "final",  
 classProbs = TRUE,  
 summaryFunction = twoClassSummary))  
  
print(stack)

## A glm ensemble of 4 base models: rf, rpart, nn, glm  
##   
## Ensemble results:  
## Generalized Linear Model   
##   
## 473 samples  
## 4 predictor  
## 2 classes: 'WithoutViolation', 'ViolatedParole'   
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 378, 379, 378, 378, 379   
## Resampling results:  
##   
## ROC Sens Spec   
## 0.8313488 0.9665232 0.09090909

*The ROC in this stacked model isn’t terribly different from the nn ensemble model as they each equal .83.*

pred\_stack = predict(stack, train, type = "raw")  
confusionMatrix(pred\_stack,train$violator)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction WithoutViolation ViolatedParole  
## WithoutViolation 418 10  
## ViolatedParole 0 45  
##   
## Accuracy : 0.9789   
## 95% CI : (0.9615, 0.9898)  
## No Information Rate : 0.8837   
## P-Value [Acc > NIR] : 1.053e-14   
##   
## Kappa : 0.8883   
##   
## Mcnemar's Test P-Value : 0.004427   
##   
## Sensitivity : 1.0000   
## Specificity : 0.8182   
## Pos Pred Value : 0.9766   
## Neg Pred Value : 1.0000   
## Prevalence : 0.8837   
## Detection Rate : 0.8837   
## Detection Prevalence : 0.9049   
## Balanced Accuracy : 0.9091   
##   
## 'Positive' Class : WithoutViolation  
##

pred\_stack\_test = predict(stack, test, type = "raw")  
confusionMatrix(pred\_stack\_test,test$violator)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction WithoutViolation ViolatedParole  
## WithoutViolation 177 18  
## ViolatedParole 2 5  
##   
## Accuracy : 0.901   
## 95% CI : (0.8512, 0.9385)  
## No Information Rate : 0.8861   
## P-Value [Acc > NIR] : 0.2968298   
##   
## Kappa : 0.2959   
##   
## Mcnemar's Test P-Value : 0.0007962   
##   
## Sensitivity : 0.9888   
## Specificity : 0.2174   
## Pos Pred Value : 0.9077   
## Neg Pred Value : 0.7143   
## Prevalence : 0.8861   
## Detection Rate : 0.8762   
## Detection Prevalence : 0.9653   
## Balanced Accuracy : 0.6031   
##   
## 'Positive' Class : WithoutViolation  
##

*The stacked model agrees with the ensemble model in that the train set is going to be a more accurate model to process.*