# Matthew Hawkins

## BAN 502: Module 5, Assignment 1

library("tidyverse")  
library("caret")

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

library("nnet")

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

library("caretEnsemble")

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

library("rpart")  
library("ranger")

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

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()  
## )

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,"other"="1","Kentucky"="2","Louisiana"="3","Virginia"="4"))  
parole = parole %>% mutate(crime = as\_factor(as.character(crime))) %>% mutate(crime = fct\_recode(crime,"other"="1","larceny"="2","drug-related"="3","driving-related"="4"))  
parole = parole %>% mutate(multiple.offenses = as\_factor(as.character(multiple.offenses))) %>% mutate(multiple.offenses = fct\_recode(multiple.offenses,"multiple"="1","otherwise"="0"))  
parole = parole %>% mutate(violator = as\_factor(as.character(violator))) %>% mutate(violator = fct\_recode(violator,"Yes"="1","No"="0"))

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

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,verbose = FALSE, trace = FALSE)

nnetBasic

## Neural Network   
##   
## 473 samples  
## 8 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 425, 426, 426, 426, 425, 427, ...   
## Resampling results:  
##   
## Accuracy Kappa   
## 0.8814157 0.2952919  
##   
## 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="Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 416 20  
## Yes 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 : Yes   
##

This model appears to be of high quality. We have an accuracy of 0.95 and a p-value well below 0.05.

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

## Time difference of 36.14825 secs

nnetFit

## Neural Network   
##   
## 473 samples  
## 8 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 425, 426, 426, 426, 425, 427, ...   
## Resampling results across tuning parameters:  
##   
## size decay Accuracy Kappa   
## 1 0.1 0.8858484 0.21364368  
## 1 0.2 0.8879760 0.17770650  
## 1 0.3 0.8837650 0.09636117  
## 1 0.4 0.8837650 0.05107117  
## 1 0.5 0.8838113 0.00000000  
## 2 0.1 0.8795078 0.26403357  
## 2 0.2 0.8690506 0.18631976  
## 2 0.3 0.8878874 0.17546482  
## 2 0.4 0.8943166 0.21880618  
## 2 0.5 0.8879780 0.10300595  
## 3 0.1 0.8732135 0.18994613  
## 3 0.2 0.8772471 0.17523506  
## 3 0.3 0.8857135 0.18398518  
## 3 0.4 0.8815487 0.12846430  
## 3 0.5 0.8836764 0.11255848  
## 4 0.1 0.8730381 0.18945113  
## 4 0.2 0.8689138 0.17869842  
## 4 0.3 0.8710415 0.14630237  
## 4 0.4 0.8815468 0.17078653  
## 4 0.5 0.8836764 0.13743579  
## 5 0.1 0.8666088 0.20464243  
## 5 0.2 0.8731691 0.19617171  
## 5 0.3 0.8794191 0.14729995  
## 5 0.4 0.8773821 0.13730301  
## 5 0.5 0.8837207 0.17364964  
## 6 0.1 0.8877525 0.30013311  
## 6 0.2 0.8793305 0.19716524  
## 6 0.3 0.8752525 0.15696256  
## 6 0.4 0.8815930 0.16819853  
## 6 0.5 0.8857597 0.16113950  
## 7 0.1 0.8602239 0.16462723  
## 7 0.2 0.8667399 0.17753118  
## 7 0.3 0.8752525 0.14805484  
## 7 0.4 0.8815025 0.16800826  
## 7 0.5 0.8899707 0.19055200  
## 8 0.1 0.8813676 0.25052109  
## 8 0.2 0.8771566 0.22607251  
## 8 0.3 0.8856711 0.18514371  
## 8 0.4 0.8815930 0.15016306  
## 8 0.5 0.8858040 0.17957557  
## 9 0.1 0.8646122 0.17434714  
## 9 0.2 0.8793286 0.24723544  
## 9 0.3 0.8752525 0.17392059  
## 9 0.4 0.8794635 0.16486060  
## 9 0.5 0.8837207 0.17364964  
## 10 0.1 0.8645236 0.23684983  
## 10 0.2 0.8773801 0.18583794  
## 10 0.3 0.8794191 0.16816524  
## 10 0.4 0.8794191 0.16533542  
## 10 0.5 0.8815468 0.17078653  
## 11 0.1 0.8707273 0.20335426  
## 11 0.2 0.8751638 0.21379387  
## 11 0.3 0.8730362 0.16516894  
## 11 0.4 0.8752081 0.15721146  
## 11 0.5 0.8836764 0.15386677  
## 12 0.1 0.8793729 0.29635686  
## 12 0.2 0.8857578 0.26362706  
## 12 0.3 0.8795540 0.18078550  
## 12 0.4 0.8794635 0.16486060  
## 12 0.5 0.8837207 0.17142742  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were size = 2 and decay = 0.4.

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 415 47  
## Yes 3 8  
##   
## Accuracy : 0.8943   
## 95% CI : (0.863, 0.9205)  
## No Information Rate : 0.8837   
## P-Value [Acc > NIR] : 0.2628   
##   
## Kappa : 0.2119   
##   
## Mcnemar's Test P-Value : 1.193e-09   
##   
## Sensitivity : 0.14545   
## Specificity : 0.99282   
## Pos Pred Value : 0.72727   
## Neg Pred Value : 0.89827   
## Prevalence : 0.11628   
## Detection Rate : 0.01691   
## Detection Prevalence : 0.02326   
## Balanced Accuracy : 0.56914   
##   
## 'Positive' Class : Yes   
##

This model is not quite as good of a fit as the first one, but still a good model. The accuracy is slightly lower at 0.89, but the biggest difference is in the p-value which is now above 0.05.

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 171 14  
## Yes 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 : Yes   
##

When used on our test dataset, this model still gives us a high accuracy (0.89). This is just slightly lower than the train dataset, so I think this model still fits well.

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 179 20  
## Yes 0 3  
##   
## Accuracy : 0.901   
## 95% CI : (0.8512, 0.9385)  
## No Information Rate : 0.8861   
## P-Value [Acc > NIR] : 0.2968   
##   
## Kappa : 0.21   
##   
## Mcnemar's Test P-Value : 2.152e-05   
##   
## Sensitivity : 0.13043   
## Specificity : 1.00000   
## Pos Pred Value : 1.00000   
## Neg Pred Value : 0.89950   
## Prevalence : 0.11386   
## Detection Rate : 0.01485   
## Detection Prevalence : 0.01485   
## Balanced Accuracy : 0.56522   
##   
## 'Positive' Class : Yes   
##

This model also performs well on our test dataset, giving us an accuracy of 0.9.

Neither model appears to be overfitted to the train dataset. Both models were still highly accurate when used on the test dataset.

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

set.seed(111)  
model\_list = caretList(violator~.,data=train,metric="ROC",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)))

## Warning in trControlCheck(x = trControl, y = target): indexes not defined  
## in trControl. Attempting to set them ourselves, so each model in the  
## ensemble will have the same resampling indexes.

modelCor(resamples(model\_list))

## rf rpart nn glm  
## rf 1.0000000 0.5472800 -0.1439831 0.3388614  
## rpart 0.5472800 1.0000000 -0.6957475 -0.4400857  
## nn -0.1439831 -0.6957475 1.0000000 0.5363745  
## glm 0.3388614 -0.4400857 0.5363745 1.0000000

The models show some moderate to strong correlation, the strongest being between nn and rpart (-0.7).

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:   
## 2.5718 -3.7948 1.9437 -0.1156 -3.7441  
## The resulting ROC is: 0.8163  
## The fit for each individual model on the ROC is:   
## method ROC ROCSD  
## rf 0.8252869 0.03943426  
## rpart 0.6713712 0.04964340  
## nn 0.8458796 0.02699027  
## glm 0.8386507 0.03410036

The ensemble performs about as well as the individual models. The ensemble ROC (0.82) is better than rpart (0.67) and on par with rf (0.83), nn (0.85), and glm (0.84).

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 418 19  
## Yes 0 36  
##   
## Accuracy : 0.9598   
## 95% CI : (0.938, 0.9756)  
## No Information Rate : 0.8837   
## P-Value [Acc > NIR] : 4.082e-09   
##   
## Kappa : 0.7701   
##   
## Mcnemar's Test P-Value : 3.636e-05   
##   
## Sensitivity : 1.0000   
## Specificity : 0.6545   
## Pos Pred Value : 0.9565   
## Neg Pred Value : 1.0000   
## Prevalence : 0.8837   
## Detection Rate : 0.8837   
## Detection Prevalence : 0.9239   
## Balanced Accuracy : 0.8273   
##   
## 'Positive' Class : No   
##

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 176 17  
## Yes 3 6  
##   
## Accuracy : 0.901   
## 95% CI : (0.8512, 0.9385)  
## No Information Rate : 0.8861   
## P-Value [Acc > NIR] : 0.29683   
##   
## Kappa : 0.3322   
##   
## Mcnemar's Test P-Value : 0.00365   
##   
## Sensitivity : 0.9832   
## Specificity : 0.2609   
## Pos Pred Value : 0.9119   
## Neg Pred Value : 0.6667   
## Prevalence : 0.8861   
## Detection Rate : 0.8713   
## Detection Prevalence : 0.9554   
## Balanced Accuracy : 0.6221   
##   
## 'Positive' Class : No   
##

The ensemble model has a very high accuracy on the train dataset, at 0.96, and a p-value well below 0.05. While the accuracy is slightly lower for the test dataset, it is still quite high 0.9. I would consider this a good model.

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: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 379, 379, 378, 378, 378   
## Resampling results:  
##   
## ROC Sens Spec   
## 0.8231315 0.9689616 0.2363636

summary(stack)

##   
## Call:  
## NULL  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.7183 -0.3741 -0.3084 -0.2861 2.5607   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 2.5718 0.6742 3.815 0.000136 \*\*\*  
## rf -3.7948 1.2206 -3.109 0.001877 \*\*   
## rpart 1.9437 1.0862 1.790 0.073526 .   
## nn -0.1156 1.2409 -0.093 0.925801   
## glm -3.7441 1.3944 -2.685 0.007251 \*\*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 340.04 on 472 degrees of freedom  
## Residual deviance: 268.71 on 468 degrees of freedom  
## AIC: 278.71  
##   
## Number of Fisher Scoring iterations: 5

The stacked method gave us a nearly identical ROC value as the ensemble did (0.82).

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 418 19  
## Yes 0 36  
##   
## Accuracy : 0.9598   
## 95% CI : (0.938, 0.9756)  
## No Information Rate : 0.8837   
## P-Value [Acc > NIR] : 4.082e-09   
##   
## Kappa : 0.7701   
##   
## Mcnemar's Test P-Value : 3.636e-05   
##   
## Sensitivity : 1.0000   
## Specificity : 0.6545   
## Pos Pred Value : 0.9565   
## Neg Pred Value : 1.0000   
## Prevalence : 0.8837   
## Detection Rate : 0.8837   
## Detection Prevalence : 0.9239   
## Balanced Accuracy : 0.8273   
##   
## 'Positive' Class : No   
##

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 176 17  
## Yes 3 6  
##   
## Accuracy : 0.901   
## 95% CI : (0.8512, 0.9385)  
## No Information Rate : 0.8861   
## P-Value [Acc > NIR] : 0.29683   
##   
## Kappa : 0.3322   
##   
## Mcnemar's Test P-Value : 0.00365   
##   
## Sensitivity : 0.9832   
## Specificity : 0.2609   
## Pos Pred Value : 0.9119   
## Neg Pred Value : 0.6667   
## Prevalence : 0.8861   
## Detection Rate : 0.8713   
## Detection Prevalence : 0.9554   
## Balanced Accuracy : 0.6221   
##   
## 'Positive' Class : No   
##

Again we see nearly identical results on the stack method as we did with the ensemble method. The accuracy for the train dataset is 0.96 and 0.9 for the test dataset.