library("tidyverse")

## ── Attaching packages ─────────────────────────────────────────────────────── tidyverse 1.3.0 ──

## ✓ ggplot2 3.2.1 ✓ purrr 0.3.3  
## ✓ tibble 2.1.3 ✓ dplyr 0.8.4  
## ✓ tidyr 1.0.2 ✓ stringr 1.4.0  
## ✓ readr 1.3.1 ✓ 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")  
library("rpart")  
library("caretEnsemble")

##   
## Attaching package: 'caretEnsemble'

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

library("ranger")  
library("e1071")

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,"female" = "0", "male" = "1"))  
parole = parole %>% mutate(race = as.factor(as.character(race))) %>%  
 mutate(race = fct\_recode(race, "white" = "1", "nonwhite" = "2"))  
parole = parole %>% mutate(state = as.factor(as.character(state))) %>%  
 mutate(state = fct\_recode(state, "OtherState" = "1", "Kentucky" = "2", "Louisiana" = "3", "Virginia" = "4"))  
parole = parole %>% mutate(crime = as.factor(as.character(crime))) %>%  
 mutate(crime = fct\_recode(crime, "OtherCrime" = "1", "larceny" = "2", "drugs" = "3", "driving" = "4"))  
parole = parole %>% mutate(multiple.offenses = as.factor(as.character(multiple.offenses))) %>%  
 mutate(multiple.offenses = fct\_recode(multiple.offenses, "SingleOffense" = "0", "MultiOffenses" = "1"))  
parole = parole %>% mutate(violator = as.factor(as.character(violator))) %>%  
 mutate(violator = fct\_recode(violator, "NoParoleViolation" = "0", "ParoleViolation" = "1"))

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

start\_time = Sys.time()  
x =as.matrix(train)   
y=as.data.frame(train$violator)  
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)  
end\_time = Sys.time()  
start\_time - end\_time

## Time difference of -1.583406 secs

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction NoParoleViolation ParoleViolation  
## NoParoleViolation 416 22  
## ParoleViolation 2 33  
##   
## Accuracy : 0.9493   
## 95% CI : (0.9254, 0.9672)  
## No Information Rate : 0.8837   
## P-Value [Acc > NIR] : 6.784e-07   
##   
## Kappa : 0.7068   
##   
## Mcnemar's Test P-Value : 0.0001052   
##   
## Sensitivity : 0.60000   
## Specificity : 0.99522   
## Pos Pred Value : 0.94286   
## Neg Pred Value : 0.94977   
## Prevalence : 0.11628   
## Detection Rate : 0.06977   
## Detection Prevalence : 0.07400   
## Balanced Accuracy : 0.79761   
##   
## 'Positive' Class : ParoleViolation   
##

The accuracy rating is 95% which is good the test p-value is below 0.05. It seems though the sensitivity rating was compromised at 60% and specificity is good at 99.5%.

start\_time = Sys.time()  
x =as.matrix(train)   
y=as.data.frame(train$violator)  
fitControl2=trainControl(method = "cv", number = 10)  
nnetGrid2 = expand.grid(size = seq(from = 1, to = 12, by = 1),  
 decay=seq(from = 0.1, to = 0.5, by = 0.1))  
set.seed(1234)  
nnetBasic2 = train(violator ~ .,train,   
 method = "nnet",   
 tuneGrid = nnetGrid2,   
 trControl = fitControl2,   
 verbose = FALSE,   
 trace = FALSE)  
end\_time = Sys.time()  
start\_time - end\_time

## Time difference of -32.66953 secs

predNetBasic2 = predict(nnetBasic2, train)  
confusionMatrix(predNetBasic2, train$violator, positive = "ParoleViolation")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction NoParoleViolation ParoleViolation  
## NoParoleViolation 410 24  
## ParoleViolation 8 31  
##   
## Accuracy : 0.9323   
## 95% CI : (0.9058, 0.9533)  
## No Information Rate : 0.8837   
## P-Value [Acc > NIR] : 0.0002879   
##   
## Kappa : 0.6232   
##   
## Mcnemar's Test P-Value : 0.0080099   
##   
## Sensitivity : 0.56364   
## Specificity : 0.98086   
## Pos Pred Value : 0.79487   
## Neg Pred Value : 0.94470   
## Prevalence : 0.11628   
## Detection Rate : 0.06554   
## Detection Prevalence : 0.08245   
## Balanced Accuracy : 0.77225   
##   
## 'Positive' Class : ParoleViolation   
##

The accuracy rating is 93% which is not as good as the training model. The sensitivity rating is 56% which is lower than the training model.

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction NoParoleViolation ParoleViolation  
## NoParoleViolation 172 15  
## ParoleViolation 7 8  
##   
## Accuracy : 0.8911   
## 95% CI : (0.8398, 0.9305)  
## No Information Rate : 0.8861   
## P-Value [Acc > NIR] : 0.4672   
##   
## Kappa : 0.3639   
##   
## Mcnemar's Test P-Value : 0.1356   
##   
## Sensitivity : 0.34783   
## Specificity : 0.96089   
## Pos Pred Value : 0.53333   
## Neg Pred Value : 0.91979   
## Prevalence : 0.11386   
## Detection Rate : 0.03960   
## Detection Prevalence : 0.07426   
## Balanced Accuracy : 0.65436   
##   
## 'Positive' Class : ParoleViolation   
##

The accuracy is 89% which isn’t too bad but not as good as the training model. The sensitivity rate is very low at 35%. This testing model is not as good as the training model.

predNetBasic2 = predict(nnetBasic2, test)  
confusionMatrix(predNetBasic2, test$violator, positive = "ParoleViolation")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction NoParoleViolation ParoleViolation  
## NoParoleViolation 171 16  
## ParoleViolation 8 7  
##   
## Accuracy : 0.8812   
## 95% CI : (0.8284, 0.9224)  
## No Information Rate : 0.8861   
## P-Value [Acc > NIR] : 0.6396   
##   
## Kappa : 0.306   
##   
## Mcnemar's Test P-Value : 0.1530   
##   
## Sensitivity : 0.30435   
## Specificity : 0.95531   
## Pos Pred Value : 0.46667   
## Neg Pred Value : 0.91444   
## Prevalence : 0.11386   
## Detection Rate : 0.03465   
## Detection Prevalence : 0.07426   
## Balanced Accuracy : 0.62983   
##   
## 'Positive' Class : ParoleViolation   
##

The accuracy is 88% which is similar to the task7. The sensitivity rate is low at 30% and the p-value is higher than the previous task.

There seems to be overfitting in task 2 as the p-value approaches 0.

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

model\_list = caretList(x =as.data.frame(train[,-9]),   
y=as.matrix(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)  
))

## 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.

as.data.frame(predict(model\_list, newdata = head(train)))

## rf rpart nn glm  
## 1 0.9977569 0.9097222 0.9717593 0.9520893  
## 2 0.9773333 0.9097222 0.9941539 0.8526230  
## 3 0.9880000 0.9097222 0.9740553 0.8616431  
## 4 0.9916236 0.9097222 0.9708143 0.9268397  
## 5 0.8802433 0.9097222 0.9949586 0.9233879  
## 6 0.9667333 0.9097222 0.8564004 0.7418396

modelCor(resamples(model\_list))

## rf rpart nn glm  
## rf 1.0000000 0.4459720 0.8346676 0.9757070  
## rpart 0.4459720 1.0000000 0.5551316 0.3148552  
## nn 0.8346676 0.5551316 1.0000000 0.6984932  
## glm 0.9757070 0.3148552 0.6984932 1.0000000

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:   
## 2.8955 -1.6712 1.5126 -2.6324 -3.2857  
## The resulting ROC is: 0.8654  
## The fit for each individual model on the ROC is:   
## method ROC ROCSD  
## rf 0.8434843 0.05254430  
## rpart 0.7425207 0.13368026  
## nn 0.8499713 0.05425476  
## glm 0.8327127 0.06819457

The models are pretty correlated except the rpart it’s a little lower than the others. The ensemble’s ROC is better than the invdividual models with an ROC of 0.8546. It is close to the other models but we could have just used the ensemble model.

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction NoParoleViolation ParoleViolation  
## NoParoleViolation 415 23  
## ParoleViolation 3 32  
##   
## Accuracy : 0.945   
## 95% CI : (0.9205, 0.9638)  
## No Information Rate : 0.8837   
## P-Value [Acc > NIR] : 3.851e-06   
##   
## Kappa : 0.6824   
##   
## Mcnemar's Test P-Value : 0.0001944   
##   
## Sensitivity : 0.58182   
## Specificity : 0.99282   
## Pos Pred Value : 0.91429   
## Neg Pred Value : 0.94749   
## Prevalence : 0.11628   
## Detection Rate : 0.06765   
## Detection Prevalence : 0.07400   
## Balanced Accuracy : 0.78732   
##   
## 'Positive' Class : ParoleViolation   
##

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction NoParoleViolation ParoleViolation  
## NoParoleViolation 175 15  
## ParoleViolation 4 8  
##   
## Accuracy : 0.9059   
## 95% CI : (0.857, 0.9424)  
## No Information Rate : 0.8861   
## P-Value [Acc > NIR] : 0.22246   
##   
## Kappa : 0.4112   
##   
## Mcnemar's Test P-Value : 0.02178   
##   
## Sensitivity : 0.34783   
## Specificity : 0.97765   
## Pos Pred Value : 0.66667   
## Neg Pred Value : 0.92105   
## Prevalence : 0.11386   
## Detection Rate : 0.03960   
## Detection Prevalence : 0.05941   
## Balanced Accuracy : 0.66274   
##   
## 'Positive' Class : ParoleViolation  
##

The train dataset always seems to do better than the test data set as far as test p-value and sensitivity and specificity.

model\_list = caretList(x =as.data.frame(train[,-9]),   
y=as.matrix(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)  
))

## 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.

as.data.frame(predict(model\_list, newdata = head(train)))

## rf rpart nn glm  
## 1 0.9897485 0.9097222 0.9757075 0.9520893  
## 2 0.9526153 0.9097222 0.9183948 0.8526230  
## 3 0.9731575 0.9097222 0.9401129 0.8616431  
## 4 0.9835205 0.9097222 0.9498271 0.9268397  
## 5 0.8449895 0.9097222 0.9424431 0.9233879  
## 6 0.9632013 0.9097222 0.7687076 0.7418396

modelCor(resamples(model\_list))

## rf rpart nn glm  
## rf 1.0000000 0.6497858 0.9397014 0.9616768  
## rpart 0.6497858 1.0000000 0.6200740 0.6838858  
## nn 0.9397014 0.6200740 1.0000000 0.9884822  
## glm 0.9616768 0.6838858 0.9884822 1.0000000

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: 'NoParoleViolation', 'ParoleViolation'   
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 378, 379, 379, 378, 378   
## Resampling results:  
##   
## ROC Sens Spec   
## 0.8338288 0.9712851 0.1818182

summary(stack)

##   
## Call:  
## NULL  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.5424 -0.3621 -0.2997 -0.2756 2.5910   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 3.66937 0.84045 4.366 1.27e-05 \*\*\*  
## rf -4.38506 1.79228 -2.447 0.0144 \*   
## rpart 2.40045 1.15104 2.085 0.0370 \*   
## nn -4.98945 3.31919 -1.503 0.1328   
## glm -0.02296 2.89883 -0.008 0.9937   
## ---  
## 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: 266.11 on 468 degrees of freedom  
## AIC: 276.11  
##   
## Number of Fisher Scoring iterations: 5

The stack ensemble performed better than the individual ensemble models.

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction NoParoleViolation ParoleViolation  
## NoParoleViolation 418 27  
## ParoleViolation 0 28  
##   
## Accuracy : 0.9429   
## 95% CI : (0.918, 0.962)   
## No Information Rate : 0.8837   
## P-Value [Acc > NIR] : 8.647e-06   
##   
## Kappa : 0.647   
##   
## Mcnemar's Test P-Value : 5.624e-07   
##   
## Sensitivity : 0.5091   
## Specificity : 1.0000   
## Pos Pred Value : 1.0000   
## Neg Pred Value : 0.9393   
## Prevalence : 0.1163   
## Detection Rate : 0.0592   
## Detection Prevalence : 0.0592   
## Balanced Accuracy : 0.7545   
##   
## 'Positive' Class : ParoleViolation  
##

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction NoParoleViolation ParoleViolation  
## NoParoleViolation 178 19  
## ParoleViolation 1 4  
##   
## Accuracy : 0.901   
## 95% CI : (0.8512, 0.9385)  
## No Information Rate : 0.8861   
## P-Value [Acc > NIR] : 0.2968298   
##   
## Kappa : 0.2554   
##   
## Mcnemar's Test P-Value : 0.0001439   
##   
## Sensitivity : 0.17391   
## Specificity : 0.99441   
## Pos Pred Value : 0.80000   
## Neg Pred Value : 0.90355   
## Prevalence : 0.11386   
## Detection Rate : 0.01980   
## Detection Prevalence : 0.02475   
## Balanced Accuracy : 0.58416   
##   
## 'Positive' Class : ParoleViolation   
##

The test predictions always seem to not be as good as the training set. The training set has a good test p-value but the testing set has a much larger at 0.00365 with testing 0.00000. The sensitivity is always lower with the test than with the train. They are both low but test is lower by almost 50% but not quite that much.