## Module 5 PSNNE Assignment

### Madison West

Load in libraries

library(tidyverse)

##

## v ggplot2 3.2.1 v purrr 0.3.3  
## v tibble 2.1.3 v dplyr 0.8.3  
## v tidyr 1.0.0 v stringr 1.4.0  
## v readr 1.3.1 v forcats 0.4.0

##   
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

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

library(mice)

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

##   
## Attaching package: 'mice'

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

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

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")) %>%   
 mutate(race = as\_factor(as.character(race))) %>%  
 mutate(race = fct\_recode(race,  
 "White" = "1",  
 "NotWhite" = "2")) %>%  
 mutate(state = as\_factor(as.character(state))) %>%  
 mutate(state = fct\_recode(state,  
 "Kentucky" = "2",  
 "Louisiana" = "3",  
 "Virginia" = "4",  
 "Anyotherstate" = "1")) %>%  
 mutate(crime = as\_factor(as.character(crime))) %>%  
 mutate(crime = fct\_recode(crime,  
 "larceny" = "2",  
 "drugrelated" = "3",  
 "drivingrelated" = "4",  
 "anyothercrime" = "1")) %>%  
 mutate(multiple.offenses = as\_factor(as.character(multiple.offenses))) %>%  
 mutate(multiple.offenses = fct\_recode(multiple.offenses,  
 "SingleOffense" = "0",  
 "MultipleOffense" = "1")) %>%  
 mutate(violator = as\_factor(as.character(violator))) %>%  
 mutate(violator = fct\_recode(violator,  
 "CompletedParole" = "0",  
 "ViolatedParole" = "1"))

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.

set.seed(12345)  
train.rows = createDataPartition(y = parole$violator, p=0.7, list = FALSE) #70% in training  
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.

imp\_age = mice(train, m=1, method='pmm', printFlag=FALSE)  
train = complete(imp\_age)

start\_time = Sys.time() #for timing  
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=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 5.17455 secs

nnetBasic

## Neural Network   
##   
## 473 samples  
## 8 predictor  
## 2 classes: 'CompletedParole', 'ViolatedParole'   
##   
## 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

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.

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction CompletedParole ViolatedParole  
## CompletedParole 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   
##

**The accuracy of this model is 0.9535 which is very high. The sensitivity is 0.636, and the specificity is 0.995. This is one of our best models that we have produced on this data.**

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() #for timing  
fitControl = trainControl(method = "cv",   
 number = 10)  
  
nnetGrid = expand.grid(size = seq(from = 2, to = 12, by = 1), #rule of thumb --> between # of input and # of output layers  
 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",  
 tuneGrid = nnetGrid,  
 trControl = fitControl,  
 verbose = FALSE,  
 trace = FALSE)  
  
end\_time = Sys.time()  
end\_time-start\_time

## Time difference of 1.878392 mins

nnetFit

## Neural Network   
##   
## 473 samples  
## 8 predictor  
## 2 classes: 'CompletedParole', 'ViolatedParole'   
##   
## 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   
## 2 0.1 0.8689581 0.13637924  
## 2 0.2 0.8732616 0.21113829  
## 2 0.3 0.8879336 0.16224380  
## 2 0.4 0.8900613 0.18838547  
## 2 0.5 0.8815950 0.04082244  
## 3 0.1 0.8647491 0.17526818  
## 3 0.2 0.8773358 0.20653353  
## 3 0.3 0.8815911 0.19090764  
## 3 0.4 0.8815044 0.10263245  
## 3 0.5 0.8836764 0.13329768  
## 4 0.1 0.8687770 0.25257194  
## 4 0.2 0.8880204 0.26177876  
## 4 0.3 0.8794635 0.16310342  
## 4 0.4 0.8815487 0.15050166  
## 4 0.5 0.8858040 0.16092237  
## 5 0.1 0.8707697 0.22697344  
## 5 0.2 0.8793286 0.23383795  
## 5 0.3 0.8773377 0.14964396  
## 5 0.4 0.8836764 0.15608899  
## 5 0.5 0.8878874 0.18610910  
## 6 0.1 0.8731248 0.22060486  
## 6 0.2 0.8793266 0.22556787  
## 6 0.3 0.8752987 0.13393754  
## 6 0.4 0.8794635 0.16486060  
## 6 0.5 0.8836764 0.15386677  
## 7 0.1 0.8625771 0.18065506  
## 7 0.2 0.8794191 0.22023180  
## 7 0.3 0.8752544 0.11210256  
## 7 0.4 0.8794191 0.14729995  
## 7 0.5 0.8836764 0.15608899  
## 8 0.1 0.8708622 0.24381297  
## 8 0.2 0.8792842 0.21763088  
## 8 0.3 0.8877968 0.24067539  
## 8 0.4 0.8772915 0.13905753  
## 8 0.5 0.8858040 0.18092237  
## 9 0.1 0.8688695 0.22950299  
## 9 0.2 0.8813232 0.26573252  
## 9 0.3 0.8773801 0.19589030  
## 9 0.4 0.8773821 0.13878603  
## 9 0.5 0.8858040 0.17735335  
## 10 0.1 0.8751619 0.27311932  
## 10 0.2 0.8750732 0.18513695  
## 10 0.3 0.8815911 0.18440148  
## 10 0.4 0.8815930 0.16819853  
## 10 0.5 0.8837207 0.17364964  
## 11 0.1 0.8813213 0.27154203  
## 11 0.2 0.8708622 0.15799701  
## 11 0.3 0.8835858 0.21014955  
## 11 0.4 0.8773358 0.14137402  
## 11 0.5 0.8858040 0.17735335  
## 12 0.1 0.8793286 0.28999711  
## 12 0.2 0.8772915 0.19097162  
## 12 0.3 0.8709528 0.12993939  
## 12 0.4 0.8836764 0.15608899  
## 12 0.5 0.8816374 0.16772372  
##   
## 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.

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.

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction CompletedParole ViolatedParole  
## CompletedParole 414 46  
## ViolatedParole 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 : ViolatedParole   
##

**The accuracy of this model is 0.8943 which is relatively high. The sensitivity is 0.164, and the specificity is 0.99. This model is slightly weaker than the original model.**

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.

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction CompletedParole ViolatedParole  
## CompletedParole 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   
##

**The accuracy of this model is 0.8911, the sensitivity is 0.391, and the specificity is 0.955. This model is not as strong as the model on the training data, but not terrible.**

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.

test\_preds2 = predict(nnetFit, test)  
confusionMatrix(test\_preds2,test$violator,positive="ViolatedParole")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction CompletedParole ViolatedParole  
## CompletedParole 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   
##

**The accuracy of this model is 0.9158, the sensitivity is 0.391, and the specificity is 0.98. This model is higher than the model on the train data.**

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

**It does not appear that there is overfitting in the second model, but there may be on the first. Both were pretty highly accurate on the test data, but the accuracy of the model from task 2 was lower than that of the train model.**

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.

control = trainControl(  
 method = "cv",  
 number = 5, #to save time, we'll use 5 fold cross-validation rather than 10  
 savePredictions = "final",  
 classProbs = TRUE, #instructs caret to calculate probabilities (rather than providing final classifications)  
 summaryFunction = twoClassSummary #enables calculation of AUC  
 )

set.seed(111)  
model\_list = caretList(x=as.data.frame(train[,-9]), y=as.matrix(train$violator),  
 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 0.7851787 -0.3306601 0.1659973  
## rpart 0.7851787 1.0000000 -0.6957475 -0.4400857  
## nn -0.3306601 -0.6957475 1.0000000 0.5363745  
## glm 0.1659973 -0.4400857 0.5363745 1.0000000

ensemble = caretEnsemble(  
 model\_list,   
 metric="ROC",  
 trControl=trainControl(  
 method = "cv", #cross-validation during ensembling  
 number= 5, #number of folds  
 summaryFunction=twoClassSummary,  
 classProbs=TRUE  
 ))

summary(ensemble)

## The following models were ensembled: rf, rpart, nn, glm   
## They were weighted:   
## 3.2747 -4.8994 1.609 -0.1582 -3.0419  
## The resulting ROC is: 0.8185  
## The fit for each individual model on the ROC is:   
## method ROC ROCSD  
## rf 0.8329317 0.04390970  
## rpart 0.6713712 0.04964340  
## nn 0.8458796 0.02699027  
## glm 0.8386507 0.03410036

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction CompletedParole ViolatedParole  
## CompletedParole 418 16  
## ViolatedParole 0 39  
##   
## Accuracy : 0.9662   
## 95% CI : (0.9456, 0.9805)  
## No Information Rate : 0.8837   
## P-Value [Acc > NIR] : 1.024e-10   
##   
## Kappa : 0.8116   
##   
## Mcnemar's Test P-Value : 0.0001768   
##   
## Sensitivity : 1.0000   
## Specificity : 0.7091   
## Pos Pred Value : 0.9631   
## Neg Pred Value : 1.0000   
## Prevalence : 0.8837   
## Detection Rate : 0.8837   
## Detection Prevalence : 0.9175   
## Balanced Accuracy : 0.8545   
##   
## 'Positive' Class : CompletedParole   
##

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction CompletedParole ViolatedParole  
## CompletedParole 175 17  
## ViolatedParole 4 6  
##   
## Accuracy : 0.896   
## 95% CI : (0.8455, 0.9345)  
## No Information Rate : 0.8861   
## P-Value [Acc > NIR] : 0.379652   
##   
## Kappa : 0.3165   
##   
## Mcnemar's Test P-Value : 0.008829   
##   
## Sensitivity : 0.9777   
## Specificity : 0.2609   
## Pos Pred Value : 0.9115   
## Neg Pred Value : 0.6000   
## Prevalence : 0.8861   
## Detection Rate : 0.8663   
## Detection Prevalence : 0.9505   
## Balanced Accuracy : 0.6193   
##   
## 'Positive' Class : CompletedParole   
##

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.

**The rf and rpart models in the ensemble appear to be highly correlated with a correlation coefficient of 0.786, and the rpart and nn models are also correlated with a coefficient of -0.696. The remaining models are fairly weakly correlated.**

**The ensemble has a AUC of 0.827, which is higher than the rpart model, but slightly lower than the remaining models.**

**The accuracy of the train model (0.9662) appears to be the highest between the ensemble model predictions on train and test (accuracy=0.896), but these accuracy of the test set is not terrible. Slight overfitting may have occured.**