Marci B Copeland BAN502 Module 5 Assignment 1

install.packages(“nnet”) install.packages(“rpart”) install.packages(“caretEnsemble”)

library(tidyverse)

## -- Attaching packages --------------------------------------------------------- tidyverse 1.2.1 --

## 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

## -- Conflicts ------------------------------------------------------------ tidyverse\_conflicts() --  
## 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)

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

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()  
## )

str(parole)

## Classes 'spec\_tbl\_df', 'tbl\_df', 'tbl' and 'data.frame': 675 obs. of 9 variables:  
## $ male : num 1 0 1 1 1 1 1 0 0 1 ...  
## $ race : num 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 : num 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 : num 18 12 12 18 12 18 18 12 13 12 ...  
## $ multiple.offenses: num 0 0 0 0 0 0 0 0 0 0 ...  
## $ crime : num 4 3 3 1 1 4 3 1 3 2 ...  
## $ violator : num 0 0 0 0 0 0 0 0 0 0 ...  
## - attr(\*, "spec")=  
## .. 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()  
## .. )

summary(parole)

## male race age state   
## Min. :0.0000 Min. :1.000 Min. :18.40 Min. :1.000   
## 1st Qu.:1.0000 1st Qu.:1.000 1st Qu.:25.35 1st Qu.:2.000   
## Median :1.0000 Median :1.000 Median :33.70 Median :3.000   
## Mean :0.8074 Mean :1.424 Mean :34.51 Mean :2.887   
## 3rd Qu.:1.0000 3rd Qu.:2.000 3rd Qu.:42.55 3rd Qu.:4.000   
## Max. :1.0000 Max. :2.000 Max. :67.00 Max. :4.000   
## time.served max.sentence multiple.offenses crime   
## Min. :0.000 Min. : 1.00 Min. :0.0000 Min. :1.000   
## 1st Qu.:3.250 1st Qu.:12.00 1st Qu.:0.0000 1st Qu.:1.000   
## Median :4.400 Median :12.00 Median :1.0000 Median :2.000   
## Mean :4.198 Mean :13.06 Mean :0.5363 Mean :2.059   
## 3rd Qu.:5.200 3rd Qu.:15.00 3rd Qu.:1.0000 3rd Qu.:3.000   
## Max. :6.000 Max. :18.00 Max. :1.0000 Max. :4.000   
## violator   
## Min. :0.0000   
## 1st Qu.:0.0000   
## Median :0.0000   
## Mean :0.1156   
## 3rd Qu.:0.0000   
## Max. :1.0000

parole = parole %>%  
mutate(male = as.factor(male)) %>%  
mutate(male = fct\_recode(male, "Female" = "0", "Male" = "1")) %>%  
mutate(race = as.factor(race)) %>%  
mutate(race = fct\_recode(race, "White"="1", "Other"="2")) %>%  
mutate(state = as.factor(state)) %>%  
mutate(state = fct\_recode(state, "Other"="1","KY"="2","LA"="3","VA"="4")) %>%  
mutate(crime = as.factor(crime)) %>%  
mutate(crime = fct\_recode(crime, "Other"="1","Larceny"="2","Drug"="3","Driving"="4")) %>%  
mutate(multiple.offenses = as.factor(multiple.offenses)) %>%  
mutate(multiple.offenses = fct\_recode(multiple.offenses, "Yes"="1","No"="0")) %>%  
mutate(violator = as.factor(violator)) %>%  
mutate(violator = fct\_recode(violator, "Yes"="1","No"="0"))  
str(parole)

## Classes 'spec\_tbl\_df', 'tbl\_df', 'tbl' and 'data.frame': 675 obs. of 9 variables:  
## $ male : Factor w/ 2 levels "Female","Male": 2 1 2 2 2 2 2 1 1 2 ...  
## $ race : Factor w/ 2 levels "White","Other": 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 "Other","KY","LA",..: 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 : num 18 12 12 18 12 18 18 12 13 12 ...  
## $ multiple.offenses: Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...  
## $ crime : Factor w/ 4 levels "Other","Larceny",..: 4 3 3 1 1 4 3 1 3 2 ...  
## $ violator : Factor w/ 2 levels "No","Yes": 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,]

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.

install.packages(“randomForest”)

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

## Class: mids  
## Number of multiple imputations: 1   
## Imputation methods:  
## male race age state   
## "" "" "" ""   
## time.served max.sentence multiple.offenses crime   
## "" "" "" ""   
## violator   
## ""   
## PredictorMatrix:  
## male race age state time.served max.sentence  
## male 0 1 1 1 1 1  
## race 1 0 1 1 1 1  
## age 1 1 0 1 1 1  
## state 1 1 1 0 1 1  
## time.served 1 1 1 1 0 1  
## max.sentence 1 1 1 1 1 0  
## multiple.offenses crime violator  
## male 1 1 1  
## race 1 1 1  
## age 1 1 1  
## state 1 1 1  
## time.served 1 1 1  
## max.sentence 1 1 1

train\_complete = complete(imp\_age)   
summary(train\_complete)

## male race age state time.served   
## Female: 98 White:271 Min. :18.40 Other: 95 Min. :0.000   
## Male :375 Other:202 1st Qu.:25.10 KY : 83 1st Qu.:3.300   
## Median :33.50 LA : 58 Median :4.400   
## Mean :34.15 VA :237 Mean :4.185   
## 3rd Qu.:42.40 3rd Qu.:5.200   
## Max. :65.10 Max. :6.000   
## max.sentence multiple.offenses crime violator   
## Min. : 1.00 No :212 Other :231 No :418   
## 1st Qu.:12.00 Yes:261 Larceny: 74 Yes: 55   
## Median :12.00 Drug :103   
## Mean :13.03 Driving: 65   
## 3rd Qu.:15.00   
## Max. :18.00

start\_time = Sys.time()   
train\_Control = trainControl(method = "cv",   
 number = 10)  
  
nnetGrid <- expand.grid(size = 12, decay = 0.1)  
  
set.seed(1234)  
  
nnetBasic = train(x=train\_complete[,-1], y= train\_complete$violator,  
 method = "nnet",  
 tuneGrid = nnetGrid,  
 trControl = train\_Control,  
 verbose = FALSE,  
 trace = FALSE)  
  
end\_time = Sys.time()  
end\_time-start\_time

## Time difference of 2.2649 secs

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  
## 1 1   
##   
## Tuning parameter 'size' was held constant at a value of 12  
##   
## Tuning parameter 'decay' was held constant at a value of 0.1

predtrain = predict(nnetBasic, train\_complete)

confusionMatrix(predtrain, train\_complete$violator, positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 418 0  
## Yes 0 55  
##   
## Accuracy : 1   
## 95% CI : (0.9922, 1)  
## No Information Rate : 0.8837   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 1   
##   
## Mcnemar's Test P-Value : NA   
##   
## Sensitivity : 1.0000   
## Specificity : 1.0000   
## Pos Pred Value : 1.0000   
## Neg Pred Value : 1.0000   
## Prevalence : 0.1163   
## Detection Rate : 0.1163   
## Detection Prevalence : 0.1163   
## Balanced Accuracy : 1.0000   
##   
## 'Positive' Class : Yes   
##

Based on percentage, this is good quality model.

start\_time = Sys.time()   
train\_Control2 = 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=train\_complete[,-1],y=train\_complete$violator,   
 method = "nnet",  
 trControl = train\_Control2,  
 tuneGrid = nnetGrid,  
 verbose = FALSE,  
 trace = FALSE)  
  
end\_time = Sys.time()  
end\_time-start\_time

## Time difference of 51.23942 secs

predtrain2 = predict(nnetBasic, train\_complete)

confusionMatrix(predtrain2, train\_complete$violator, positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 418 0  
## Yes 0 55  
##   
## Accuracy : 1   
## 95% CI : (0.9922, 1)  
## No Information Rate : 0.8837   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 1   
##   
## Mcnemar's Test P-Value : NA   
##   
## Sensitivity : 1.0000   
## Specificity : 1.0000   
## Pos Pred Value : 1.0000   
## Neg Pred Value : 1.0000   
## Prevalence : 0.1163   
## Detection Rate : 0.1163   
## Detection Prevalence : 0.1163   
## Balanced Accuracy : 1.0000   
##   
## 'Positive' Class : Yes   
##

Same as previous, percentage shows higher quality model.

imp\_age = mice(test, m=1, method='pmm', printFlag=FALSE)  
summary(imp\_age)

## Class: mids  
## Number of multiple imputations: 1   
## Imputation methods:  
## male race age state   
## "" "" "" ""   
## time.served max.sentence multiple.offenses crime   
## "" "" "" ""   
## violator   
## ""   
## PredictorMatrix:  
## male race age state time.served max.sentence  
## male 0 1 1 1 1 1  
## race 1 0 1 1 1 1  
## age 1 1 0 1 1 1  
## state 1 1 1 0 1 1  
## time.served 1 1 1 1 0 1  
## max.sentence 1 1 1 1 1 0  
## multiple.offenses crime violator  
## male 1 1 1  
## race 1 1 1  
## age 1 1 1  
## state 1 1 1  
## time.served 1 1 1  
## max.sentence 1 1 1

test\_complete = complete(imp\_age)   
summary(test\_complete)

## male race age state time.served   
## Female: 32 White:118 Min. :19.00 Other:48 Min. :0.100   
## Male :170 Other: 84 1st Qu.:26.32 KY :37 1st Qu.:3.225   
## Median :34.10 LA :24 Median :4.400   
## Mean :35.36 VA :93 Mean :4.228   
## 3rd Qu.:43.48 3rd Qu.:5.200   
## Max. :67.00 Max. :6.000   
## max.sentence multiple.offenses crime violator   
## Min. : 1.00 No :101 Other :84 No :179   
## 1st Qu.:12.00 Yes:101 Larceny:32 Yes: 23   
## Median :12.00 Drug :50   
## Mean :13.12 Driving:36   
## 3rd Qu.:15.00   
## Max. :18.00

start\_time = Sys.time()   
testControl = trainControl(method = "cv",   
 number = 10)  
  
nnetGrid1 <- expand.grid(size = 12, decay = 0.1)  
  
set.seed(1234)  
nnetBasic1 = train(x=test\_complete[,-1], y= test\_complete$violator,  
 method = "nnet",  
 tuneGrid = nnetGrid,  
 trControl = testControl,  
 verbose = FALSE,  
 trace = FALSE)  
  
end\_time = Sys.time()  
end\_time-start\_time

## Time difference of 26.7835 secs

nnetBasic1

## Neural Network   
##   
## 202 samples  
## 8 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 182, 183, 181, 182, 182, 181, ...   
## Resampling results across tuning parameters:  
##   
## size decay Accuracy Kappa   
## 1 0.1 1.0000000 1.0000000  
## 1 0.2 1.0000000 1.0000000  
## 1 0.3 1.0000000 1.0000000  
## 1 0.4 0.9952381 0.9774194  
## 1 0.5 0.9904762 0.9548387  
## 2 0.1 1.0000000 1.0000000  
## 2 0.2 1.0000000 1.0000000  
## 2 0.3 1.0000000 1.0000000  
## 2 0.4 1.0000000 1.0000000  
## 2 0.5 0.9952381 0.9774194  
## 3 0.1 1.0000000 1.0000000  
## 3 0.2 1.0000000 1.0000000  
## 3 0.3 1.0000000 1.0000000  
## 3 0.4 1.0000000 1.0000000  
## 3 0.5 0.9952381 0.9774194  
## 4 0.1 1.0000000 1.0000000  
## 4 0.2 1.0000000 1.0000000  
## 4 0.3 1.0000000 1.0000000  
## 4 0.4 1.0000000 1.0000000  
## 4 0.5 0.9952381 0.9774194  
## 5 0.1 1.0000000 1.0000000  
## 5 0.2 1.0000000 1.0000000  
## 5 0.3 1.0000000 1.0000000  
## 5 0.4 1.0000000 1.0000000  
## 5 0.5 0.9952381 0.9774194  
## 6 0.1 1.0000000 1.0000000  
## 6 0.2 1.0000000 1.0000000  
## 6 0.3 1.0000000 1.0000000  
## 6 0.4 1.0000000 1.0000000  
## 6 0.5 0.9952381 0.9774194  
## 7 0.1 1.0000000 1.0000000  
## 7 0.2 1.0000000 1.0000000  
## 7 0.3 1.0000000 1.0000000  
## 7 0.4 1.0000000 1.0000000  
## 7 0.5 0.9952381 0.9774194  
## 8 0.1 1.0000000 1.0000000  
## 8 0.2 1.0000000 1.0000000  
## 8 0.3 1.0000000 1.0000000  
## 8 0.4 1.0000000 1.0000000  
## 8 0.5 0.9952381 0.9774194  
## 9 0.1 1.0000000 1.0000000  
## 9 0.2 1.0000000 1.0000000  
## 9 0.3 1.0000000 1.0000000  
## 9 0.4 1.0000000 1.0000000  
## 9 0.5 0.9952381 0.9774194  
## 10 0.1 1.0000000 1.0000000  
## 10 0.2 1.0000000 1.0000000  
## 10 0.3 1.0000000 1.0000000  
## 10 0.4 1.0000000 1.0000000  
## 10 0.5 0.9952381 0.9774194  
## 11 0.1 1.0000000 1.0000000  
## 11 0.2 1.0000000 1.0000000  
## 11 0.3 1.0000000 1.0000000  
## 11 0.4 1.0000000 1.0000000  
## 11 0.5 0.9952381 0.9774194  
## 12 0.1 1.0000000 1.0000000  
## 12 0.2 1.0000000 1.0000000  
## 12 0.3 1.0000000 1.0000000  
## 12 0.4 1.0000000 1.0000000  
## 12 0.5 0.9952381 0.9774194  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were size = 1 and decay = 0.3.

predtest = predict(nnetBasic1, test\_complete)

confusionMatrix(predtest, test\_complete$violator, positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 179 0  
## Yes 0 23  
##   
## Accuracy : 1   
## 95% CI : (0.9819, 1)  
## No Information Rate : 0.8861   
## P-Value [Acc > NIR] : 2.485e-11   
##   
## Kappa : 1   
##   
## Mcnemar's Test P-Value : NA   
##   
## Sensitivity : 1.0000   
## Specificity : 1.0000   
## Pos Pred Value : 1.0000   
## Neg Pred Value : 1.0000   
## Prevalence : 0.1139   
## Detection Rate : 0.1139   
## Detection Prevalence : 0.1139   
## Balanced Accuracy : 1.0000   
##   
## 'Positive' Class : Yes   
##

Percentage is a little less, but still a good quality model.

start\_time = Sys.time()   
testControl2 = 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)  
nnetFit2 = train(x=test\_complete[,-1],y=test\_complete$violator,   
 method = "nnet",  
 trControl = testControl2,  
 tuneGrid = nnetGrid,  
 verbose = FALSE,  
 trace = FALSE)  
  
end\_time = Sys.time()  
end\_time-start\_time

## Time difference of 26.92241 secs

predtest2 = predict(nnetBasic1, test\_complete)

confusionMatrix(predtest2, test\_complete$violator, positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 179 0  
## Yes 0 23  
##   
## Accuracy : 1   
## 95% CI : (0.9819, 1)  
## No Information Rate : 0.8861   
## P-Value [Acc > NIR] : 2.485e-11   
##   
## Kappa : 1   
##   
## Mcnemar's Test P-Value : NA   
##   
## Sensitivity : 1.0000   
## Specificity : 1.0000   
## Pos Pred Value : 1.0000   
## Neg Pred Value : 1.0000   
## Prevalence : 0.1139   
## Detection Rate : 0.1139   
## Detection Prevalence : 0.1139   
## Balanced Accuracy : 1.0000   
##   
## 'Positive' Class : Yes   
##

Same percentage as previous and same quality model.

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

Specify list of models to include in the ensemble.

set.seed(111)  
model\_list = caretList(  
 violator ~ .,   
 data = train\_complete,   
 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, new\_train\_data=head(train\_complete)))

## Warning in predict.caretList(model\_list, new\_train\_data =  
## head(train\_complete)): Predicting without new data is not well supported.  
## Attempting to predict on the training data.

## rf rpart nn glm  
## 1 0.9901944 0.9253012 0.9336210 0.9520893  
## 2 0.9831941 0.9253012 0.9443406 0.8526230  
## 3 0.9885992 0.9253012 0.9503245 0.8616431  
## 4 0.9916014 0.9253012 0.9310976 0.9268397  
## 5 0.9473743 0.9253012 0.9549287 0.9233879  
## 6 0.9441209 0.9253012 0.7650537 0.7418396  
## 7 0.9746306 0.9253012 0.9606765 0.9409661  
## 8 0.9581090 0.9253012 0.9321716 0.9308133  
## 9 0.9707271 0.9253012 0.9895047 0.9563217  
## 10 0.9943842 0.9253012 0.9906777 0.9563752  
## 11 0.9900808 0.9253012 0.9849258 0.8443472  
## 12 0.9709200 0.9253012 0.9481064 0.9227400  
## 13 0.9914291 0.9253012 0.9720080 0.9498325  
## 14 0.9520470 0.9253012 0.9815080 0.9460723  
## 15 0.9605320 0.9253012 0.9434133 0.9270246  
## 16 0.9892641 0.9253012 0.9523793 0.8593733  
## 17 0.4434692 0.9253012 0.8760130 0.9036776  
## 18 0.9836141 0.9253012 0.9724874 0.8505092  
## 19 0.9357977 0.9253012 0.9261524 0.8682995  
## 20 0.9761287 0.9253012 0.9008991 0.7455148  
## 21 0.9409873 0.9253012 0.9775827 0.9246827  
## 22 0.9297372 0.9253012 0.7982197 0.8892686  
## 23 0.4294859 0.9253012 0.7723472 0.8797820  
## 24 0.9578393 0.9253012 0.9534346 0.8747907  
## 25 0.8837024 0.9253012 0.8041569 0.8674777  
## 26 0.8803495 0.9253012 0.7643111 0.8778485  
## 27 0.9616022 0.9253012 0.9562261 0.8961551  
## 28 0.8820368 0.9253012 0.9622986 0.8599089  
## 29 0.3152038 0.9253012 0.5176597 0.7084483  
## 30 0.2407689 0.9253012 0.5857912 0.7314023  
## 31 0.7615732 0.9253012 0.9361514 0.6079839  
## 32 0.2281488 0.9253012 0.4942415 0.5836878  
## 33 0.2259809 0.9253012 0.4772696 0.6906688  
## 34 0.7872672 0.9253012 0.8227711 0.8048343  
## 35 0.2012992 0.9253012 0.4856408 0.6957043  
## 36 0.8494165 0.9253012 0.8136480 0.7201592  
## 37 0.8213705 0.9253012 0.6325843 0.6341653  
## 38 0.7369211 0.9253012 0.5980130 0.6976313  
## 39 0.1551162 0.9253012 0.5031185 0.6860738  
## 40 0.8616950 0.9253012 0.8963498 0.6556206  
## 41 0.1257924 0.9253012 0.4908752 0.6793111  
## 42 0.3258842 0.9253012 0.4237986 0.6113551  
## 43 0.7868540 0.9253012 0.4615461 0.5902054  
## 44 0.2716272 0.9253012 0.5065179 0.7037818  
## 45 0.8980576 0.9253012 0.8062160 0.5178830  
## 46 0.9401073 0.9253012 0.9118092 0.9129655  
## 47 0.8960147 0.9253012 0.9592324 0.8774399  
## 48 0.9026480 0.9253012 0.8619522 0.9086209  
## 49 0.8907813 0.9253012 0.9670381 0.8695254  
## 50 0.9115654 0.9253012 0.8132910 0.7655234  
## 51 0.8473079 0.9253012 0.4116707 0.4398478  
## 52 0.9553659 0.9253012 0.8272677 0.9104883  
## 53 0.9553659 0.9253012 0.8272677 0.9104883  
## 54 0.9692899 0.9253012 0.9242181 0.9141919  
## 55 0.9692899 0.9253012 0.9242181 0.9141919  
## 56 0.3350801 0.9253012 0.7298793 0.8212655  
## 57 0.9652033 0.9253012 0.8341093 0.7921323  
## 58 0.9812490 0.9253012 0.8532883 0.9279188  
## 59 0.4502956 0.9253012 0.9521882 0.9344113  
## 60 0.9784112 0.9253012 0.9735404 0.9559306  
## 61 0.8590574 0.9253012 0.6082749 0.7517103  
## 62 0.3863004 0.9253012 0.7419155 0.6543581  
## 63 0.9163191 0.9253012 0.3949501 0.6041413  
## 64 0.9495139 0.9253012 0.9316274 0.9018676  
## 65 0.9205517 0.9253012 0.8935692 0.9337832  
## 66 0.9494479 0.9253012 0.8276301 0.7176718  
## 67 0.9673466 0.9253012 0.8441400 0.9065716  
## 68 0.9588803 0.9253012 0.9179417 0.9336355  
## 69 0.8531280 0.9253012 0.6615373 0.7726623  
## 70 0.8534791 0.9253012 0.9242978 0.9244189  
## 71 0.9765549 0.9253012 0.8266274 0.9242969  
## 72 0.9772792 0.9253012 0.9256915 0.9320796  
## 73 0.9705549 0.9253012 0.8270132 0.9243707  
## 74 0.9472420 0.9253012 0.8952141 0.9404579  
## 75 0.9216541 0.9253012 0.9258769 0.8133492  
## 76 0.8684774 0.9253012 0.8601119 0.9264679  
## 77 0.9666633 0.9253012 0.9056076 0.9428888  
## 78 0.9138976 0.9253012 0.9188457 0.9331229  
## 79 0.8396437 0.9253012 0.8483513 0.9159841  
## 80 0.9745777 0.9253012 0.9713039 0.9164831  
## 81 0.7943042 0.9253012 0.8580151 0.9271791  
## 82 0.9526324 0.9253012 0.9304829 0.8965766  
## 83 0.9715076 0.9253012 0.9403599 0.9385809  
## 84 0.2676765 0.9253012 0.4281821 0.4455107  
## 85 0.9111890 0.9253012 0.3954273 0.6215669  
## 86 0.8360089 0.9253012 0.8758679 0.9284257  
## 87 0.9329842 0.9253012 0.9195751 0.9132177  
## 88 0.9683049 0.9253012 0.8824746 0.7920822  
## 89 0.8629060 0.9253012 0.7977752 0.8348000  
## 90 0.3777219 0.9253012 0.6962605 0.8074859  
## 91 0.9733183 0.9253012 0.8898089 0.8329819  
## 92 0.9571299 0.9253012 0.9004780 0.9416997  
## 93 0.9458277 0.9253012 0.9335757 0.9428358  
## 94 0.9394440 0.9253012 0.7058234 0.8760060  
## 95 0.2675479 0.9253012 0.3758370 0.3537066  
## 96 0.9480328 0.9253012 0.7872592 0.5578978  
## 97 0.3461579 0.9253012 0.6536761 0.8674397  
## 98 0.8987304 0.9253012 0.9377276 0.5962405  
## 99 0.9775114 0.9253012 0.9696259 0.9287671  
## 100 0.4374246 0.9253012 0.8197992 0.9140668  
## 101 0.9785669 0.9253012 0.9830308 0.9241015  
## 102 0.9493887 0.9253012 0.9158932 0.7602490  
## 103 0.9142797 0.9253012 0.9366864 0.9219579  
## 104 0.4575485 0.9253012 0.9254298 0.9199155  
## 105 0.9906697 0.9253012 0.9546051 0.9313550  
## 106 0.9906697 0.9253012 0.9546051 0.9313550  
## 107 0.3775497 0.9253012 0.6183755 0.7494603  
## 108 0.3977960 0.9253012 0.8905985 0.9193915  
## 109 0.8959580 0.9253012 0.5140982 0.6561730  
## 110 0.9893553 0.9253012 0.9775947 0.9188305  
## 111 0.8118089 0.9253012 0.9470698 0.9180684  
## 112 0.9615551 0.9253012 0.8842062 0.9380224  
## 113 0.9448680 0.9253012 0.9300730 0.9377409  
## 114 0.9448680 0.9253012 0.9300730 0.9377409  
## 115 0.8983922 0.9253012 0.4208161 0.6860247  
## 116 0.9891072 0.9253012 0.9560008 0.9268152  
## 117 0.5206347 0.9253012 0.9153964 0.9394662  
## 118 0.9467618 0.9253012 0.9653694 0.9414670  
## 119 0.9484055 0.9253012 0.8035502 0.9059918  
## 120 0.9079279 0.9253012 0.8156909 0.9005244  
## 121 0.9511727 0.9253012 0.8596152 0.9298369  
## 122 0.9394105 0.9253012 0.9291975 0.8381297  
## 123 0.9622051 0.9253012 0.9689762 0.8729854  
## 124 0.9805582 0.9253012 0.9670397 0.9250524  
## 125 0.9503441 0.9253012 0.9626064 0.9177400  
## 126 0.9801021 0.9253012 0.9809523 0.9360959  
## 127 0.9801021 0.9253012 0.9809523 0.9360959  
## 128 0.9186456 0.9253012 0.8810617 0.9370820  
## 129 0.8891801 0.8000000 0.6971675 0.8599369  
## 130 0.6610944 0.3529412 0.3622230 0.2756135  
## 131 0.8896678 0.8000000 0.9820574 0.8631941  
## 132 0.6817241 0.5714286 0.4249311 0.4524097  
## 133 0.2289780 0.5714286 0.3615621 0.3222171  
## 134 0.3250276 0.5714286 0.4416326 0.4525339  
## 135 0.2611702 0.8000000 0.4440830 0.7379369  
## 136 0.2442881 0.5714286 0.3573696 0.2856222  
## 137 0.7353859 0.3529412 0.7281455 0.5159938  
## 138 0.7513886 0.3529412 0.6515380 0.4416182  
## 139 0.1550834 0.3529412 0.3499621 0.2432156  
## 140 0.8736704 0.5714286 0.4574765 0.4655271  
## 141 0.8326833 0.8000000 0.4333758 0.7142945  
## 142 0.7932731 0.5714286 0.4093802 0.4737687  
## 143 0.3524536 0.5714286 0.6971681 0.5123529  
## 144 0.2587236 0.8000000 0.5539661 0.6629096  
## 145 0.8716704 0.5714286 0.4530544 0.4664467  
## 146 0.8410519 0.8000000 0.4622672 0.6797728  
## 147 0.2679949 0.3529412 0.6009683 0.2147262  
## 148 0.1743003 0.5714286 0.5982055 0.4542655  
## 149 0.9306394 0.8000000 0.6147618 0.7295666  
## 150 0.3389324 0.5714286 0.3879665 0.4634548  
## 151 0.6987788 0.3529412 0.3608908 0.3034851  
## 152 0.8450396 0.8000000 0.9295045 0.8118726  
## 153 0.3516430 0.3529412 0.5985274 0.7651161  
## 154 0.8649540 0.8000000 0.5600072 0.5291341  
## 155 0.8656043 0.8000000 0.9790783 0.8675112  
## 156 0.2302984 0.3529412 0.5527857 0.4255518  
## 157 0.8567715 0.5714286 0.4275699 0.4924330  
## 158 0.8921862 0.8000000 0.9273548 0.5265800  
## 159 0.7993096 0.5714286 0.5931607 0.7173171  
## 160 0.6316483 0.3529412 0.4241010 0.3220798  
## 161 0.3183150 0.8000000 0.5663918 0.6662787  
## 162 0.3283144 0.8000000 0.4524584 0.6589912  
## 163 0.2375059 0.3529412 0.3642559 0.3562135  
## 164 0.7983183 0.8000000 0.9206714 0.5293016  
## 165 0.2679455 0.5714286 0.5351690 0.6878761  
## 166 0.2146879 0.3529412 0.4715756 0.5030309  
## 167 0.7711640 0.5714286 0.6692110 0.8034899  
## 168 0.1527975 0.3529412 0.3827611 0.4614459  
## 169 0.7228468 0.5714286 0.4023876 0.4580789  
## 170 0.1584752 0.3529412 0.3518445 0.2250361  
## 171 0.6780721 0.5714286 0.3612625 0.2759946  
## 172 0.8716121 0.5714286 0.9714416 0.8406585  
## 173 0.2560142 0.3529412 0.3613914 0.3209439  
## 174 0.3064021 0.3529412 0.4526978 0.3112780  
## 175 0.8759649 0.8000000 0.8904137 0.8560764  
## 176 0.2717497 0.5714286 0.3897214 0.4894183  
## 177 0.9241325 0.8000000 0.6495980 0.9003066  
## 178 0.2995347 0.5714286 0.6019885 0.7787085  
## 179 0.9048281 0.5714286 0.9833549 0.8143655  
## 180 0.9214293 0.8000000 0.8910310 0.8860610  
## 181 0.7960934 0.5714286 0.7184513 0.8138206  
## 182 0.9287706 0.8000000 0.9943161 0.9706156  
## 183 0.9207237 0.8000000 0.9655221 0.9419882  
## 184 0.7716932 0.3529412 0.4940895 0.7110242  
## 185 0.2638059 0.3529412 0.6184856 0.7425304  
## 186 0.9123887 0.8000000 0.9726344 0.9480090  
## 187 0.9863538 0.9253012 0.8784990 0.8886912  
## 188 0.9909093 0.9253012 0.9299258 0.8872430  
## 189 0.9408955 0.9253012 0.9604105 0.8199437  
## 190 0.9437071 0.9253012 0.9680481 0.9433742  
## 191 0.9492153 0.9253012 0.9604495 0.9286168  
## 192 0.9438449 0.9253012 0.9263833 0.9255966  
## 193 0.9681521 0.9253012 0.9122117 0.9094036  
## 194 0.9549421 0.9253012 0.9818100 0.9225074  
## 195 0.8486850 0.9253012 0.5484242 0.5799212  
## 196 0.9808611 0.9253012 0.8948349 0.7455539  
## 197 0.8486427 0.9253012 0.5673416 0.7373817  
## 198 0.3333985 0.9253012 0.6462047 0.7153288  
## 199 0.9862493 0.9253012 0.9814255 0.9433135  
## 200 0.9958087 0.9253012 0.9889532 0.9640644  
## 201 0.9884982 0.9253012 0.9799883 0.9701548  
## 202 0.9831633 0.9253012 0.9712735 0.9353745  
## 203 0.9520430 0.9253012 0.9355377 0.7010846  
## 204 0.9845156 0.9253012 0.9559654 0.9378195  
## 205 0.9401854 0.9253012 0.9862289 0.9499726  
## 206 0.8558256 0.9253012 0.8030618 0.7441482  
## 207 0.9490565 0.9253012 0.9820919 0.9557493  
## 208 0.3904050 0.9253012 0.9193223 0.9176028  
## 209 0.3361526 0.9253012 0.6383094 0.8336886  
## 210 0.9906165 0.9253012 0.9797263 0.9072934  
## 211 0.9721029 0.9253012 0.9703218 0.9437088  
## 212 0.9813361 0.9253012 0.9267603 0.9461718  
## 213 0.9758063 0.9253012 0.9829984 0.9196746  
## 214 0.9923889 0.9253012 0.9641909 0.9537306  
## 215 0.9978028 0.9253012 0.9903451 0.9518604  
## 216 0.9787694 0.9253012 0.9559052 0.9315937  
## 217 0.9919028 0.9253012 0.9916782 0.9602504  
## 218 0.9854028 0.9253012 0.9544300 0.9355609  
## 219 0.9744343 0.9253012 0.9367822 0.9206855  
## 220 0.9181606 0.9253012 0.9623334 0.6259205  
## 221 0.9830694 0.9253012 0.9106207 0.9461066  
## 222 0.9706939 0.9253012 0.9479335 0.9343105  
## 223 0.9296302 0.9253012 0.8730666 0.8687479  
## 224 0.9865361 0.9253012 0.9708874 0.9392260  
## 225 0.9656816 0.9253012 0.9833846 0.9446315  
## 226 0.9997361 0.9253012 0.9702954 0.9422810  
## 227 0.9840176 0.9253012 0.9732989 0.9269556  
## 228 0.9790224 0.9253012 0.9586752 0.9587094  
## 229 0.9731902 0.9253012 0.9543514 0.8985363  
## 230 0.9692147 0.9253012 0.9567524 0.9407287  
## 231 0.9997361 0.9253012 0.9666862 0.9413430  
## 232 0.9567480 0.9253012 0.9629924 0.9407899  
## 233 0.9817361 0.9253012 0.9490547 0.9431527  
## 234 1.0000000 0.9253012 0.9875937 0.9618545  
## 235 0.9970558 0.9253012 0.9792419 0.9893944  
## 236 0.9987960 0.9253012 0.9874482 0.9906678  
## 237 0.9894000 0.9253012 0.9942471 0.9867200  
## 238 0.9995977 0.9253012 0.9934819 0.9978378  
## 239 0.9754798 0.9253012 0.9736753 0.9874176  
## 240 0.9980000 0.9253012 0.9975030 0.9981320  
## 241 0.9957142 0.9253012 0.9820297 0.9874423  
## 242 0.9928650 0.9253012 0.9726365 0.9416660  
## 243 0.9988819 0.9253012 0.9837584 0.9640858  
## 244 0.9990039 0.9253012 0.9947079 0.9882661  
## 245 0.9793967 0.9253012 0.9814235 0.9626479  
## 246 0.9932333 0.9253012 0.9936773 0.9981942  
## 247 0.9807251 0.9253012 0.9740626 0.9698749  
## 248 0.9901099 0.9253012 0.9670911 0.9508704  
## 249 0.9449208 0.9253012 0.9949696 0.9967719  
## 250 0.9537914 0.9253012 0.9719534 0.9874696  
## 251 0.9998553 0.9253012 0.9783081 0.9726096  
## 252 0.9851397 0.9253012 0.9965986 0.9980887  
## 253 0.9885796 0.9253012 0.8965957 0.9419001  
## 254 0.9120954 0.9253012 0.8149482 0.9441872  
## 255 0.9267212 0.9253012 0.9671161 0.9803321  
## 256 0.9977881 0.9253012 0.9967792 0.9903787  
## 257 0.9687333 0.9253012 0.9561919 0.9666563  
## 258 0.9967496 0.9253012 0.9741227 0.9460819  
## 259 0.9931587 0.9253012 0.9921103 0.9877316  
## 260 0.9962625 0.9253012 0.9499672 0.9687919  
## 261 0.9994285 0.9253012 0.9946663 0.9883613  
## 262 0.9920076 0.9253012 0.9650385 0.9822536  
## 263 0.9973883 0.9253012 0.9898269 0.9942713  
## 264 0.9941688 0.9253012 0.9682680 0.9711440  
## 265 0.9863375 0.9253012 0.9935084 0.9967615  
## 266 0.9967604 0.9253012 0.9871925 0.9820405  
## 267 0.9999375 0.9253012 0.9878715 0.9942667  
## 268 0.9997848 0.9253012 0.9985663 0.9932450  
## 269 0.9991624 0.9253012 0.9247220 0.9565806  
## 270 0.9980000 0.9253012 0.9930383 0.9942053  
## 271 0.9454496 0.9253012 0.9923280 0.9865620  
## 272 0.9962444 0.9253012 0.9954622 0.9878857  
## 273 0.9995264 0.9253012 0.9947372 0.9990191  
## 274 0.9874724 0.9253012 0.9187065 0.9417059  
## 275 0.9998553 0.9253012 0.9632044 0.9717571  
## 276 0.9997972 0.9253012 0.9211125 0.9549811  
## 277 0.9576919 0.9253012 0.9299088 0.9591350  
## 278 0.9970000 0.9253012 0.9991583 0.9929576  
## 279 0.9856705 0.9253012 0.9952894 0.9888690  
## 280 0.9944720 0.9253012 0.9353743 0.9655452  
## 281 0.9966026 0.9253012 0.9063742 0.9488811  
## 282 0.9920302 0.9253012 0.9361921 0.9595420  
## 283 0.9962453 0.9253012 0.9926373 0.9929788  
## 284 0.9998078 0.9253012 0.9409988 0.9618532  
## 285 0.9971995 0.9253012 0.9874519 0.9850005  
## 286 0.9959991 0.9253012 0.9981557 0.9879009  
## 287 0.9961688 0.9253012 0.9607425 0.9718312  
## 288 0.9795299 0.9253012 0.9965899 0.9904722  
## 289 0.9842641 0.9253012 0.9777304 0.9822092  
## 290 0.9940458 0.9253012 0.9571723 0.9694040  
## 291 0.9927619 0.9253012 0.9995642 0.9978560  
## 292 0.9819681 0.9253012 0.9863240 0.9825739  
## 293 0.9940980 0.9253012 0.9923525 0.9921057  
## 294 0.9980717 0.9253012 0.9922579 0.9891577  
## 295 0.9989083 0.9253012 0.9780792 0.9630222  
## 296 0.9944314 0.9253012 0.9928524 0.9930223  
## 297 0.9938786 0.9253012 0.9504352 0.9461800  
## 298 0.9824606 0.9253012 0.9708193 0.9876708  
## 299 0.9523792 0.9253012 0.9375506 0.9823724  
## 300 0.9968356 0.9253012 0.9945113 0.9880676  
## 301 0.9925822 0.9253012 0.9852579 0.9933440  
## 302 0.9916331 0.9253012 0.9995955 0.9970901  
## 303 0.9856740 0.9253012 0.9353913 0.9803733  
## 304 0.9938786 0.9253012 0.9392521 0.9458835  
## 305 0.9940980 0.9253012 0.9932849 0.9911540  
## 306 0.9970197 0.9253012 0.9912129 0.9932912  
## 307 0.9828195 0.9253012 0.9973225 0.9901900  
## 308 0.9779269 0.9253012 0.9704856 0.9857522  
## 309 0.9935833 0.9253012 0.9984631 0.9935494  
## 310 0.9948979 0.9253012 0.9834771 0.9656866  
## 311 0.9868242 0.9253012 0.9532994 0.9372236  
## 312 0.9994188 0.9253012 0.9913315 0.9735964  
## 313 0.9990824 0.9253012 0.9935471 0.9945919  
## 314 0.9866593 0.9253012 0.9825720 0.9808810  
## 315 0.9878347 0.9253012 0.9662740 0.9795912  
## 316 0.9964865 0.9253012 0.9784268 0.9881801  
## 317 0.9983405 0.9253012 0.9962930 0.9899828  
## 318 0.9735397 0.9253012 0.9938941 0.9980915  
## 319 0.9919833 0.9253012 0.9856435 0.9896876  
## 320 0.9959976 0.9253012 0.9848259 0.9838320  
## 321 0.9786568 0.9253012 0.9478590 0.9646412  
## 322 0.9974882 0.9253012 0.9957183 0.9946515  
## 323 0.9986026 0.9253012 0.9792458 0.9517712  
## 324 0.9999375 0.9253012 0.9841863 0.9928389  
## 325 0.9544523 0.9253012 0.9120822 0.9801245  
## 326 0.9861214 0.9253012 0.9831256 0.9897087  
## 327 1.0000000 0.9253012 0.9950500 0.9979703  
## 328 0.9940000 0.9253012 0.9956965 0.9924910  
## 329 0.9848910 0.9253012 0.9304099 0.9634626  
## 330 0.9794862 0.9253012 0.9964777 0.9910390  
## 331 0.9977768 0.9253012 0.9877229 0.9863223  
## 332 0.9970884 0.9253012 0.9734918 0.9843473  
## 333 0.9935000 0.9253012 0.9951342 0.9982456  
## 334 0.9997768 0.9253012 0.9876441 0.9855852  
## 335 0.9711152 0.9253012 0.9931840 0.9924795  
## 336 0.9987715 0.9253012 0.9487243 0.9566132  
## 337 0.9954568 0.9253012 0.9774636 0.9651825  
## 338 0.9830675 0.9253012 0.9906441 0.9895564  
## 339 0.9938589 0.9253012 0.9705379 0.9834621  
## 340 0.9877388 0.9253012 0.9930492 0.9874191  
## 341 0.9974882 0.9253012 0.9933238 0.9949729  
## 342 0.9322867 0.9253012 0.9624748 0.9793859  
## 343 0.9957647 0.9253012 0.9928812 0.9926305  
## 344 0.9997142 0.9253012 0.9936624 0.9876667  
## 345 0.9512839 0.9253012 0.9773878 0.9461604  
## 346 0.9975159 0.9253012 0.9850516 0.9693938  
## 347 0.9993969 0.9253012 0.9930740 0.9943834  
## 348 0.9911995 0.9253012 0.9798501 0.9846579  
## 349 0.9996667 0.9253012 0.9932485 0.9948039  
## 350 0.9992809 0.9253012 0.9772674 0.9681305  
## 351 0.9956458 0.9253012 0.9492968 0.9683707  
## 352 0.9866260 0.9253012 0.9336003 0.9616673  
## 353 0.9973333 0.9253012 0.9934963 0.9978438  
## 354 0.9919024 0.9253012 0.9252371 0.9595308  
## 355 0.9978960 0.9253012 0.9781130 0.9887870  
## 356 0.9977142 0.9253012 0.9955312 0.9893281  
## 357 0.9973135 0.9253012 0.9587796 0.9707913  
## 358 0.9769633 0.9253012 0.9415247 0.9657198  
## 359 0.9760493 0.9253012 0.9257798 0.9555200  
## 360 0.9995021 0.9253012 0.9581084 0.9712693  
## 361 0.9980607 0.9253012 0.9765639 0.9393003  
## 362 0.9800753 0.9253012 0.9709272 0.9671662  
## 363 0.9991142 0.9253012 0.9953303 0.9888289  
## 364 0.9939024 0.9253012 0.9931499 0.9937038  
## 365 0.9714612 0.9253012 0.9871979 0.9814096  
## 366 0.9996701 0.9253012 0.9895881 0.9946490  
## 367 0.9923931 0.9253012 0.9975412 0.9989157  
## 368 0.9242555 0.9253012 0.9940227 0.9880618  
## 369 0.9927611 0.9253012 0.9778166 0.9826013  
## 370 0.9725571 0.9253012 0.9866298 0.9640923  
## 371 0.9964791 0.9253012 0.9817126 0.9712551  
## 372 0.9995021 0.9253012 0.9914041 0.9785019  
## 373 0.9888638 0.9253012 0.9854532 0.9883399  
## 374 0.9609879 0.9253012 0.9727886 0.9811741  
## 375 0.9931100 0.9253012 0.9681101 0.9679903  
## 376 0.9949675 0.9253012 0.9822400 0.9903922  
## 377 0.9954432 0.9253012 0.9791065 0.9515357  
## 378 0.9948322 0.9253012 0.9604024 0.9787367  
## 379 0.9974809 0.9253012 0.9669088 0.9665679  
## 380 0.9943185 0.9253012 0.9885892 0.9943033  
## 381 0.9976273 0.9253012 0.9687293 0.9872934  
## 382 0.9871528 0.9253012 0.9599873 0.9836299  
## 383 0.9962833 0.9253012 0.9813288 0.9826071  
## 384 0.9995534 0.9253012 0.9573150 0.9617217  
## 385 0.9823550 0.9253012 0.9919561 0.9956217  
## 386 0.9996667 0.9253012 0.9934605 0.9948717  
## 387 0.9998960 0.9253012 0.9951940 0.9892155  
## 388 0.9940000 0.9253012 0.9936839 0.9974632  
## 389 0.9969066 0.9253012 0.9807622 0.9858350  
## 390 0.9957142 0.9253012 0.9865585 0.9868902  
## 391 0.9914724 0.9253012 0.9967008 0.9914682  
## 392 0.5074512 0.9253012 0.9737047 0.9869962  
## 393 0.9964865 0.9253012 0.9789599 0.9881431  
## 394 0.9996701 0.9253012 0.9955741 0.9950148  
## 395 0.9428718 0.9253012 0.9440153 0.9461664  
## 396 0.9498525 0.9253012 0.9912430 0.9899640  
## 397 0.9943401 0.9253012 0.9944108 0.9882827  
## 398 0.9996667 0.9253012 0.9933447 0.9949918  
## 399 0.9918553 0.9253012 0.9547643 0.9691727  
## 400 0.9992809 0.9253012 0.9683636 0.9678196  
## 401 0.9994809 0.9253012 0.9726246 0.9640653  
## 402 0.9974996 0.9253012 0.9741837 0.9470109  
## 403 0.9891667 0.9253012 0.9944519 0.9924991  
## 404 0.9798527 0.9253012 0.9665553 0.9637101  
## 405 0.9980076 0.9253012 0.9654901 0.9824002  
## 406 0.9979672 0.9253012 0.9918981 0.9857805  
## 407 0.9955299 0.9253012 0.9797478 0.9898492  
## 408 0.9949724 0.9253012 0.9985827 0.9907985  
## 409 0.9989437 0.9253012 0.9959695 0.9895339  
## 410 0.9865055 0.9253012 0.9172018 0.9623823  
## 411 0.9820858 0.9253012 0.9199203 0.9626816  
## 412 0.9990507 0.9253012 0.9875439 0.9664497  
## 413 0.9370242 0.9253012 0.9807877 0.9887857  
## 414 0.9760128 0.9253012 0.9699015 0.9833992  
## 415 0.9355453 0.9253012 0.8405177 0.9476902  
## 416 0.9997577 0.9253012 0.9687314 0.9676200  
## 417 0.4147493 0.9253012 0.9617407 0.9809847  
## 418 0.4101861 0.9253012 0.9242744 0.9592167  
## 419 0.9974791 0.9253012 0.9784471 0.9678269  
## 420 0.9996667 0.9253012 0.9935579 0.9959525  
## 421 0.9837100 0.9253012 0.9587287 0.9643145  
## 422 0.9715341 0.9253012 0.9858675 0.9912454  
## 423 0.9743170 0.9253012 0.9651709 0.9635045  
## 424 0.9951886 0.9253012 0.9719069 0.9713414  
## 425 0.9977142 0.9253012 0.9950160 0.9885207  
## 426 0.9952448 0.9253012 0.9895731 0.9839431  
## 427 0.9999437 0.9253012 0.9974088 0.9911913  
## 428 0.9709375 0.9253012 0.9943103 0.9940885  
## 429 0.3167768 0.9253012 0.7706272 0.9351015  
## 430 0.9983575 0.9253012 0.9950224 0.9879610  
## 431 0.9997768 0.9253012 0.9907599 0.9866102  
## 432 0.9877847 0.9253012 0.9926668 0.9922276  
## 433 0.9979024 0.9253012 0.9795506 0.9572988  
## 434 0.9977577 0.9253012 0.9783213 0.9679656  
## 435 0.9963362 0.9253012 0.9782717 0.9674348  
## 436 0.9854893 0.9253012 0.9756409 0.9366148  
## 437 0.9981740 0.9253012 0.9791916 0.9506244  
## 438 0.9940000 0.9253012 0.9959377 0.9984311  
## 439 0.9789437 0.9253012 0.9960577 0.9895775  
## 440 0.9995638 0.9253012 0.9896709 0.9714041  
## 441 0.9979024 0.9253012 0.9956567 0.9936548  
## 442 0.9916667 0.9253012 0.9932445 0.9946178  
## 443 0.9987715 0.9253012 0.9838761 0.9598881  
## 444 0.9856495 0.9253012 0.9580815 0.9837625  
## 445 0.9914769 0.9253012 0.9907300 0.9938869  
## 446 0.9377020 0.9253012 0.9776027 0.9883922  
## 447 0.9973333 0.9253012 0.9844861 0.9644940  
## 448 0.9995901 0.9253012 0.9784698 0.9666228  
## 449 0.9998960 0.9253012 0.9958691 0.9894424  
## 450 0.9977024 0.9253012 0.9316015 0.9612265  
## 451 0.9930993 0.9253012 0.9870198 0.9943879  
## 452 0.8548976 0.9253012 0.8922811 0.9484905  
## 453 0.9991084 0.9253012 0.9992978 0.9964544  
## 454 0.4588553 0.9253012 0.8296358 0.9457533  
## 455 0.9979024 0.9253012 0.9818823 0.9575355  
## 456 0.9990286 0.9253012 0.9941085 0.9983057  
## 457 0.9999024 0.9253012 0.9927452 0.9934005  
## 458 0.9499487 0.9253012 0.9780751 0.9797248  
## 459 0.9919385 0.9253012 0.9989032 0.9897192  
## 460 0.9967667 0.9253012 0.9987951 0.9984288  
## 461 0.9996705 0.9253012 0.9952803 0.9887874  
## 462 0.9988213 0.9253012 0.9811672 0.9638474  
## 463 0.9996273 0.9253012 0.9953657 0.9888966  
## 464 0.9936856 0.9253012 0.9852302 0.9890671  
## 465 0.9969066 0.9253012 0.9802323 0.9844981  
## 466 0.9730771 0.9253012 0.9934131 0.9974615  
## 467 0.9985991 0.9253012 0.9653100 0.9638920  
## 468 0.9837140 0.9253012 0.9812629 0.9823316  
## 469 0.9949558 0.9253012 0.9226654 0.9574777  
## 470 0.9713026 0.9253012 0.9678651 0.9862000  
## 471 0.9967493 0.9253012 0.9865308 0.9440667  
## 472 0.8521481 0.9253012 0.7211780 0.8617918  
## 473 0.9732183 0.9253012 0.9906143 0.9665166

modelCor(resamples(model\_list))

## rf rpart nn glm  
## rf 1.00000000 0.6762461 0.06602417 0.2791765  
## rpart 0.67624612 1.0000000 -0.41844801 -0.4400857  
## nn 0.06602417 -0.4184480 1.00000000 0.8664434  
## glm 0.27917647 -0.4400857 0.86644339 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:   
## 2.8848 -4.3636 2.2361 -2.2699 -1.7239  
## The resulting ROC is: 0.8394  
## The fit for each individual model on the ROC is:   
## method ROC ROCSD  
## rf 0.8337232 0.05122000  
## rpart 0.6713712 0.04964340  
## nn 0.8444636 0.02425292  
## glm 0.8386507 0.03410036

From the summary, we see that the resulting ROC for the ensemble is 0.8639. While this is better than the ROC for the glm and rpart models, it is not better than the ranger model (0.867) on its own. We may be better off just using the ranger Random Forest instead of the ensemble model.

We can (anyway) evaluate the performance of the ensemble on the training and testing sets.

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 418 14  
## Yes 0 41  
##   
## Accuracy : 0.9704   
## 95% CI : (0.9508, 0.9837)  
## No Information Rate : 0.8837   
## P-Value [Acc > NIR] : 6.464e-12   
##   
## Kappa : 0.8381   
##   
## Mcnemar's Test P-Value : 0.000512   
##   
## Sensitivity : 1.0000   
## Specificity : 0.7455   
## Pos Pred Value : 0.9676   
## Neg Pred Value : 1.0000   
## Prevalence : 0.8837   
## Detection Rate : 0.8837   
## Detection Prevalence : 0.9133   
## Balanced Accuracy : 0.8727   
##   
## 'Positive' Class : No   
##

Lesser quality model in comparsion to individual model.

modelCor(resamples(model\_list))

## rf rpart nn glm  
## rf 1.00000000 0.6762461 0.06602417 0.2791765  
## rpart 0.67624612 1.0000000 -0.41844801 -0.4400857  
## nn 0.06602417 -0.4184480 1.00000000 0.8664434  
## glm 0.27917647 -0.4400857 0.86644339 1.0000000

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

set.seed(111)  
model\_test\_list1 = caretList(  
 violator ~ .,   
 data = test\_complete,   
 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\_test\_list1, new\_test\_data=head(test\_complete)))

## Warning in predict.caretList(model\_test\_list1, new\_test\_data =  
## head(test\_complete)): Predicting without new data is not well supported.  
## Attempting to predict on the training data.

## rf rpart nn glm  
## 1 0.99600000 0.9438202 0.99801195 0.9971601  
## 2 0.98550000 0.9438202 0.94222817 0.9369596  
## 3 0.88581234 0.9438202 0.87901244 0.9250579  
## 4 0.97880000 0.9438202 0.98622623 0.9974633  
## 5 0.90774243 0.9438202 0.96679449 0.9767259  
## 6 0.93925000 0.9438202 0.92090914 0.9521997  
## 7 0.99600000 0.9438202 0.85418224 0.9268771  
## 8 0.96325183 0.9438202 0.96167465 0.9529297  
## 9 0.98262393 0.9438202 0.89975457 0.9594721  
## 10 0.97964615 0.9438202 0.97012422 0.9603344  
## 11 0.97950000 0.9438202 0.98622595 0.9274506  
## 12 0.97670000 0.9438202 0.94855586 0.9370920  
## 13 0.98860000 0.9438202 0.94342530 0.9255394  
## 14 0.31230300 0.9438202 0.53550269 0.7832402  
## 15 0.86161462 0.9438202 0.76628483 0.5201143  
## 16 0.81537652 0.9438202 0.65919929 0.5146921  
## 17 0.26799239 0.9438202 0.31666159 0.5298309  
## 18 0.91814203 0.9438202 0.83361709 0.8332645  
## 19 0.90215697 0.9438202 0.91888110 0.8020544  
## 20 0.87252299 0.9438202 0.86518443 0.7565908  
## 21 0.80841584 0.9438202 0.83940410 0.7379235  
## 22 0.28866650 0.9438202 0.39639243 0.7803329  
## 23 0.96961111 0.9438202 0.94053333 0.8937057  
## 24 0.96961111 0.9438202 0.94053333 0.8937057  
## 25 0.98181828 0.9438202 0.98883052 0.9974720  
## 26 0.48367545 0.9438202 0.77793383 0.9611726  
## 27 0.99235157 0.9438202 0.98307919 0.9968383  
## 28 0.95903968 0.9438202 0.93154209 0.9247170  
## 29 0.44366593 0.9438202 0.66366385 0.9651869  
## 30 0.96429729 0.9438202 0.99443900 0.9569814  
## 31 0.95537778 0.9438202 0.98477134 0.9966006  
## 32 0.99200000 0.9438202 0.96087822 0.9179878  
## 33 0.83907988 0.9438202 0.72019679 0.9635027  
## 34 0.95448650 0.9438202 0.99543681 0.9620923  
## 35 0.97718980 0.9438202 0.97929223 0.9566735  
## 36 0.99550000 0.9438202 0.98342441 0.9350020  
## 37 0.92472691 0.9438202 0.97776826 0.9619143  
## 38 0.96209729 0.9438202 0.99472452 0.9571544  
## 39 0.95296602 0.9438202 0.98505254 0.9627470  
## 40 0.95411244 0.9438202 0.98760904 0.9542804  
## 41 0.97622855 0.9438202 0.99451496 0.9971836  
## 42 0.95366667 0.9438202 0.98701692 0.9754637  
## 43 0.98333333 0.9438202 0.99046872 0.9943894  
## 44 0.97347411 0.9438202 0.99153337 0.9590563  
## 45 0.98240113 0.9438202 0.94236970 0.9555857  
## 46 0.97573810 0.9438202 0.98234338 0.9301379  
## 47 0.95046689 0.9438202 0.98223624 0.9619963  
## 48 0.99977778 0.9438202 0.98340216 0.9204220  
## 49 0.95318139 0.9438202 0.98958757 0.9658778  
## 50 0.82858872 0.9438202 0.65852892 0.7688767  
## 51 0.91603144 0.9438202 0.95314990 0.9655207  
## 52 0.91603144 0.9438202 0.95314990 0.9655207  
## 53 0.99185114 0.9438202 0.99880129 0.9969118  
## 54 0.90163191 0.9438202 0.92295437 0.9579266  
## 55 0.99020664 0.9438202 0.99017267 0.9397935  
## 56 0.90534444 0.9438202 0.96887816 0.7636105  
## 57 0.96215556 0.9438202 0.93619385 0.9387085  
## 58 0.99777778 0.9438202 0.99637407 0.9233799  
## 59 0.95409546 0.9438202 0.98356566 0.9619038  
## 60 0.27058333 0.2857143 0.13053208 0.7263662  
## 61 0.06919048 0.2857143 0.17754425 0.2139901  
## 62 0.24926667 0.2857143 0.29784177 0.2763234  
## 63 0.65210000 0.2857143 0.59623352 0.2589518  
## 64 0.15490714 0.2857143 0.30728345 0.1709122  
## 65 0.23107619 0.2857143 0.34226401 0.2497088  
## 66 0.12656667 0.2857143 0.06387987 0.2522582  
## 67 0.07520714 0.2857143 0.05216843 0.2084244  
## 68 0.21032900 0.7000000 0.21924544 0.4851554  
## 69 0.80552381 0.2857143 0.81384532 0.7674292  
## 70 0.62413333 0.2857143 0.86200142 0.1974876  
## 71 0.81800000 0.2857143 0.91171221 0.8716554  
## 72 0.18027381 0.2857143 0.05602597 0.2147863  
## 73 0.24593333 0.2857143 0.19075291 0.1894920  
## 74 0.87713333 0.7000000 0.96008105 0.5147820  
## 75 0.22592424 0.7000000 0.19246093 0.5140796  
## 76 0.86389091 0.7000000 0.99237076 0.9216755  
## 77 0.15415476 0.2857143 0.30715406 0.2521947  
## 78 0.83275758 0.7000000 0.88180313 0.6886031  
## 79 0.82482900 0.7000000 0.56779678 0.6038212  
## 80 0.84825758 0.7000000 0.92155242 0.4840958  
## 81 0.82013333 0.7000000 0.82514183 0.8557180  
## 82 0.81982900 0.7000000 0.88488041 0.5615497  
## 83 0.20399567 0.7000000 0.25210614 0.5205394  
## 84 0.90865342 0.9438202 0.96336262 0.9572593  
## 85 0.92577778 0.9438202 0.97277354 0.9377365  
## 86 0.90242364 0.9438202 0.90003934 0.8001831  
## 87 0.98515000 0.9438202 0.99386518 0.9972987  
## 88 0.37110274 0.9438202 0.50019453 0.8464771  
## 89 0.35935476 0.9438202 0.50920756 0.9169698  
## 90 0.31762186 0.9438202 0.51960054 0.9201425  
## 91 0.90141412 0.9438202 0.95749304 0.7812923  
## 92 0.90276891 0.9438202 0.80388861 0.8519718  
## 93 0.89825476 0.9438202 0.93858775 0.9097768  
## 94 0.98139712 0.9438202 0.96197494 0.9416599  
## 95 0.92338013 0.9438202 0.96955461 0.7523702  
## 96 0.99000000 0.9438202 0.99787892 0.9957458  
## 97 0.97899311 0.9438202 0.99093167 0.8678047  
## 98 0.91836733 0.9438202 0.86156004 0.9644272  
## 99 0.98690909 0.9438202 0.97394344 0.7879955  
## 100 0.96733333 0.9438202 0.89155951 0.9329714  
## 101 0.98673706 0.9438202 0.97647426 0.9666209  
## 102 0.90297576 0.9438202 0.75495033 0.8158425  
## 103 0.97266667 0.9438202 0.97883398 0.9860674  
## 104 0.98068687 0.9438202 0.88077654 0.8189917  
## 105 0.96466667 0.9438202 0.98759100 0.9864102  
## 106 0.87132496 0.9438202 0.84327149 0.8385327  
## 107 0.97568687 0.9438202 0.72039325 0.8101258  
## 108 0.97033333 0.9438202 0.94516722 0.9851036  
## 109 1.00000000 0.9438202 0.99943958 0.9998241  
## 110 1.00000000 0.9438202 0.99799674 0.9985862  
## 111 0.99600000 0.9438202 0.99961646 0.9938368  
## 112 0.99600000 0.9438202 0.99963037 0.9927365  
## 113 0.99666667 0.9438202 0.95901747 0.9828346  
## 114 1.00000000 0.9438202 0.99789486 0.9795475  
## 115 0.99800000 0.9438202 0.99768124 0.9981093  
## 116 0.93490902 0.9438202 0.98798851 0.9452663  
## 117 0.99800000 0.9438202 0.98900037 0.9730112  
## 118 1.00000000 0.9438202 0.99027620 0.9812990  
## 119 0.92419089 0.9438202 0.98387928 0.9524343  
## 120 1.00000000 0.9438202 0.99360599 0.9837161  
## 121 1.00000000 0.9438202 0.99491761 0.9774092  
## 122 0.99600000 0.9438202 0.99386577 0.9895853  
## 123 0.98766297 0.9438202 0.91592077 0.9417997  
## 124 0.99800000 0.9438202 0.99970313 0.9995888  
## 125 1.00000000 0.9438202 0.99817443 0.9940066  
## 126 1.00000000 0.9438202 0.94414704 0.9733099  
## 127 0.99800000 0.9438202 0.94867752 0.9952012  
## 128 0.37875344 0.9438202 0.83226869 0.9353177  
## 129 1.00000000 0.9438202 0.98922124 0.9710021  
## 130 0.99400000 0.9438202 0.94420751 0.9757383  
## 131 0.99660994 0.9438202 0.96991587 0.9566888  
## 132 0.99760902 0.9438202 0.99359582 0.9960899  
## 133 0.93687249 0.9438202 0.95333159 0.9318588  
## 134 0.99800000 0.9438202 0.99920809 0.9998094  
## 135 1.00000000 0.9438202 0.95568547 0.9803212  
## 136 0.99371429 0.9438202 0.97129609 0.9793368  
## 137 1.00000000 0.9438202 0.91597428 0.9825340  
## 138 1.00000000 0.9438202 0.96444381 0.9713425  
## 139 0.98286297 0.9438202 0.90736859 0.9428806  
## 140 0.98800000 0.9438202 0.99673835 0.9935804  
## 141 0.99119236 0.9438202 0.99589619 0.9965983  
## 142 0.98744236 0.9438202 0.97098884 0.9472468  
## 143 1.00000000 0.9438202 0.98659662 0.9809342  
## 144 1.00000000 0.9438202 0.93842268 0.9808674  
## 145 0.99616667 0.9438202 0.99846301 0.9983155  
## 146 0.97900000 0.9438202 0.99189832 0.9830701  
## 147 1.00000000 0.9438202 0.99409350 0.9859459  
## 148 0.96175756 0.9438202 0.99305826 0.9517352  
## 149 0.92935714 0.9438202 0.99445448 0.9840022  
## 150 1.00000000 0.9438202 0.98746726 0.9808544  
## 151 0.99600000 0.9438202 0.95591228 0.9953284  
## 152 0.99800000 0.9438202 0.99408082 0.9944994  
## 153 0.95819089 0.9438202 0.98999743 0.9503183  
## 154 0.99600000 0.9438202 0.99653953 0.9829693  
## 155 1.00000000 0.9438202 0.98987370 0.9788897  
## 156 1.00000000 0.9438202 0.97449347 0.9802348  
## 157 0.99371429 0.9438202 0.99093720 0.9813389  
## 158 0.99000000 0.9438202 0.99892920 0.9996014  
## 159 0.99800000 0.9438202 0.99429632 0.9943104  
## 160 0.99600000 0.9438202 0.99944373 0.9974743  
## 161 1.00000000 0.9438202 0.97162780 0.9766047  
## 162 1.00000000 0.9438202 0.99539047 0.9837030  
## 163 0.99800000 0.9438202 0.99840968 0.9899777  
## 164 0.98495779 0.9438202 0.98396645 0.9851681  
## 165 1.00000000 0.9438202 0.92761727 0.9813041  
## 166 1.00000000 0.9438202 0.99501474 0.9945266  
## 167 0.99466667 0.9438202 0.98935372 0.9914822  
## 168 0.97944539 0.9438202 0.99242481 0.9277677  
## 169 1.00000000 0.9438202 0.99224220 0.9867320  
## 170 0.99866667 0.9438202 0.99773576 0.9901927  
## 171 0.97186297 0.9438202 0.94431974 0.9402374  
## 172 0.99600000 0.9438202 0.99111341 0.9942728  
## 173 1.00000000 0.9438202 0.99266567 0.9792776  
## 174 1.00000000 0.9438202 0.99513043 0.9976821  
## 175 0.99800000 0.9438202 0.96766795 0.9853156  
## 176 1.00000000 0.9438202 0.99293858 0.9832078  
## 177 0.99200000 0.9438202 0.98863822 0.9954826  
## 178 1.00000000 0.9438202 0.99798990 0.9985559  
## 179 0.99800000 0.9438202 0.98857849 0.9815387  
## 180 0.96894327 0.9438202 0.78940591 0.9558087  
## 181 0.99866667 0.9438202 0.99556219 0.9837115  
## 182 0.99200000 0.9438202 0.98084479 0.9823536  
## 183 1.00000000 0.9438202 0.99854887 0.9967709  
## 184 1.00000000 0.9438202 0.99308034 0.9953810  
## 185 0.89085253 0.9438202 0.89890507 0.9219327  
## 186 1.00000000 0.9438202 0.99482590 0.9748423  
## 187 0.98900000 0.9438202 0.99562684 0.9821586  
## 188 0.99400000 0.9438202 0.96130358 0.9851455  
## 189 0.99394327 0.9438202 0.90636878 0.9561636  
## 190 0.99600000 0.9438202 0.99861473 0.9983254  
## 191 0.99600000 0.9438202 0.98070771 0.9866148  
## 192 1.00000000 0.9438202 0.99362374 0.9799821  
## 193 0.99800000 0.9438202 0.99554183 0.9936181  
## 194 0.98414236 0.9438202 0.99554540 0.9469511  
## 195 1.00000000 0.9438202 0.97161559 0.9803834  
## 196 1.00000000 0.9438202 0.94177548 0.9818861  
## 197 1.00000000 0.9438202 0.98586618 0.9960100  
## 198 0.38727331 0.9438202 0.77334257 0.9451651  
## 199 0.99600000 0.9438202 0.99807841 0.9744791  
## 200 1.00000000 0.9438202 0.99203479 0.9770925  
## 201 0.85053375 0.9438202 0.93040735 0.9290565  
## 202 0.99600000 0.9438202 0.99417949 0.9969171

modelCor(resamples(model\_test\_list1))

## rf rpart nn glm  
## rf 1.0000000 0.9312784 0.8584738 0.8848930  
## rpart 0.9312784 1.0000000 0.6863573 0.6717896  
## nn 0.8584738 0.6863573 1.0000000 0.8073009  
## glm 0.8848930 0.6717896 0.8073009 1.0000000

ensemble1 = caretEnsemble(  
 model\_test\_list1,   
 metric="ROC",  
 trControl=trainControl(  
 method = "cv",   
 number= 5,   
 summaryFunction=twoClassSummary,  
 classProbs=TRUE  
 ))

summary(ensemble1)

## The following models were ensembled: rf, rpart, nn, glm   
## They were weighted:   
## 3.1018 -4.9771 -1.4082 -1.0946 1.1829  
## The resulting ROC is: 0.8077  
## The fit for each individual model on the ROC is:   
## method ROC ROCSD  
## rf 0.8658889 0.11132850  
## rpart 0.7579762 0.12607815  
## nn 0.8659841 0.09953562  
## glm 0.7783254 0.11819068

pred\_ensemble\_test1 = predict(ensemble1, test\_complete, type = "raw")  
confusionMatrix(pred\_ensemble\_test1,test\_complete$violator)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 179 2  
## Yes 0 21  
##   
## Accuracy : 0.9901   
## 95% CI : (0.9647, 0.9988)  
## No Information Rate : 0.8861   
## P-Value [Acc > NIR] : 8.999e-09   
##   
## Kappa : 0.949   
##   
## Mcnemar's Test P-Value : 0.4795   
##   
## Sensitivity : 1.0000   
## Specificity : 0.9130   
## Pos Pred Value : 0.9890   
## Neg Pred Value : 1.0000   
## Prevalence : 0.8861   
## Detection Rate : 0.8861   
## Detection Prevalence : 0.8960   
## Balanced Accuracy : 0.9565   
##   
## 'Positive' Class : No   
##

Lesser percentage than individual model, however high percent of accuracy reflecting a high quality model but less than individual model.

stack\_train = 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  
 )  
)  
  
print(stack)

## function (x, ...)   
## UseMethod("stack")  
## <bytecode: 0x00000000142f7240>  
## <environment: namespace:utils>

Now use the stacked model to make predictions on the training and testing set.

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 418 14  
## Yes 0 41  
##   
## Accuracy : 0.9704   
## 95% CI : (0.9508, 0.9837)  
## No Information Rate : 0.8837   
## P-Value [Acc > NIR] : 6.464e-12   
##   
## Kappa : 0.8381   
##   
## Mcnemar's Test P-Value : 0.000512   
##   
## Sensitivity : 1.0000   
## Specificity : 0.7455   
## Pos Pred Value : 0.9676   
## Neg Pred Value : 1.0000   
## Prevalence : 0.8837   
## Detection Rate : 0.8837   
## Detection Prevalence : 0.9133   
## Balanced Accuracy : 0.8727   
##   
## 'Positive' Class : No   
##

stack\_test = 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  
 )  
)  
  
print(stack)

## function (x, ...)   
## UseMethod("stack")  
## <bytecode: 0x00000000142f7240>  
## <environment: namespace:utils>

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 177 16  
## Yes 2 7  
##   
## Accuracy : 0.9109   
## 95% CI : (0.8628, 0.9463)  
## No Information Rate : 0.8861   
## P-Value [Acc > NIR] : 0.159196   
##   
## Kappa : 0.399   
##   
## Mcnemar's Test P-Value : 0.002183   
##   
## Sensitivity : 0.9888   
## Specificity : 0.3043   
## Pos Pred Value : 0.9171   
## Neg Pred Value : 0.7778   
## Prevalence : 0.8861   
## Detection Rate : 0.8762   
## Detection Prevalence : 0.9554   
## Balanced Accuracy : 0.6466   
##   
## 'Positive' Class : No   
##