Module 5 Assignment

library(tidyverse)

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

## -- Attaching packages --------------------------------------------------------------------------- tidyverse 1.3.0 --

## 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)  
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

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(male)) %>%   
 mutate(male = fct\_recode(male, "Female" = "0", "Male" = "1" ))  
parole = parole %>% mutate(race = as.factor(race)) %>%   
 mutate(race = fct\_recode(race, "Other" = "2", "White" = "1" ))  
parole = parole %>% mutate(state = as.factor(state)) %>%   
 mutate(state = fct\_recode(state, "Other" = "1", "Kentucky" = "2", "Louisianna" = "3", "Virginia" = "4" ))  
parole = parole %>% mutate(crime = as.factor(crime)) %>%   
 mutate(crime = fct\_recode(crime, "Other" = "1", "Larceny" = "2", "Drugs" = "3", "Driving" = "4" ))  
parole = parole %>% mutate(multiple.offenses = as.factor(multiple.offenses)) %>%   
 mutate(multiple.offenses = fct\_recode(multiple.offenses, "No" = "0", "Yes" = "1" ))  
parole = parole %>% mutate(violator = as.factor(violator)) %>%   
 mutate(violator = fct\_recode(violator, "No" = "0", "Yes" = "1" ))  
  
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","Kentucky",..: 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 ...

**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)  
ptrain = parole[train.rows,]   
ptest = 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.

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(ptrain[,-9]), y=as.matrix(ptrain$violator),  
 method = "nnet",  
 tuneGrid = nnetGrid,  
 trControl = fitControl,  
 verbose = FALSE,   
 trace = FALSE)  
  
end\_time = Sys.time()  
end\_time-start\_time

## Time difference of 1.899489 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   
## 0.8711301 0.2265754  
##   
## Tuning parameter 'size' was held constant at a value of 12  
##   
## Tuning parameter 'decay' was held constant at a value of 0.1

NOTES: When I use “as.matrix” for x, I get the errors: model fit failed for Fold10: size=12, decay=0.1 Error in nnet.default(x, y, w, entropy = TRUE, …) : too many (3817) weights

**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, ptrain)  
confusionMatrix(predNetBasic, ptrain$violator, positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 415 23  
## Yes 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 : Yes   
##

ANSWER: We have an accuracy of 0.945 which is pretty good. It’s higher than the “No Information Rate” of 0.8837 so this model is better than the null prediction.

**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.

start\_time = Sys.time() #for timing  
fitControl = 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(x=as.data.frame(ptrain[,-9]), y=as.matrix(ptrain$violator),  
 method = "nnet",  
 tuneGrid = nnetGrid2,  
 trControl = fitControl,  
 verbose = FALSE,  
 trace = FALSE)  
  
end\_time = Sys.time()  
end\_time-start\_time

## Time difference of 47.00083 secs

nnetBasic2

## 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 0.19252451  
## 1 0.2 0.8815930 0.12873404  
## 1 0.3 0.8837207 0.11358290  
## 1 0.4 0.8773821 0.01283588  
## 1 0.5 0.8838113 0.00000000  
## 2 0.1 0.8689157 0.21542030  
## 2 0.2 0.8773821 0.22863708  
## 2 0.3 0.8901499 0.22326632  
## 2 0.4 0.8922333 0.19510248  
## 2 0.5 0.8838113 0.07337632  
## 3 0.1 0.8685534 0.20718187  
## 3 0.2 0.8690044 0.12502538  
## 3 0.3 0.8816817 0.21817579  
## 3 0.4 0.8858484 0.19747446  
## 3 0.5 0.8879760 0.18361467  
## 4 0.1 0.8730362 0.24251556  
## 4 0.2 0.8730362 0.18221357  
## 4 0.3 0.8815468 0.20437056  
## 4 0.4 0.8773821 0.13838045  
## 4 0.5 0.8900150 0.20308951  
## 5 0.1 0.8815025 0.33808427  
## 5 0.2 0.8645217 0.15397001  
## 5 0.3 0.8709971 0.12548863  
## 5 0.4 0.8837207 0.17364964  
## 5 0.5 0.8900150 0.20308951  
## 6 0.1 0.8666493 0.18565720  
## 6 0.2 0.8753430 0.17503287  
## 6 0.3 0.8795097 0.16213793  
## 6 0.4 0.8837207 0.17364964  
## 6 0.5 0.8858040 0.17735335  
## 7 0.1 0.8730786 0.28295210  
## 7 0.2 0.8792842 0.21019384  
## 7 0.3 0.8815468 0.17078653  
## 7 0.4 0.8816374 0.16772372  
## 7 0.5 0.8858040 0.17735335  
## 8 0.1 0.8686440 0.27396135  
## 8 0.2 0.8688232 0.14571186  
## 8 0.3 0.8773801 0.19527790  
## 8 0.4 0.8816374 0.18073605  
## 8 0.5 0.8858040 0.17735335  
## 9 0.1 0.8709066 0.22760948  
## 9 0.2 0.8773821 0.19285732  
## 9 0.3 0.8794635 0.16966577  
## 9 0.4 0.8773377 0.13777782  
## 9 0.5 0.8837207 0.17364964  
## 10 0.1 0.8900150 0.33170758  
## 10 0.2 0.8541975 0.06847940  
## 10 0.3 0.8794654 0.19784036  
## 10 0.4 0.8773821 0.13730301  
## 10 0.5 0.8837207 0.17364964  
## 11 0.1 0.8792842 0.28217396  
## 11 0.2 0.8711282 0.19125440  
## 11 0.3 0.8731711 0.13173951  
## 11 0.4 0.8795097 0.16227261  
## 11 0.5 0.8837207 0.17364964  
## 12 0.1 0.8879741 0.33281680  
## 12 0.2 0.8752968 0.20155517  
## 12 0.3 0.8752987 0.17359794  
## 12 0.4 0.8837207 0.17364964  
## 12 0.5 0.8837207 0.17364964  
##   
## 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.

predNetBasic2 = predict(nnetBasic2, ptrain)  
confusionMatrix(predNetBasic2, ptrain$violator, positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 415 48  
## Yes 3 7  
##   
## Accuracy : 0.8922   
## 95% CI : (0.8607, 0.9187)  
## No Information Rate : 0.8837   
## P-Value [Acc > NIR] : 0.3127   
##   
## Kappa : 0.1863   
##   
## Mcnemar's Test P-Value : 7.218e-10   
##   
## Sensitivity : 0.12727   
## Specificity : 0.99282   
## Pos Pred Value : 0.70000   
## Neg Pred Value : 0.89633   
## Prevalence : 0.11628   
## Detection Rate : 0.01480   
## Detection Prevalence : 0.02114   
## Balanced Accuracy : 0.56005   
##   
## 'Positive' Class : Yes   
##

ANSWER: We have an accuracy of 0.8922. It’s less than 1% higher than “No Information Rate” of 0.8837, but it’s still an improvement. We’ll see if it’s a keeper.

**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.

predNetBasicTest = predict(nnetBasic, ptest)  
confusionMatrix(predNetBasicTest, ptest$violator, positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 172 15  
## Yes 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 : Yes   
##

ANSWER: I guess this would be a disappointment. We have an accuracy of 0.8911. It’s 0.5% higher than “No Information Rate” of 0.8861. But is 5% lower than the accuracy of the training set of 0.9493. It’s in line with the model from Task 4.

**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.

predNetBasic2Test = predict(nnetBasic2, ptest)  
confusionMatrix(predNetBasic2Test, ptest$violator, positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 179 21  
## Yes 0 2  
##   
## Accuracy : 0.896   
## 95% CI : (0.8455, 0.9345)  
## No Information Rate : 0.8861   
## P-Value [Acc > NIR] : 0.3797   
##   
## Kappa : 0.1444   
##   
## Mcnemar's Test P-Value : 1.275e-05   
##   
## Sensitivity : 0.086957   
## Specificity : 1.000000   
## Pos Pred Value : 1.000000   
## Neg Pred Value : 0.895000   
## Prevalence : 0.113861   
## Detection Rate : 0.009901   
## Detection Prevalence : 0.009901   
## Balanced Accuracy : 0.543478   
##   
## 'Positive' Class : Yes   
##

ANSWER: We have an accuracy of 0.896, which is just 1% higher than the “No Information Rate” of 0.8861. Our accuracy against the training set was 0.8922, so we actually got a slightly better result. Maybe a 1% better model is the best we can hope for.

**Task 8** Comment on whether there appears to be overﬁtting in one or both of your models from Tasks 2 and 4. ANSWER: The model for Task 2 seems to be overfitting. With the introduction of new (test) data the accuracy goes down by 5%. Clearly the model is not as good as it initially seems to be when new data is introduced.

For Task 4, the accuracy for the testing data is in line with the accuracy for the training data. This means we have a good model that is consistent when it comes to predicting new data.

So, high accuracy doesn’t mean squat if the model can’t be as accurate on new data.

**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.

set.seed(111)  
control = trainControl(  
 method = "cv",  
 number = 5, #to save time, we'll use 5 fold cross-validation rather than 10  
 index = createFolds(ptrain$violator, 5),  
 savePredictions = "final",  
 classProbs = TRUE, #instructs caret to calculate probabilities (rather than providing final classifications)  
 summaryFunction = twoClassSummary #enables calculation of AUC  
 )  
  
model\_list = caretList(x=as.data.frame(ptrain[,-9]), y=as.matrix(ptrain$violator),  
 metric = "ROC", #specify that maximizing AUC is our objective  
 trControl = control, #using the previously defined trControl object  
 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: glm.fit: fitted probabilities numerically 0 or 1 occurred  
  
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

modelCor(resamples(model\_list))

## rf rpart nn glm  
## rf 1.0000000 -0.8733446 -0.2737874 -0.8434680  
## rpart -0.8733446 1.0000000 0.3660415 0.5114403  
## nn -0.2737874 0.3660415 1.0000000 0.3278191  
## glm -0.8434680 0.5114403 0.3278191 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:   
## 2.1838 -4.2915 0.6192 -1.2225 -0.2254  
## The resulting ROC is: 0.7801  
## The fit for each individual model on the ROC is:   
## method ROC ROCSD  
## rf 0.7721399 0.01686434  
## rpart 0.5535278 0.09574115  
## nn 0.7717975 0.01751593  
## glm 0.7404861 0.05905255

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 418 5  
## Yes 0 50  
##   
## Accuracy : 0.9894   
## 95% CI : (0.9755, 0.9966)  
## No Information Rate : 0.8837   
## P-Value [Acc > NIR] : < 2e-16   
##   
## Kappa : 0.9465   
##   
## Mcnemar's Test P-Value : 0.07364   
##   
## Sensitivity : 1.0000   
## Specificity : 0.9091   
## Pos Pred Value : 0.9882   
## Neg Pred Value : 1.0000   
## Prevalence : 0.8837   
## Detection Rate : 0.8837   
## Detection Prevalence : 0.8943   
## Balanced Accuracy : 0.9545   
##   
## 'Positive' Class : No   
##

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 176 18  
## Yes 3 5  
##   
## Accuracy : 0.896   
## 95% CI : (0.8455, 0.9345)  
## No Information Rate : 0.8861   
## P-Value [Acc > NIR] : 0.37965   
##   
## Kappa : 0.2803   
##   
## Mcnemar's Test P-Value : 0.00225   
##   
## Sensitivity : 0.9832   
## Specificity : 0.2174   
## Pos Pred Value : 0.9072   
## Neg Pred Value : 0.6250   
## Prevalence : 0.8861   
## Detection Rate : 0.8713   
## Detection Prevalence : 0.9604   
## Balanced Accuracy : 0.6003   
##   
## 'Positive' Class : No   
##

QUESTIONS AND ANSWERS: How correlated are the models in the ensemble? ANSWER: We have some high negative correlation (> 0.75) between rf and rpart (-0.8733446) and also between rf and glm (-0.8434680).

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. ANSWER: The AUC/ROC for the ensemble model is 0.779. This is less than 0.001 better than the rf model (AUC = 0.7721399) and the nn model (AUC = 0.7721399), but it is 0.05 better than GLM and 0.22 better than the rpart model. The accuracy for the ensemble model using the training data is 0.9894, which is crazy good. The accuracy for the ensemble model using the test data is 0.896. This is a difference of 0.09, which leads to the conclusion that we are overfitting our model.

**Task 10** Build a stacked ensemble 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.

set.seed(111)  
stack = caretStack(  
 model\_list, #use the list of models already specified  
 method ="glm", #stack models linearly  
 metric ="ROC", #maximize AUC  
 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)

## A glm ensemble of 4 base models: rf, rpart, nn, glm  
##   
## Ensemble results:  
## Generalized Linear Model   
##   
## 1892 samples  
## 4 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 1513, 1514, 1513, 1514, 1514   
## Resampling results:  
##   
## ROC Sens Spec   
## 0.7784581 0.9814532 0.1136364

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 418 5  
## Yes 0 50  
##   
## Accuracy : 0.9894   
## 95% CI : (0.9755, 0.9966)  
## No Information Rate : 0.8837   
## P-Value [Acc > NIR] : < 2e-16   
##   
## Kappa : 0.9465   
##   
## Mcnemar's Test P-Value : 0.07364   
##   
## Sensitivity : 1.0000   
## Specificity : 0.9091   
## Pos Pred Value : 0.9882   
## Neg Pred Value : 1.0000   
## Prevalence : 0.8837   
## Detection Rate : 0.8837   
## Detection Prevalence : 0.8943   
## Balanced Accuracy : 0.9545   
##   
## 'Positive' Class : No   
##

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 176 18  
## Yes 3 5  
##   
## Accuracy : 0.896   
## 95% CI : (0.8455, 0.9345)  
## No Information Rate : 0.8861   
## P-Value [Acc > NIR] : 0.37965   
##   
## Kappa : 0.2803   
##   
## Mcnemar's Test P-Value : 0.00225   
##   
## Sensitivity : 0.9832   
## Specificity : 0.2174   
## Pos Pred Value : 0.9072   
## Neg Pred Value : 0.6250   
## Prevalence : 0.8861   
## Detection Rate : 0.8713   
## Detection Prevalence : 0.9604   
## Balanced Accuracy : 0.6003   
##   
## 'Positive' Class : No   
##

How does the ensemble perform (with regard to AUC) versus the individual models in the ensemble? How does the model perform on the training and testing sets?

ANSWER: The results for the stacked ensemble model are very similar to the ensemble model. ROC for the stacked ensemble is 0.7784581 , which is almost identical to the ROC for the ensemble model of 0.779. This is less than 0.001 better than the rf model (AUC = 0.7721399) and the nn model (AUC = 0.7721399), but it is 0.05 better than GLM and 0.22 better than the rpart model. The accuracy for the ensemble model using the training data is 0.9894, which is crazy good. The accuracy for the ensemble model using the test data is 0.896. This is a difference of 0.09, which leads to the conclusion that we are overfitting our model.