Parameter Selection, Neural Networks, and Ensemble Models

## Prince Agyabeng

Loading the libraries

library(tidyverse)

## -- 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)

## Loading required package: lattice

##   
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':  
##   
## lift

library(rpart)  
library(caretEnsemble)

##   
## Attaching package: 'caretEnsemble'

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

library(ranger)  
library(nnet)

Loading the data transforming the data

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(as.character(male))) %>%  
mutate(male = fct\_recode(male,  
"male" = "1",  
"female" = "0"))  
  
parole = parole %>% mutate(race = as\_factor(as.character(race))) %>%  
mutate(race = fct\_recode(race,  
"White" = "1",  
"Other" = "2"))  
  
parole = parole %>% mutate(state = as\_factor(as.character(state))) %>%  
mutate(state = fct\_recode(state,  
"Kentucky" = "2",  
"Louisiana" = "3",  
"Virginia" = "4",  
"any other state" = "1"))  
  
parole = parole %>% mutate(multiple.offenses = as\_factor(as.character(multiple.offenses))) %>%  
mutate(multiple.offenses = fct\_recode(multiple.offenses,  
"multiple.offenses" = "1",  
"Otherwise" = "0"))  
  
parole = parole %>% mutate(crime = as\_factor(as.character(crime))) %>%  
mutate(crime = fct\_recode(crime,  
"larceny" = "2",  
"drug-related crime" = "3",  
"drug-related crime" = "4",  
"any other crime" = "1"))  
  
parole = parole %>% mutate(violator = as\_factor(as.character(violator))) %>%  
mutate(violator = fct\_recode(violator,  
"violate\_the\_parole" = "1",  
"completed\_the\_parole\_without\_violation" = "0"))  
str(parole)

## Classes 'spec\_tbl\_df', 'tbl\_df', 'tbl' and 'data.frame': 675 obs. of 9 variables:  
## $ male : Factor w/ 2 levels "male","female": 1 2 1 1 1 1 1 2 2 1 ...  
## $ 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 "any other state",..: 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 "Otherwise","multiple.offenses": 1 1 1 1 1 1 1 1 1 1 ...  
## $ crime : Factor w/ 3 levels "drug-related crime",..: 1 1 1 2 2 1 1 2 1 3 ...  
## $ violator : Factor w/ 2 levels "completed\_the\_parole\_without\_violation",..: 1 1 1 1 1 1 1 1 1 1 ...

Task 1 - Splitting the data

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

Task 2 - Building Neural network

start\_time = Sys.time()   
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 2.162707 secs

nnetBasic

## Neural Network   
##   
## 473 samples  
## 8 predictor  
## 2 classes: 'completed\_the\_parole\_without\_violation', 'violate\_the\_parole'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 425, 426, 426, 426, 425, 427, ...   
## Resampling results:  
##   
## Accuracy Kappa   
## 0.8773377 0.1879936  
##   
## 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 Prediction and Confustion matrix on the basic model

predNetBasic = predict(nnetBasic, train)  
  
confusionMatrix(predNetBasic, train$violator, positive = "completed\_the\_parole\_without\_violation")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction completed\_the\_parole\_without\_violation  
## completed\_the\_parole\_without\_violation 413  
## violate\_the\_parole 5  
## Reference  
## Prediction violate\_the\_parole  
## completed\_the\_parole\_without\_violation 20  
## violate\_the\_parole 35  
##   
## Accuracy : 0.9471   
## 95% CI : (0.923, 0.9655)   
## No Information Rate : 0.8837   
## P-Value [Acc > NIR] : 1.65e-06   
##   
## Kappa : 0.7083   
##   
## Mcnemar's Test P-Value : 0.00511   
##   
## Sensitivity : 0.9880   
## Specificity : 0.6364   
## Pos Pred Value : 0.9538   
## Neg Pred Value : 0.8750   
## Prevalence : 0.8837   
## Detection Rate : 0.8732   
## Detection Prevalence : 0.9154   
## Balanced Accuracy : 0.8122   
##   
## 'Positive' Class : completed\_the\_parole\_without\_violation  
##

The accuracy for the train data is 0.9471 which is good for the model. IT means we are sure that about 95% completed the parole without violation.

Task 4 - changing the size

start\_time = Sys.time()  
fitControl = trainControl(method = "cv",   
 number = 10)  
  
nnetGrid = expand.grid(size = seq(from = 2, to = 12, by = 1),   
 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",  
 trControl = fitControl,  
 tuneGrid = nnetGrid,  
 verbose = FALSE,  
 trace = FALSE)  
  
end\_time = Sys.time()  
end\_time-start\_time

## Time difference of 50.99148 secs

nnetFit

## Neural Network   
##   
## 473 samples  
## 8 predictor  
## 2 classes: 'completed\_the\_parole\_without\_violation', 'violate\_the\_parole'   
##   
## 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 0.8647028 0.1780927  
## 2 0.2 0.8836764 0.2155552  
## 2 0.3 0.8943166 0.2300613  
## 2 0.4 0.8879317 0.1577616  
## 2 0.5 0.8795560 0.0388175  
## 3 0.1 0.8604032 0.1370382  
## 3 0.2 0.8665626 0.1187391  
## 3 0.3 0.8774264 0.1530263  
## 3 0.4 0.8858040 0.1587001  
## 3 0.5 0.8815930 0.1073718  
## 4 0.1 0.8644812 0.1617855  
## 4 0.2 0.8816817 0.2025228  
## 4 0.3 0.8816374 0.1688012  
## 4 0.4 0.8920984 0.1933480  
## 4 0.5 0.8878874 0.1646261  
## 5 0.1 0.8645698 0.1427676  
## 5 0.2 0.8772915 0.1863311  
## 5 0.3 0.8836301 0.2110458  
## 5 0.4 0.8773821 0.1535684  
## 5 0.5 0.8878874 0.1832793  
## 6 0.1 0.8625752 0.1937997  
## 6 0.2 0.8774245 0.2098666  
## 6 0.3 0.8752101 0.1394304  
## 6 0.4 0.8857135 0.1999346  
## 6 0.5 0.8815930 0.1301631  
## 7 0.1 0.8794191 0.2446408  
## 7 0.2 0.8815487 0.2290044  
## 7 0.3 0.8732135 0.1661765  
## 7 0.4 0.8730805 0.1155208  
## 7 0.5 0.8857597 0.1597927  
## 8 0.1 0.8876156 0.3629179  
## 8 0.2 0.8646585 0.1295596  
## 8 0.3 0.8816374 0.2046793  
## 8 0.4 0.8794654 0.1447120  
## 8 0.5 0.8878874 0.1832793  
## 9 0.1 0.8814581 0.2609212  
## 9 0.2 0.8835858 0.2157081  
## 9 0.3 0.8795078 0.1988920  
## 9 0.4 0.8773358 0.1594095  
## 9 0.5 0.8878874 0.1832793  
## 10 0.1 0.8815487 0.2954608  
## 10 0.2 0.8815911 0.2662112  
## 10 0.3 0.8816374 0.2119558  
## 10 0.4 0.8794654 0.1432289  
## 10 0.5 0.8858040 0.1573533  
## 11 0.1 0.8731711 0.2451658  
## 11 0.2 0.8880204 0.2813214  
## 11 0.3 0.8879317 0.2461753  
## 11 0.4 0.8815930 0.1681985  
## 11 0.5 0.8836321 0.1528586  
## 12 0.1 0.8877987 0.2649279  
## 12 0.2 0.8815930 0.2324185  
## 12 0.3 0.8752081 0.1580482  
## 12 0.4 0.8751638 0.1349147  
## 12 0.5 0.8836764 0.1499960  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were size = 2 and decay = 0.3.

Task 5 - Prediction and Confusion on the train data on the fit model

predNet = predict(nnetFit, train)  
  
confusionMatrix(predNet, train$violator, positive = "completed\_the\_parole\_without\_violation")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction completed\_the\_parole\_without\_violation  
## completed\_the\_parole\_without\_violation 412  
## violate\_the\_parole 6  
## Reference  
## Prediction violate\_the\_parole  
## completed\_the\_parole\_without\_violation 44  
## violate\_the\_parole 11  
##   
## Accuracy : 0.8943   
## 95% CI : (0.863, 0.9205)   
## No Information Rate : 0.8837   
## P-Value [Acc > NIR] : 0.2628   
##   
## Kappa : 0.2652   
##   
## Mcnemar's Test P-Value : 1.672e-07   
##   
## Sensitivity : 0.9856   
## Specificity : 0.2000   
## Pos Pred Value : 0.9035   
## Neg Pred Value : 0.6471   
## Prevalence : 0.8837   
## Detection Rate : 0.8710   
## Detection Prevalence : 0.9641   
## Balanced Accuracy : 0.5928   
##   
## 'Positive' Class : completed\_the\_parole\_without\_violation  
##

From the figures above, when the grid search size and decay rates changes, the Accuracy for the prediction is now 0.89 which means that if grid search size and decay rates changes more the accuracy might decrease more thereby affecting the quality of the model.

Task 6 - Prediction and Confusion on the test data on the basic model

predNetBasic = predict(nnetBasic, test)  
  
confusionMatrix(predNetBasic, test$violator, positive = "completed\_the\_parole\_without\_violation")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction completed\_the\_parole\_without\_violation  
## completed\_the\_parole\_without\_violation 173  
## violate\_the\_parole 6  
## Reference  
## Prediction violate\_the\_parole  
## completed\_the\_parole\_without\_violation 15  
## violate\_the\_parole 8  
##   
## Accuracy : 0.896   
## 95% CI : (0.8455, 0.9345)   
## No Information Rate : 0.8861   
## P-Value [Acc > NIR] : 0.37965   
##   
## Kappa : 0.3789   
##   
## Mcnemar's Test P-Value : 0.08086   
##   
## Sensitivity : 0.9665   
## Specificity : 0.3478   
## Pos Pred Value : 0.9202   
## Neg Pred Value : 0.5714   
## Prevalence : 0.8861   
## Detection Rate : 0.8564   
## Detection Prevalence : 0.9307   
## Balanced Accuracy : 0.6572   
##   
## 'Positive' Class : completed\_the\_parole\_without\_violation  
##

Looking at the accuracy for the train data on the basic model which is 0.947 and the accuracy for the test data which is 0.89 for the basic model, it means that the model is good and can be on other data sets.

Task 7 - Prediction and Confusion on the test data for the fit model

predNet = predict(nnetFit, test)  
confusionMatrix(predNet, test$violator, positive = "completed\_the\_parole\_without\_violation")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction completed\_the\_parole\_without\_violation  
## completed\_the\_parole\_without\_violation 177  
## violate\_the\_parole 2  
## Reference  
## Prediction violate\_the\_parole  
## completed\_the\_parole\_without\_violation 16  
## violate\_the\_parole 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 : completed\_the\_parole\_without\_violation  
##

Looking at the accuracy for the train data on the fit model which is 0.894 and the accuracy for the test data which is 0.91 for the fit model, it means that the model is good.

Task 8 - Look at both models From the train data and test data for the basic model and fit model for the predictions, the models are not overfitting and thus can be test on other data.

Task 9 Building an ensemble model - setting the control object

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

Method selection for the ensemble model

set.seed(111)  
model\_list = caretList(x=as.data.frame(train[,-9]), y=as.matrix(train$violator) ,  
 metric = "ROC",   
 trControl= control,  
 methodList=c("glm"),  
tuneList=list(  
rf = caretModelSpec(method="ranger", tuneLength=6),  
rpart = caretModelSpec(method="rpart", tuneLength=6),  
nn = caretModelSpec(method="nnet", tuneLength=6, trace=FALSE)))

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

Checking correlation

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

## rf rpart nn glm  
## 1 0.998 0.9253012 0.9430616 0.9335069  
## 2 0.984 0.9253012 0.9504741 0.8699564  
## 3 0.990 0.9253012 0.9486250 0.8158784  
## 4 0.994 0.9253012 0.9614902 0.9371659  
## 5 0.976 0.9253012 0.9493690 0.9239369  
## 6 0.924 0.9253012 0.6069506 0.7451402

modelCor(resamples(model\_list))

## rf rpart nn glm  
## rf 1.0000000 0.5845768 -0.5214056 0.3201898  
## rpart 0.5845768 1.0000000 -0.7967597 -0.3836722  
## nn -0.5214056 -0.7967597 1.0000000 0.6220834  
## glm 0.3201898 -0.3836722 0.6220834 1.0000000

The model is not correlated so we can build the model now

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

## The following models were ensembled: rf, rpart, nn, glm   
## They were weighted:   
## 2.8073 -3.4768 1.8771 -2.6964 -1.7528  
## The resulting ROC is: 0.8361  
## The fit for each individual model on the ROC is:   
## method ROC ROCSD  
## rf 0.8336710 0.04860020  
## rpart 0.6713712 0.04964340  
## nn 0.8455719 0.02425156  
## glm 0.8399494 0.03033282

Prediction on ensemble for train data

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction completed\_the\_parole\_without\_violation  
## completed\_the\_parole\_without\_violation 418  
## violate\_the\_parole 0  
## Reference  
## Prediction violate\_the\_parole  
## completed\_the\_parole\_without\_violation 17  
## violate\_the\_parole 38  
##   
## Accuracy : 0.9641   
## 95% CI : (0.9431, 0.9789)   
## No Information Rate : 0.8837   
## P-Value [Acc > NIR] : 3.706e-10   
##   
## Kappa : 0.798   
##   
## Mcnemar's Test P-Value : 0.0001042   
##   
## Sensitivity : 1.0000   
## Specificity : 0.6909   
## Pos Pred Value : 0.9609   
## Neg Pred Value : 1.0000   
## Prevalence : 0.8837   
## Detection Rate : 0.8837   
## Detection Prevalence : 0.9197   
## Balanced Accuracy : 0.8455   
##   
## 'Positive' Class : completed\_the\_parole\_without\_violation  
##

Prediction on ensemble for test data

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction completed\_the\_parole\_without\_violation  
## completed\_the\_parole\_without\_violation 176  
## violate\_the\_parole 3  
## Reference  
## Prediction violate\_the\_parole  
## completed\_the\_parole\_without\_violation 16  
## violate\_the\_parole 7  
##   
## Accuracy : 0.9059   
## 95% CI : (0.857, 0.9424)   
## No Information Rate : 0.8861   
## P-Value [Acc > NIR] : 0.222456   
##   
## Kappa : 0.3816   
##   
## Mcnemar's Test P-Value : 0.005905   
##   
## Sensitivity : 0.9832   
## Specificity : 0.3043   
## Pos Pred Value : 0.9167   
## Neg Pred Value : 0.7000   
## Prevalence : 0.8861   
## Detection Rate : 0.8713   
## Detection Prevalence : 0.9505   
## Balanced Accuracy : 0.6438   
##   
## 'Positive' Class : completed\_the\_parole\_without\_violation  
##

The ensemble model performed well on prediction for both the train data and test data for the accuracy.

Task 10 Building a stacked ensemble mode

stack = caretStack(  
 model\_list,   
 method ="glm",   
 metric ="ROC",   
 trControl = trainControl(  
 method = "cv",  
 number = 5,   
 savePredictions = "final",  
 classProbs = TRUE,  
 summaryFunction = twoClassSummary  
 )  
)  
  
print(stack)

## A glm ensemble of 4 base models: rf, rpart, nn, glm  
##   
## Ensemble results:  
## Generalized Linear Model   
##   
## 473 samples  
## 4 predictor  
## 2 classes: 'completed\_the\_parole\_without\_violation', 'violate\_the\_parole'   
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 378, 379, 378, 379, 378   
## Resampling results:  
##   
## ROC Sens Spec   
## 0.8377979 0.9688755 0.2545455

The ensemble model perform equally good as the stack model

Train data on the stack model

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction completed\_the\_parole\_without\_violation  
## completed\_the\_parole\_without\_violation 418  
## violate\_the\_parole 0  
## Reference  
## Prediction violate\_the\_parole  
## completed\_the\_parole\_without\_violation 17  
## violate\_the\_parole 38  
##   
## Accuracy : 0.9641   
## 95% CI : (0.9431, 0.9789)   
## No Information Rate : 0.8837   
## P-Value [Acc > NIR] : 3.706e-10   
##   
## Kappa : 0.798   
##   
## Mcnemar's Test P-Value : 0.0001042   
##   
## Sensitivity : 1.0000   
## Specificity : 0.6909   
## Pos Pred Value : 0.9609   
## Neg Pred Value : 1.0000   
## Prevalence : 0.8837   
## Detection Rate : 0.8837   
## Detection Prevalence : 0.9197   
## Balanced Accuracy : 0.8455   
##   
## 'Positive' Class : completed\_the\_parole\_without\_violation  
##

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction completed\_the\_parole\_without\_violation  
## completed\_the\_parole\_without\_violation 176  
## violate\_the\_parole 3  
## Reference  
## Prediction violate\_the\_parole  
## completed\_the\_parole\_without\_violation 16  
## violate\_the\_parole 7  
##   
## Accuracy : 0.9059   
## 95% CI : (0.857, 0.9424)   
## No Information Rate : 0.8861   
## P-Value [Acc > NIR] : 0.222456   
##   
## Kappa : 0.3816   
##   
## Mcnemar's Test P-Value : 0.005905   
##   
## Sensitivity : 0.9832   
## Specificity : 0.3043   
## Pos Pred Value : 0.9167   
## Neg Pred Value : 0.7000   
## Prevalence : 0.8861   
## Detection Rate : 0.8713   
## Detection Prevalence : 0.9505   
## Balanced Accuracy : 0.6438   
##   
## 'Positive' Class : completed\_the\_parole\_without\_violation  
##

Looking at the accuracy for the train data on the stack prediction which is 0.96 and the accuracy for the test data which is 0.91 for the fit model, it means that the model is good.