Classification Trees

## Prince Agyabeng

### Classification Trees

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(RColorBrewer)   
library(rattle)

## Rattle: A free graphical interface for data science with R.  
## Version 5.3.0 Copyright (c) 2006-2018 Togaware Pty Ltd.  
## Type 'rattle()' to shake, rattle, and roll your data.

Loading 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

Renaming the factor level of each variable

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,  
"violated the parole" = "1",  
"completed the parole without violation" = "0"))  
str(parole)

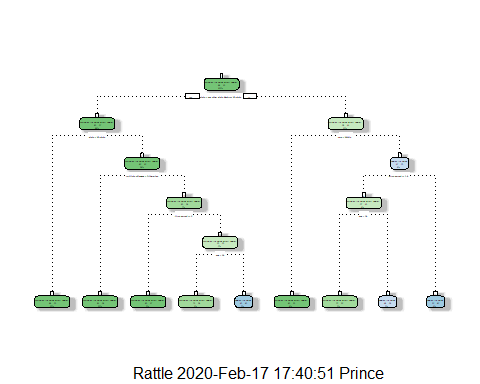
## Classes 'spec\_tbl\_df', 'tbl\_df', 'tbl' and 'data.frame': 675 obs. of 9 variables:  
## $ male : Factor w/ 2 levels "male","female": 1 2 1 1 1 1 1 2 2 1 ...  
## $ race : Factor w/ 2 levels "White","Other": 1 1 2 1 2 2 1 1 1 2 ...  
## $ age : num 33.2 39.7 29.5 22.4 21.6 46.7 31 24.6 32.6 29.1 ...  
## $ 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 - Training the data

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

Creating a Classification Tree

tree1 = rpart(violator ~., train, method="class")  
fancyRpartPlot(tree1)



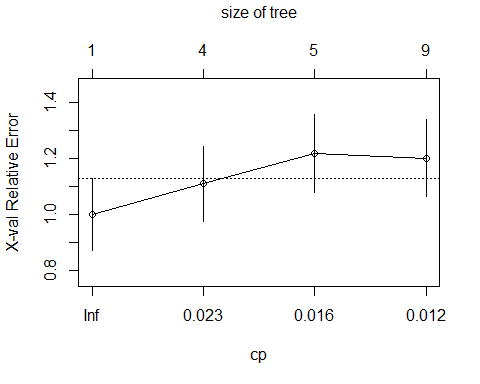
Task 3 In getting an anwer for 40 years Male in Louisiana state and spent 5 years in prison. We have to look at the Classification tree is the man violated the parole, how old was he, what state was he convicted and how many years was he in prison.

Task 4 - Evaluating the Tree Performance

printcp(tree1)

##   
## Classification tree:  
## rpart(formula = violator ~ ., data = train, method = "class")  
##   
## Variables actually used in tree construction:  
## [1] age multiple.offenses race state   
## [5] time.served   
##   
## Root node error: 55/473 = 0.11628  
##   
## n= 473   
##   
## CP nsplit rel error xerror xstd  
## 1 0.030303 0 1.00000 1.0000 0.12676  
## 2 0.018182 3 0.90909 1.1091 0.13253  
## 3 0.013636 4 0.89091 1.2182 0.13788  
## 4 0.010000 8 0.83636 1.2000 0.13702

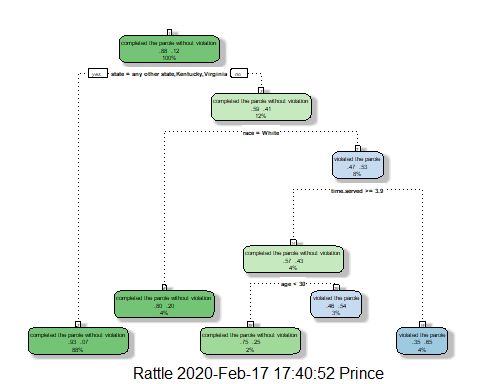
plotcp(tree1)



A higher CP should be selected as small CP leads to overfitting of the model. A selection of CP value of 0.018 wil be better for the above model.

Prunning the Tree

tree2 = rpart(violator ~., train, cp=0.018, method="class")  
fancyRpartPlot(tree2)



summary(tree2)

## Call:  
## rpart(formula = violator ~ ., data = train, method = "class",   
## cp = 0.018)  
## n= 473   
##   
## CP nsplit rel error xerror xstd  
## 1 0.03030303 0 1.0000000 1.000000 0.1267582  
## 2 0.01818182 3 0.9090909 1.236364 0.1387359  
## 3 0.01800000 4 0.8909091 1.272727 0.1404133  
##   
## Variable importance  
## state max.sentence race time.served age   
## 50 22 12 10 5   
##   
## Node number 1: 473 observations, complexity param=0.03030303  
## predicted class=completed the parole without violation expected loss=0.1162791 P(node) =1  
## class counts: 418 55  
## probabilities: 0.884 0.116   
## left son=2 (415 obs) right son=3 (58 obs)  
## Primary splits:  
## state splits as LLRL, improve=11.7027000, (0 missing)  
## time.served < 2.25 to the right, improve= 3.8807960, (0 missing)  
## max.sentence < 12.5 to the right, improve= 2.8491650, (0 missing)  
## multiple.offenses splits as LR, improve= 1.0008520, (0 missing)  
## race splits as LR, improve= 0.7327395, (0 missing)  
## Surrogate splits:  
## max.sentence < 11.5 to the right, agree=0.922, adj=0.362, (0 split)  
## time.served < 1.75 to the right, agree=0.890, adj=0.103, (0 split)  
##   
## Node number 2: 415 observations  
## predicted class=completed the parole without violation expected loss=0.0746988 P(node) =0.8773784  
## class counts: 384 31  
## probabilities: 0.925 0.075   
##   
## Node number 3: 58 observations, complexity param=0.03030303  
## predicted class=completed the parole without violation expected loss=0.4137931 P(node) =0.1226216  
## class counts: 34 24  
## probabilities: 0.586 0.414   
## left son=6 (20 obs) right son=7 (38 obs)  
## Primary splits:  
## race splits as LR, improve=2.7905630, (0 missing)  
## multiple.offenses splits as LR, improve=1.5324790, (0 missing)  
## max.sentence < 5 to the left, improve=1.4609480, (0 missing)  
## age < 38.55 to the left, improve=0.7155289, (0 missing)  
## time.served < 2.45 to the right, improve=0.7155289, (0 missing)  
## Surrogate splits:  
## max.sentence < 2.5 to the left, agree=0.759, adj=0.30, (0 split)  
## age < 43.7 to the right, agree=0.672, adj=0.05, (0 split)  
##   
## Node number 6: 20 observations  
## predicted class=completed the parole without violation expected loss=0.2 P(node) =0.0422833  
## class counts: 16 4  
## probabilities: 0.800 0.200   
##   
## Node number 7: 38 observations, complexity param=0.03030303  
## predicted class=violated the parole expected loss=0.4736842 P(node) =0.08033827  
## class counts: 18 20  
## probabilities: 0.474 0.526   
## left son=14 (21 obs) right son=15 (17 obs)  
## Primary splits:  
## time.served < 3.85 to the right, improve=0.8969483, (0 missing)  
## max.sentence < 8.5 to the left, improve=0.7908467, (0 missing)  
## male splits as LR, improve=0.6063546, (0 missing)  
## age < 43.45 to the left, improve=0.6063546, (0 missing)  
## multiple.offenses splits as LR, improve=0.4330827, (0 missing)  
## Surrogate splits:  
## age < 28.6 to the right, agree=0.632, adj=0.176, (0 split)  
## max.sentence < 10.5 to the left, agree=0.632, adj=0.176, (0 split)  
## male splits as RL, agree=0.579, adj=0.059, (0 split)  
## crime splits as RLL, agree=0.579, adj=0.059, (0 split)  
##   
## Node number 14: 21 observations, complexity param=0.01818182  
## predicted class=completed the parole without violation expected loss=0.4285714 P(node) =0.04439746  
## class counts: 12 9  
## probabilities: 0.571 0.429   
## left son=28 (8 obs) right son=29 (13 obs)  
## Primary splits:  
## age < 29.75 to the left, improve=0.8241758, (0 missing)  
## max.sentence < 8.5 to the left, improve=0.6311688, (0 missing)  
## multiple.offenses splits as LR, improve=0.4285714, (0 missing)  
## time.served < 5.25 to the left, improve=0.2857143, (0 missing)  
## crime splits as RLR, improve=0.1948052, (0 missing)  
## Surrogate splits:  
## time.served < 4.15 to the left, agree=0.762, adj=0.375, (0 split)  
## multiple.offenses splits as LR, agree=0.667, adj=0.125, (0 split)  
##   
## Node number 15: 17 observations  
## predicted class=violated the parole expected loss=0.3529412 P(node) =0.0359408  
## class counts: 6 11  
## probabilities: 0.353 0.647   
##   
## Node number 28: 8 observations  
## predicted class=completed the parole without violation expected loss=0.25 P(node) =0.01691332  
## class counts: 6 2  
## probabilities: 0.750 0.250   
##   
## Node number 29: 13 observations  
## predicted class=violated the parole expected loss=0.4615385 P(node) =0.02748414  
## class counts: 6 7  
## probabilities: 0.462 0.538

Nod number one has the most observation in the training set.

Task 6 - Predictions and Confusion matrix

treepred = predict(tree1, train, type = "class")  
head(treepred)

## 1 2   
## completed the parole without violation completed the parole without violation   
## 3 4   
## completed the parole without violation completed the parole without violation   
## 5 6   
## completed the parole without violation completed the parole without violation   
## Levels: completed the parole without violation violated the parole

confusionMatrix(treepred,train$violator,positive="completed the parole without violation")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction completed the parole without violation  
## completed the parole without violation 400  
## violated the parole 18  
## Reference  
## Prediction violated the parole  
## completed the parole without violation 28  
## violated the parole 27  
##   
## Accuracy : 0.9027   
## 95% CI : (0.8724, 0.9279)   
## No Information Rate : 0.8837   
## P-Value [Acc > NIR] : 0.1095   
##   
## Kappa : 0.4862   
##   
## Mcnemar's Test P-Value : 0.1845   
##   
## Sensitivity : 0.9569   
## Specificity : 0.4909   
## Pos Pred Value : 0.9346   
## Neg Pred Value : 0.6000   
## Prevalence : 0.8837   
## Detection Rate : 0.8457   
## Detection Prevalence : 0.9049   
## Balanced Accuracy : 0.7239   
##   
## 'Positive' Class : completed the parole without violation  
##

Task 7 - Prediction for Testing data

treepred2 = predict(tree1, test, type = "class")  
head(treepred2)

## 1 2   
## completed the parole without violation completed the parole without violation   
## 3 4   
## completed the parole without violation completed the parole without violation   
## 5 6   
## completed the parole without violation completed the parole without violation   
## Levels: completed the parole without violation violated the parole

confusionMatrix(treepred2,test$violator,positive="completed the parole without violation")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction completed the parole without violation  
## completed the parole without violation 171  
## violated the parole 8  
## Reference  
## Prediction violated the parole  
## completed the parole without violation 13  
## violated the parole 10  
##   
## Accuracy : 0.896   
## 95% CI : (0.8455, 0.9345)   
## No Information Rate : 0.8861   
## P-Value [Acc > NIR] : 0.3797   
##   
## Kappa : 0.4309   
##   
## Mcnemar's Test P-Value : 0.3827   
##   
## Sensitivity : 0.9553   
## Specificity : 0.4348   
## Pos Pred Value : 0.9293   
## Neg Pred Value : 0.5556   
## Prevalence : 0.8861   
## Detection Rate : 0.8465   
## Detection Prevalence : 0.9109   
## Balanced Accuracy : 0.6950   
##   
## 'Positive' Class : completed the parole without violation  
##

The accuracy for the test data is 0.869 which is lower than the accuracy of the train data of 0.902

Task 8 Loading the Blood data

Blood <- read\_csv("Blood.csv")

## Parsed with column specification:  
## cols(  
## Mnths\_Since\_Last = col\_double(),  
## TotalDonations = col\_double(),  
## Total\_Donated = col\_double(),  
## Mnths\_Since\_First = col\_double(),  
## DonatedMarch = col\_double()  
## )

str(Blood)

## Classes 'spec\_tbl\_df', 'tbl\_df', 'tbl' and 'data.frame': 748 obs. of 5 variables:  
## $ Mnths\_Since\_Last : num 2 0 1 2 1 4 2 1 2 5 ...  
## $ TotalDonations : num 50 13 16 20 24 4 7 12 9 46 ...  
## $ Total\_Donated : num 12500 3250 4000 5000 6000 1000 1750 3000 2250 11500 ...  
## $ Mnths\_Since\_First: num 98 28 35 45 77 4 14 35 22 98 ...  
## $ DonatedMarch : num 1 1 1 1 0 0 1 0 1 1 ...  
## - attr(\*, "spec")=  
## .. cols(  
## .. Mnths\_Since\_Last = col\_double(),  
## .. TotalDonations = col\_double(),  
## .. Total\_Donated = col\_double(),  
## .. Mnths\_Since\_First = col\_double(),  
## .. DonatedMarch = col\_double()  
## .. )

summary(Blood)

## Mnths\_Since\_Last TotalDonations Total\_Donated Mnths\_Since\_First  
## Min. : 0.000 Min. : 1.000 Min. : 250 Min. : 2.00   
## 1st Qu.: 2.750 1st Qu.: 2.000 1st Qu.: 500 1st Qu.:16.00   
## Median : 7.000 Median : 4.000 Median : 1000 Median :28.00   
## Mean : 9.507 Mean : 5.515 Mean : 1379 Mean :34.28   
## 3rd Qu.:14.000 3rd Qu.: 7.000 3rd Qu.: 1750 3rd Qu.:50.00   
## Max. :74.000 Max. :50.000 Max. :12500 Max. :98.00   
## DonatedMarch   
## Min. :0.000   
## 1st Qu.:0.000   
## Median :0.000   
## Mean :0.238   
## 3rd Qu.:0.000   
## Max. :1.000

Blood = Blood %>% mutate(DonatedMarch = as.factor(DonatedMarch )) %>%   
 mutate(DonatedMarch = fct\_recode(DonatedMarch , "No" = "0", "Yes" = "1" ))  
str(Blood)

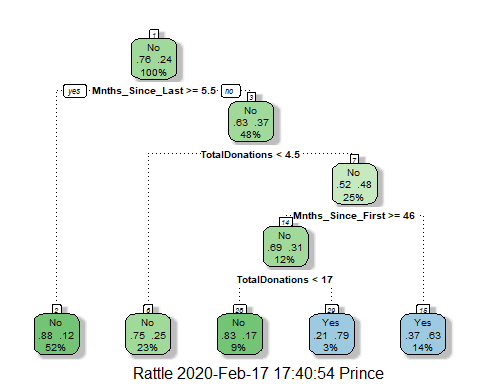
## Classes 'spec\_tbl\_df', 'tbl\_df', 'tbl' and 'data.frame': 748 obs. of 5 variables:  
## $ Mnths\_Since\_Last : num 2 0 1 2 1 4 2 1 2 5 ...  
## $ TotalDonations : num 50 13 16 20 24 4 7 12 9 46 ...  
## $ Total\_Donated : num 12500 3250 4000 5000 6000 1000 1750 3000 2250 11500 ...  
## $ Mnths\_Since\_First: num 98 28 35 45 77 4 14 35 22 98 ...  
## $ DonatedMarch : Factor w/ 2 levels "No","Yes": 2 2 2 2 1 1 2 1 2 2 ...

Task 9 - Training the Blood data

set.seed(1234)  
train.rows = createDataPartition(y = Blood$DonatedMarch, p=0.7, list = FALSE)  
train2 = Blood[train.rows,]   
test2 = Blood[-train.rows,]

Classification Tree for Blood data

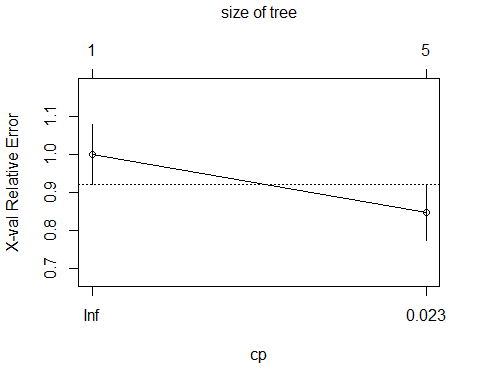
tree3 = rpart(DonatedMarch ~., train2, method="class")  
fancyRpartPlot(tree3)



printcp(tree3)

##   
## Classification tree:  
## rpart(formula = DonatedMarch ~ ., data = train2, method = "class")  
##   
## Variables actually used in tree construction:  
## [1] Mnths\_Since\_First Mnths\_Since\_Last TotalDonations   
##   
## Root node error: 125/524 = 0.23855  
##   
## n= 524   
##   
## CP nsplit rel error xerror xstd  
## 1 0.050667 0 1.000 1.000 0.078049  
## 2 0.010000 4 0.784 0.848 0.073564

plotcp(tree3)



Task 10 - Pruning the Classification Tree for Blood data

tree4 = rpart(DonatedMarch ~., train2, cp=0.050667, method="class")

Confusion Matrix for Blood Data

treepred3 = predict(tree3, train2, type = "class")  
head(treepred3)

## 1 2 3 4 5 6   
## Yes Yes Yes Yes Yes Yes   
## Levels: No Yes

confusionMatrix(treepred3,train2$DonatedMarch,positive="Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 370 69  
## Yes 29 56  
##   
## Accuracy : 0.813   
## 95% CI : (0.7769, 0.8455)  
## No Information Rate : 0.7615   
## P-Value [Acc > NIR] : 0.002713   
##   
## Kappa : 0.4216   
##   
## Mcnemar's Test P-Value : 8.162e-05   
##   
## Sensitivity : 0.4480   
## Specificity : 0.9273   
## Pos Pred Value : 0.6588   
## Neg Pred Value : 0.8428   
## Prevalence : 0.2385   
## Detection Rate : 0.1069   
## Detection Prevalence : 0.1622   
## Balanced Accuracy : 0.6877   
##   
## 'Positive' Class : Yes   
##

treepred4 = predict(tree3, test2, type = "class")  
head(treepred4)

## 1 2 3 4 5 6   
## Yes No Yes Yes No Yes   
## Levels: No Yes

confusionMatrix(treepred4,test2$DonatedMarch,positive="Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 156 35  
## Yes 15 18  
##   
## Accuracy : 0.7768   
## 95% CI : (0.7165, 0.8296)  
## No Information Rate : 0.7634   
## P-Value [Acc > NIR] : 0.35155   
##   
## Kappa : 0.2896   
##   
## Mcnemar's Test P-Value : 0.00721   
##   
## Sensitivity : 0.33962   
## Specificity : 0.91228   
## Pos Pred Value : 0.54545   
## Neg Pred Value : 0.81675   
## Prevalence : 0.23661   
## Detection Rate : 0.08036   
## Detection Prevalence : 0.14732   
## Balanced Accuracy : 0.62595   
##   
## 'Positive' Class : Yes   
##

The accuracy for the test data is 0.777 which is lower than the accuracy of the train data of 0.813