Confusion Matrix & ROC

Nikhila Reddy

2024-02-25

##loading packages

library(ISLR)  
library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

library(class)

##summary for the data

View(Default)  
summary(Default)

## default student balance income   
## No :9667 No :7056 Min. : 0.0 Min. : 772   
## Yes: 333 Yes:2944 1st Qu.: 481.7 1st Qu.:21340   
## Median : 823.6 Median :34553   
## Mean : 835.4 Mean :33517   
## 3rd Qu.:1166.3 3rd Qu.:43808   
## Max. :2654.3 Max. :73554

norm\_model <- preProcess(Default, method = c('range'))  
Default\_normalized <- predict(norm\_model,Default)  
summary(Default\_normalized)

## default student balance income   
## No :9667 No :7056 Min. :0.0000 Min. :0.0000   
## Yes: 333 Yes:2944 1st Qu.:0.1815 1st Qu.:0.2826   
## Median :0.3103 Median :0.4641   
## Mean :0.3147 Mean :0.4499   
## 3rd Qu.:0.4394 3rd Qu.:0.5913   
## Max. :1.0000 Max. :1.0000

head(Default\_normalized)

## default student balance income  
## 1 No No 0.2748447 0.59890492  
## 2 No Yes 0.3078678 0.15572704  
## 3 No No 0.4044532 0.42586159  
## 4 No No 0.1993920 0.47995931  
## 5 No No 0.2959911 0.51786692  
## 6 No Yes 0.3464494 0.09232456

set.seed(123)  
model <- train(default~balance+income, data= Default\_normalized, method = "knn")

set.seed(123)  
Serach\_grid <- expand.grid(k=c(2,7,9,15))  
model <- train(default~balance+income, data = Default\_normalized,  
method="knn", tuneGrid=Serach\_grid)  
model

## k-Nearest Neighbors   
##   
## 10000 samples  
## 2 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Bootstrapped (25 reps)   
## Summary of sample sizes: 10000, 10000, 10000, 10000, 10000, 10000, ...   
## Resampling results across tuning parameters:  
##   
## k Accuracy Kappa   
## 2 0.9555126 0.2972410  
## 7 0.9671091 0.3708247  
## 9 0.9694620 0.3958578  
## 15 0.9707537 0.3959363  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was k = 15.

set.seed(123)  
Serach\_grid <- expand.grid(k=c(2:20))  
model<- train(default~balance+income, data=Default\_normalized,  
method="knn", tuneGrid=Serach\_grid)  
print(model)

## k-Nearest Neighbors   
##   
## 10000 samples  
## 2 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Bootstrapped (25 reps)   
## Summary of sample sizes: 10000, 10000, 10000, 10000, 10000, 10000, ...   
## Resampling results across tuning parameters:  
##   
## k Accuracy Kappa   
## 2 0.9554380 0.3005431  
## 3 0.9582678 0.3138866  
## 4 0.9605429 0.3278467  
## 5 0.9636439 0.3463714  
## 6 0.9649411 0.3515434  
## 7 0.9670004 0.3716492  
## 8 0.9682077 0.3823089  
## 9 0.9693432 0.3944818  
## 10 0.9693327 0.3894331  
## 11 0.9697672 0.3934384  
## 12 0.9698426 0.3886953  
## 13 0.9703201 0.3966232  
## 14 0.9706349 0.3985500  
## 15 0.9706779 0.3935936  
## 16 0.9708535 0.3948190  
## 17 0.9712224 0.3996117  
## 18 0.9712117 0.3966457  
## 19 0.9712441 0.3947316  
## 20 0.9713972 0.3989892  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was k = 20.

Default\_normalized <- Default\_normalized[, -2]  
Index\_Train <-createDataPartition(Default\_normalized$default, p=0.8,  
list=FALSE)  
Train <- Default\_normalized[Index\_Train,]  
Test <- Default\_normalized[-Index\_Train,]  
head(Train)

## default balance income  
## 2 No 0.3078678 0.15572704  
## 3 No 0.4044532 0.42586159  
## 4 No 0.1993920 0.47995931  
## 6 No 0.3464494 0.09232456  
## 8 No 0.3046606 0.23121681  
## 9 No 0.4374215 0.50419647

head(Test)

## default balance income  
## 1 No 0.2748447 0.5989049  
## 5 No 0.2959911 0.5178669  
## 7 No 0.3110072 0.3315816  
## 12 No 0.4598476 0.1716983  
## 16 No 0.1078364 0.6082587  
## 20 No 0.4125620 0.3530072

Train\_Predictors <- Train[, 2:3]  
Test\_Predictors <- Test[, 2:3]  
Train\_labels <- Train[,1]  
Test\_labels <- Test[,1]  
  
Predicted\_Test\_labels <- knn(Train\_Predictors, Test\_Predictors,  
cl=Train\_labels, k=4)  
head(Predicted\_Test\_labels)

## [1] No No No No No No  
## Levels: No Yes

chooseCRANmirror(ind=1)  
library("gmodels")

CrossTable(x=Test\_labels, y=Predicted\_Test\_labels, prop.chisq = FALSE)

##   
##   
## Cell Contents  
## |-------------------------|  
## | N |  
## | N / Row Total |  
## | N / Col Total |  
## | N / Table Total |  
## |-------------------------|  
##   
##   
## Total Observations in Table: 1999   
##   
##   
## | Predicted\_Test\_labels   
## Test\_labels | No | Yes | Row Total |   
## -------------|-----------|-----------|-----------|  
## No | 1920 | 13 | 1933 |   
## | 0.993 | 0.007 | 0.967 |   
## | 0.976 | 0.419 | |   
## | 0.960 | 0.007 | |   
## -------------|-----------|-----------|-----------|  
## Yes | 48 | 18 | 66 |   
## | 0.727 | 0.273 | 0.033 |   
## | 0.024 | 0.581 | |   
## | 0.024 | 0.009 | |   
## -------------|-----------|-----------|-----------|  
## Column Total | 1968 | 31 | 1999 |   
## | 0.984 | 0.016 | |   
## -------------|-----------|-----------|-----------|  
##   
##

library(class)  
Predicted\_Test\_labels <- knn (Train\_Predictors, Test\_Predictors,cl=Train\_labels, k=100, prob = TRUE)  
class\_prob <- attr(Predicted\_Test\_labels, "prob")  
head(class\_prob)

## [1] 1.00 1.00 1.00 0.99 1.00 1.00

##Creating confusion matrix for the data

library(caret)  
example <- confusionMatrix(data = Predicted\_Test\_labels,  
reference=Test\_labels)  
example

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 1933 57  
## Yes 0 9  
##   
## Accuracy : 0.9715   
## 95% CI : (0.9632, 0.9783)  
## No Information Rate : 0.967   
## P-Value [Acc > NIR] : 0.1429   
##   
## Kappa : 0.2339   
##   
## Mcnemar's Test P-Value : 1.195e-13   
##   
## Sensitivity : 1.0000   
## Specificity : 0.1364   
## Pos Pred Value : 0.9714   
## Neg Pred Value : 1.0000   
## Prevalence : 0.9670   
## Detection Rate : 0.9670   
## Detection Prevalence : 0.9955   
## Balanced Accuracy : 0.5682   
##   
## 'Positive' Class : No   
##

library(pROC) #loading pROC package

## Type 'citation("pROC")' for a citation.

##   
## Attaching package: 'pROC'

## The following object is masked from 'package:gmodels':  
##   
## ci

## The following objects are masked from 'package:stats':  
##   
## cov, smooth, var

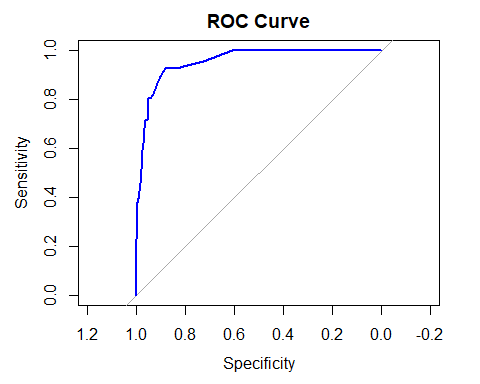
##Creating ROC curve for the data

roc\_curve <- roc(Test\_labels, class\_prob)

## Setting levels: control = No, case = Yes

## Setting direction: controls > cases

plot(roc\_curve, ,main = "ROC Curve", col = "blue")



roc\_curve <- roc(Predicted\_Test\_labels, class\_prob)

## Setting levels: control = No, case = Yes

## Setting direction: controls > cases

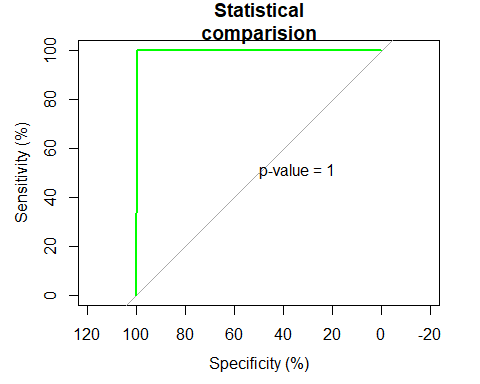
roc1 <- plot.roc(Predicted\_Test\_labels, class\_prob, main= "Statistical  
comparision",  
 percent=TRUE,  
 col="blue")

## Setting levels: control = No, case = Yes  
## Setting direction: controls > cases

roc2 <- lines.roc(Predicted\_Test\_labels, class\_prob, percent=TRUE,  
 col="green")

## Setting levels: control = No, case = Yes  
## Setting direction: controls > cases

testobj <- roc.test(roc1, roc2)   
text(50, 50, labels=paste("p-value =", format.pval(testobj$p.value)),  
adj=c(0, .2))



plot.roc(Predicted\_Test\_labels, class\_prob, main = "Confidence interval of a threshold", percent=TRUE,  
ci=TRUE, of="thresholds",  
thresholds="best",  
print.thres="best")

## Setting levels: control = No, case = Yes

## Setting direction: controls > cases

