Assignment 3

Saipriya Gourineni

2022-10-17

```
library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
library(lattice)
library(knitr)
library(rmarkdown)
library(e1071)
```

Now I'll load the UniversalBank.csv file.

```
getwd()
```

```
## [1] "C:/Users/Saipr/OneDrive/Desktop/New folder"
```

```
setwd("C:/Users/Saipr/OneDrive/Desktop/New folder")
Original <- read.csv("UniversalBank.csv")
DF_Universal_Bank <- Original %>% select(Age, Experience, Income, Family, CCAvg, Education, Mortgage, Pe
DF_Universal_Bank$CreditCard <- as.factor(DF_Universal_Bank$CreditCard)
DF_Universal_Bank$Personal.Loan <- as.factor((DF_Universal_Bank$Personal.Loan))
DF_Universal_Bank$Online <- as.factor(DF_Universal_Bank$Online)
```

Removing ID and ZipCode ##Create Partition

```
selected.var <- c(8,11,12)
set.seed(20)
Train_Index = createDataPartition(DF_Universal_Bank$Personal.Loan, p=0.60, list=FALSE)
Train_Data = DF_Universal_Bank[Train_Index,selected.var]
Validation_Data = DF_Universal_Bank[-Train_Index,selected.var]
```

Creates the data partition, train data and validation data ##A

```
attach(Train_Data)
ftable(CreditCard,Personal.Loan,Online)
```

```
##                          Online    0    1
## CreditCard Personal.Loan                 
## 0          0                     800 1126
##            1                      82  121
## 1          0                     308  478
##            1                      36   49
```

```
detach(Train_Data)
```

The pivot table is now created with online as a column and CC and LOAN as rows.

B) (probability not using Naive Bayes) With Online=1 and CC=1, we can calculate the likelihood that Loan=1 by , we add 53(Loan=1 from ftable) and 497(Loan=0 from ftable) which gives us 550. So the probability is 53/(53+497) =53/550 = 0.096363 or 9.64% . Hence the probability is 9.64%

```
prop.table(ftable(Train_Data$CreditCard,Train_Data$Online,Train_Data$Personal.Loan),margin=1)
```

```
##                0          1
##                            
## 0 0   0.90702948 0.09297052
##   1   0.90296712 0.09703288
## 1 0   0.89534884 0.10465116
##   1   0.90702087 0.09297913
```

The code above gives a proportion pivot table that can assist in answering question B.This table shows the chances of getting a loan if you have a credit card and you apply online. ##C)

```
attach(Train_Data)
ftable(Personal.Loan,Online)
```

```
##               Online    0    1
## Personal.Loan                 
## 0                    1108 1604
## 1                     118  170
```

```
ftable(Personal.Loan,CreditCard)
```

```
##               CreditCard    0    1
## Personal.Loan                     
## 0                        1926  786
## 1                         203   85
```

```
detach(Train_Data)
```

The two pivot tables necessary for C are returned above. The first is a column with Online as a column and Loans as a row, while the second is a column with Credit Card as a column. ##D

```
prop.table(ftable(Train_Data$Personal.Loan,Train_Data$CreditCard),margin=1)
```

```
##             0         1
##
## 0  0.7101770 0.2898230
## 1  0.7048611 0.2951389
```

```
prop.table(ftable(Train_Data$Personal.Loan,Train_Data$Online),margin=1)
```

```
##             0         1
##
## 0  0.4085546 0.5914454
## 1  0.4097222 0.5902778
```

The code above displays a proportion pivot table that can assist in answering question D. Di) 92/288 = 0.3194 or 31.94%

Dii) 167/288 = 0.5798 or 57.986%

Diii) total loans= 1 from table (288) is now divided by total count from table (3000) = 0.096 or 9.6%

DiV) 812/2712 = 0.2994 or 29.94%

DV) 1624/2712 = 0.5988 or 59.88%

DVi) total loans=0 from table(2712) which is divided by total count from table (3000) = 0.904 or 90.4%

##E)Naive Bayes calculation (0.3194 * 0.5798 * 0.096)/[(0.3194 * 0.5798 * 0.096)+(0.2994 * 0.5988 * 0.904)] = 0.0988505642823701 or 9.885%

##F) B employs a direct computation based on a count, whereas E employs probability for each of the counts. As a result, whereas E is ideal for broad generality, B is more precise.

##G)

```
Universal.nb <- naiveBayes(Personal.Loan ~ ., data = Train_Data)
Universal.nb
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##     0     1
## 0.904 0.096
##
```

```
## Conditional probabilities:
##    Online
## Y           0         1
##   0 0.4085546 0.5914454
##   1 0.4097222 0.5902778
##
##    CreditCard
## Y           0         1
##   0 0.7101770 0.2898230
##   1 0.7048611 0.2951389
```

While using the two tables made in step C makes it simple to understand how you're computing P(LOAN=1|CC=1,Online=1) using the Naive Bayes model, you can also quickly compute P(LOAN=1|CC=1,Online=1) using the pivot table made in step B. The probability predicted by the Naive Bayes model is lower than that determined manually in step E, but it is the same as that predicted by the earlier methods. This likelihood is more in line with the one determined in step B. This might be because we are manually performing the computations in step E, which gives room for error when rounding fractions and leads to approximations. ## NB confusion matrix for Train_Data

```
pred.class <- predict(Universal.nb, newdata = Train_Data)
confusionMatrix(pred.class, Train_Data$Personal.Loan)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2712  288
##          1    0    0
##
##                Accuracy : 0.904
##                  95% CI : (0.8929, 0.9143)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : 0.5157
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 1.000
##             Specificity : 0.000
##          Pos Pred Value : 0.904
##          Neg Pred Value :   NaN
##              Prevalence : 0.904
##          Detection Rate : 0.904
##    Detection Prevalence : 1.000
##       Balanced Accuracy : 0.500
##
##        'Positive' Class : 0
##
```

Despite its high sensitivity, this model has a low specificity. The reference had all true values while the model predicted that all values would be 0. The model still yields a 90.4 percent accuracy despite lacking all 1 data because of the vast majority of 0 values. ##Validation set

```r
pred.prob <- predict(Universal.nb, newdata=Validation_Data, type="raw")
pred.class <- predict(Universal.nb, newdata = Validation_Data)
confusionMatrix(pred.class, Validation_Data$Personal.Loan)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1808  192
##          1    0    0
##
##                Accuracy : 0.904
##                  95% CI : (0.8902, 0.9166)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : 0.5192
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 1.000
##             Specificity : 0.000
##          Pos Pred Value : 0.904
##          Neg Pred Value :   NaN
##              Prevalence : 0.904
##          Detection Rate : 0.904
##    Detection Prevalence : 1.000
##       Balanced Accuracy : 0.500
##
##        'Positive' Class : 0
##
```

Let's look at the model graphically and see what the best threshold is for it.

##ROC

```r
library(pROC)
```

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

```
##
## Attaching package: 'pROC'
```

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

```r
roc(Validation_Data$Personal.Loan,pred.prob[,1])
```

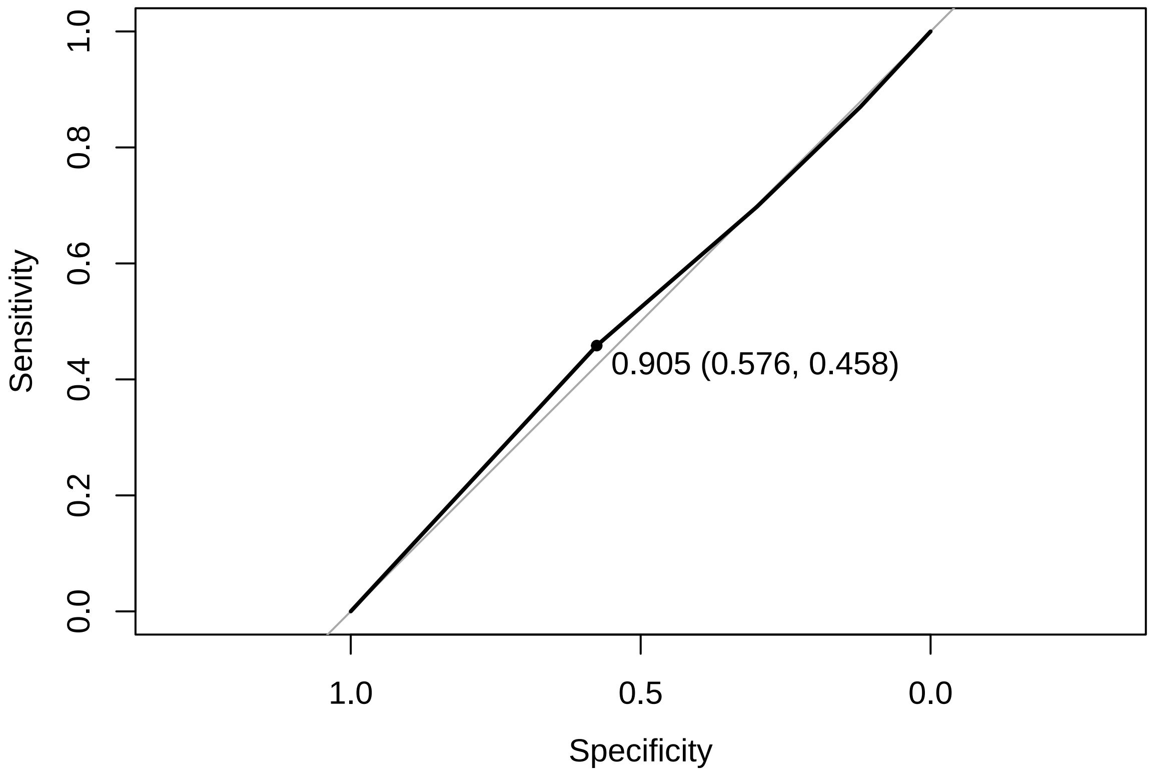
```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
##
## Call:
## roc.default(response = Validation_Data$Personal.Loan, predictor = pred.prob[,     1])
##
## Data: pred.prob[, 1] in 1808 controls (Validation_Data$Personal.Loan 0) < 192 cases (Validation_Data
## Area under the curve: 0.5099
```

```
plot.roc(Validation_Data$Personal.Loan,pred.prob[,1],print.thres="best")
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```



This demonstrates that lowering the sensitivity to 0.498 and raising the specificity to 0.576 by using a threshold of 0.905 could enhance the model.