64060_Assignment 3

Tejaswini Yeruva

2022-10-16

##loading all the packages:

```
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)
```

##loading the UniversalBank.csv file and calling csv and factor variables

```
getwd()
```

```
## [1] "C:/Users/tejar/OneDrive/Desktop/ML Assignments"
```

```
setwd("C:/Users/tejar/OneDrive/Documents")
BankInfo <- read.csv("C:/Users/tejar/Downloads/UniversalBank.csv")
DF_Universal_Bank <- BankInfo %>% select(Age, Experience, Income, Family, CCAvg, Education, Mortgage, P
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

####Creating the Partition

```
selected.var <- c(8,11,12)
set.seed(23)
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]
```

##Then it 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                      773 1127
##            1                       82  114
## 1          0                      315  497
##            1                       39   53
```

```
detach(Train_Data)
```

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

(probability of not using Naive Bayes)

With Online=1 and CC=1, likelihood can calculated with the 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  0.90409357 0.09590643
##    1  0.90813860 0.09186140
## 1 0  0.88983051 0.11016949
##    1  0.90363636 0.09636364
```

##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                     1088 1624
## 1                      121  167
```

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

```
##                CreditCard    0    1
## Personal.Loan
## 0                         1900  812
## 1                          196   92
```

```
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  0.7005900 0.2994100
## 1  0.6805556 0.3194444
```

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

```
##             0         1
##
## 0  0.4011799 0.5988201
## 1  0.4201389 0.5798611
```

##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 0.4011799 0.5988201
##   1 0.4201389 0.5798611
##
##    CreditCard
## Y           0         1
##   0 0.7005900 0.2994100
##   1 0.6805556 0.3194444
```

##While utilizing the two tables created in step C makes it easy to see how you're computing P(LOAN=1|CC=1,Online=1) using the Naive Bayes model, you can also use the pivot table built in step B to rapidly compute P(LOAN=1|CC=1,Online=1) without using the Naive Bayes model.

##The Naive Bayes model predicts the same probability as the previous techniques, although it is lower than the probability calculated by hand in step E. This probability is closer to the one calculated in step B. This could be due to the fact that we are doing the calculations by hand in step E, which leaves space for mistake when rounding fractions, resulting in simply an approximation.

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
##
```

##This model exhibited a low specificity despite being super sensitive. The model anticipated that all values would be zero, but the reference had all true values. Despite missing all 1 data, the model still returns a 90.4 percent accuracy due to the enormous number of 0 values.

##Validation set

```
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

```
library(pROC)
```

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

##
## Attaching package: 'pROC'

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

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

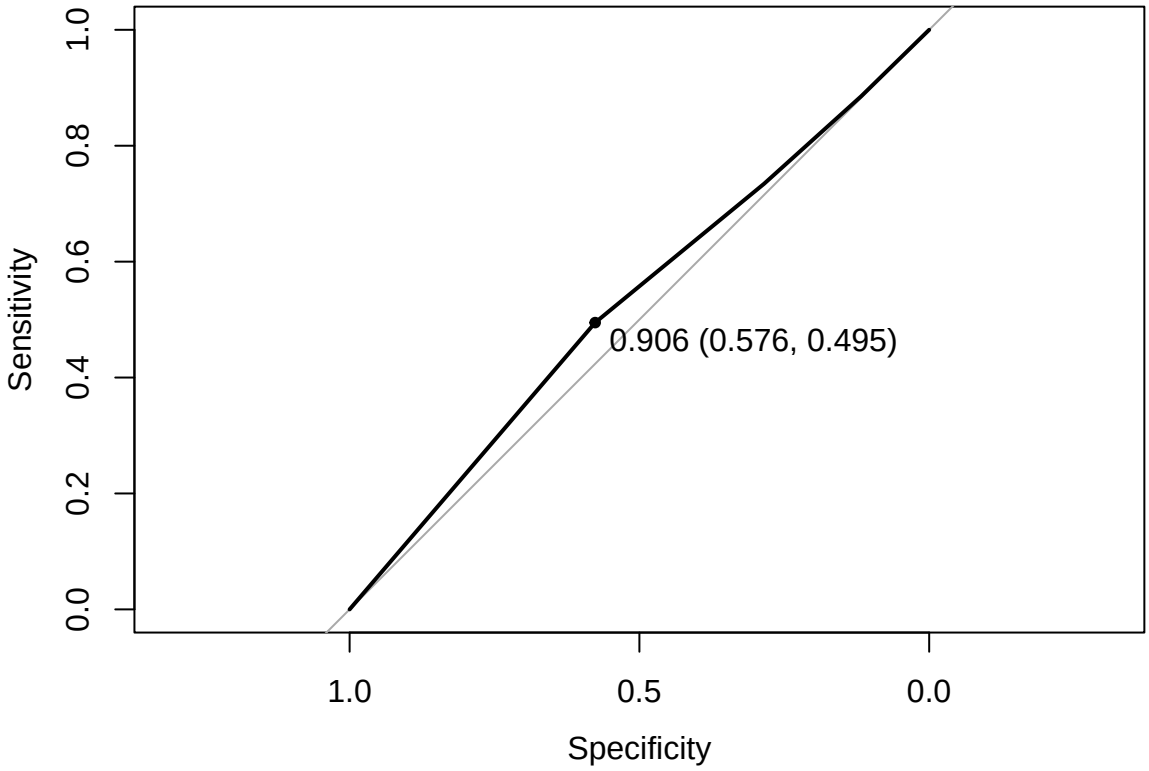
```
## Setting levels: control = 0, case = 1

## 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: 0.5302
```

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

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



##This shows that setting a threshold of 0.906 could improve the model by lowering sensitivity to 0.495 and increasing specificity to 0.576.