FML_Assignment_3

Ravi Gadde

2024-03-11

```
#Calling the libraries caret,dplyr,ggplot2,lattice,knitr,rmarkdown,e1071
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)

#Reading The Data Set UniversalBank
library(readr)
UniB_data <- read.csv("C:/Users/ravi/Downloads/UniversalBank.csv")
View(UniB_data)
```

#In this section, the data is extracted from the CSV file, some fields (such ID and Zip Code) are removed, variable factors are created, and numerical variables are converted to categorical data types.

```
b1 <- UniB_data %>% select(Age, Experience, Income, Family, CCAvg, Education,
Mortgage, Personal.Loan , Securities.Account, CD.Account, Online, CreditCard)
b1$CreditCard <- as.factor(b1$CreditCard)
b1$Personal.Loan <- as.factor((b1$Personal.Loan))
b1$Online <- as.factor(b1$Online)

#Here, Separation of data is done, 60% Training and Test Data 40%
select.variable <- c(8,11,12)
set.seed(23)
```

```
Train.Index_1 = createDataPartition(b1$Personal.Loan, p=0.60, list=0)
Train_UBData = b1[Train.Index_1,select.variable]
Val.Data = b1[-Train.Index_1,select.variable]
```

#A. Create a pivot table for the training data with Online as a column variable, CC as a row variable,and Loan as a secondary row variable. The values inside the table should convey the count. In R use functions melt() and cast(), or function table(). In Python, use panda dataframe methods melt() and pivot().

```
# Demonstrating the pivot table with credit card and Personal LOAN as both
rows,
# and online transaction as a column.
attach(Train_UBData)
ftable(CreditCard,Personal.Loan,Online) #ftable is defined as "function
table"

##                           Online    0    1
## CreditCard Personal.Loan
## 0          0                      773 1127
##            1                       82  114
## 1          0                      315  497
##            1                       39   53

detach(Train_UBData)
```

#B. Consider the task of classifying a customer who owns a bank credit card and is actively using online banking services. Looking at the pivot table, what is the probability that this customer will accept the loan offer? [This is the probability of loan acceptance (Loan = 1) conditional on having a bank credit card (CC = 1) and being an active user of online banking services (Online = 1)].

```
# Methodology: - To determine the conditional probability that Loan=1,
#given Online=1 and CC=1, we add 53 (Loan=1 from ftable) to 497 (Loan=0 from
ftable),
#which is 550. 53/550 = 0.096363 or 9.64% of the time.
prop.table(ftable(Train_UBData$CreditCard,Train_UBData$Online,Train_UBData$Pe
rsonal.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
```

#Question C. Create two separate pivot tables for the training data. #One will have Loan (rows) as a function of Online (columns) #and the other will have Loan (rows) as a function of CC.

```
#Generating 2 pivot tables with 'Online' and 'Credit Card' #as columns in
both and considering 'Personal Loan' as row
```

```
attach(Train_UBData)
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_UBData)
```

#Question D. Compute the following quantities [P(A | B) means “the probability ofA given B”]:

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

##             0          1
##
## 0  0.63333333 0.27066667
## 1  0.06533333 0.03066667

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

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

#Probability of credit card holders among the loan acceptors i.e P(CC = 1 | Loan = 1) i) 92/288 = 0.3194 or 31.94% #Probability of customers with online transactions and are also loan acceptors ii) 167/288 = 0.5798 or 57.986% #The proportion of loan acceptors iii) P(loans= 1) -> 288/3000 = 0.096 or 9.6% ###P(CC = 1 | Loan = 0) iv) 812/2712 = 0.2994 or 29.94% ##P(Online = 1 | Loan = 0) V) 1624/2712 = 0.5988 or 59.88% #P(Loan = 0) Vi) total loans=0 from table(2712) divided by total from table (3000) = 0.904 or 90.4% #Question E. Use the quantities computed above to compute the naive Bayes probability P(Loan = 1 | CC = 1,Online = 1). (0.3194 x 0.5798 x 0.096)/[(0.3194 x 0.5798 x 0.096)+(0.2994 x 0.5988 x 0.904)] = 0.0988505642823 or 9.885% #Question F. Compare this value with the one obtained from the pivot table in (B). #Which is a more accurate estimate? #There is little variance between 0.096363, or 9.64%, and 0.0988505642823, or 9.885%. The pivot Table value is the more accurate estimated value since it does not depend on the probabilities being independent. While E determines the likelihood of each count, B makes a simple computation based on a count. As a result, B is more specific whereas E is broader. #Question G. Which of the entries in this table are needed for computing P(Loan = 1 | CC = 1, Online = 1)? #Run naive Bayes on the data. Examine the model output on training data, #and find the entry that corresponds to P(Loan = 1 | CC = 1, Online = 1). #Compare this to the number you obtained in (E).

```
# Displaying the training dataset
UniB_data.sb <- naiveBayes(Personal.Loan ~ ., data = Train_UBData)
UniB_data.sb

##
## 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 the pivot table in step B can be used to quickly compute P(LOAN=1|CC=1,Online=1) without utilizing the Naive Bayes model, the two tables constructed in step C make it straightforward and apparent how we are computing P(LOAN=1|CC=1,Online=1) using the Naive Bayes model.

#But compared to the manually calculated probability in step E, the model forecast is less accurate. The Naive Bayes model predicts probability in the same way as earlier techniques. The likelihood that is anticipated is more like the one from step B. This is made possible by the fact that step E necessitates human computation, which raises the risk of error when rounding fractions and producing an estimate.

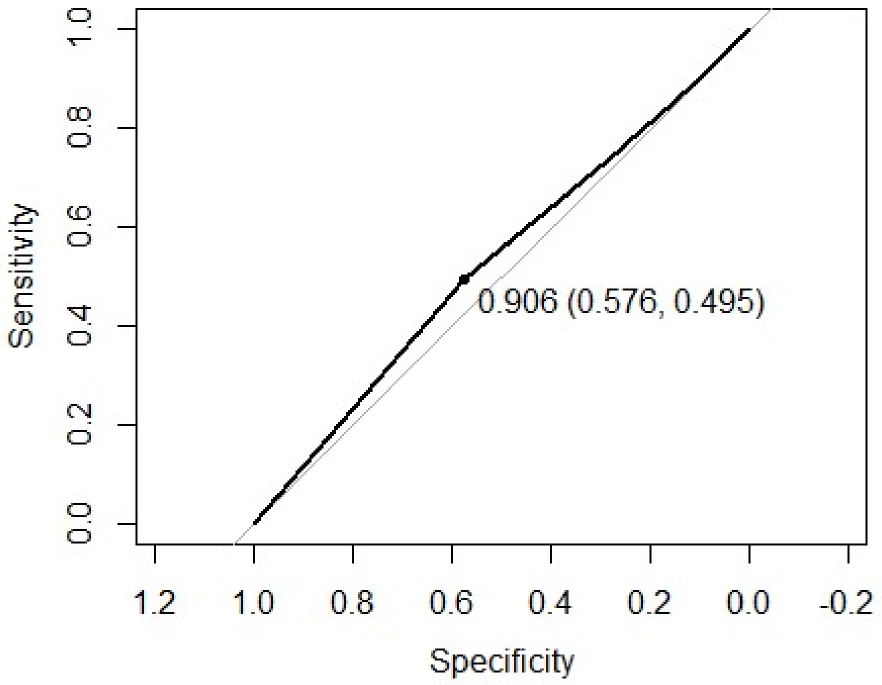
```
#Generating confusion matrix for Training Data
prediction_class <- predict(UniB_data.sb, newdata = Train_UBData)
confusionMatrix(prediction_class, Train_UBData$Personal.Loan)

## 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’s great sensitivity was counterbalanced by its low precision. The model projected that all values would be 0 in the absence of all real values from the reference. Because there are so many 0s in the data, the model would still yield an accuracy of 90.4% even if it missed every value of 1.

```
prediction.probab <- predict(UniB_data.sb, newdata=Val.Data, type="raw")
prediction_class <- predict(UniB_data.sb, newdata = Val.Data)
confusionMatrix(prediction_class, Val.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
##
```

#Visualizing the model graphically and comparing the best results.

```
library(pROC)

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

##
## Attaching package: 'pROC'

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

roc(Val.Data$Personal.Loan,prediction.probab[,1])

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

## Setting direction: controls < cases

##
## Call:
## roc.default(response = Val.Data$Personal.Loan, predictor =
prediction.probab[,     1])
##
## Data: prediction.probab[, 1] in 1808 controls (Val.Data$Personal.Loan 0) <
192 cases (Val.Data$Personal.Loan 1).
## Area under the curve: 0.5302

plot.roc(Val.Data$Personal.Loan,prediction.probab[,1],print.thres="best")

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

## Setting direction: controls < cases
```



0.906 (0.576, 0.495)
Sensitivity
Specificity
0.0
0.2
0.4
0.6
0.8
1.0
1.2
-0.2