Assignment_3

vivek rao kathheragandla

2024-02-26

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

```
## Warning: package 'e1071' was built under R version 4.3.3
```

```
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(caret)
```

```
## Warning: package 'caret' was built under R version 4.3.3
```

```
#Reading The Data Set Universal Bank
library(readr)
UniB_data <- read.csv("UniversalBank.csv")
View(UniB_data)
# The t function creates a transpose of the data frame
t(t(names(UniB_data)))
```

```
##       [,1]
##  [1,] "ID"
##  [2,] "Age"
##  [3,] "Experience"
##  [4,] "Income"
##  [5,] "ZIP.Code"
##  [6,] "Family"
##  [7,] "CCAvg"
##  [8,] "Education"
##  [9,] "Mortgage"
## [10,] "Personal.Loan"
## [11,] "Securities.Account"
## [12,] "CD.Account"
## [13,] "Online"
## [14,] "CreditCard"
```

#Here in this chunk we are extracting the data from the csv file and removing certain fields such as ID and Zip Code and creating variable factors along with changing the numerical variables into categorical data type variables.

```
b1 <- UniB_data %>% select(Age, Experience, Income, Family, CCAvg, Education, Mortgage, Perso
nal.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=FALSE)
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 is defined as "function table".
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
```

#Demonstrating the pivot table with credit card and Personal LOAN as both rows, #and online transaction as a column.

```
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$Personal.Loan),mar
gin=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
```

#Generating the pivot table that depicts the probability of #getting a loan depending on Credit cards and online transaction.

#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? Between 0.096363, or 9.64%, and 0.0988505642823, or 9.885%, there is no noticeable difference. As the pivot table value does not rely on the probabilities being independent, it is the more accurate estimated value. B uses a straight calculation from a count, whereas E evaluates the probability of each of those counts. B is therefore more particular while E is more general.

#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
```

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

#However, the model prediction is less accurate than the manually determined probability in step E. The Naive Bayes model makes the same probability prediction as the methods used before it. The predicted probability is more similar to that from step B. This is possible because step E requires human computation, increasing the possibility of error while rounding fractions and leading to an approximation.

```
#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 low precision balanced its high sensitivity. In the absence of all real values from the reference,the model predicted that all values would be 0. Even if the model missed all values of 1, it would still produce a 90.4% accuracy because to the large amount of 0s.

```
prediction.probability<- 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 comaring the best result

```
library(pROC)
```

```
## Warning: package 'pROC' was built under R version 4.3.2
```

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

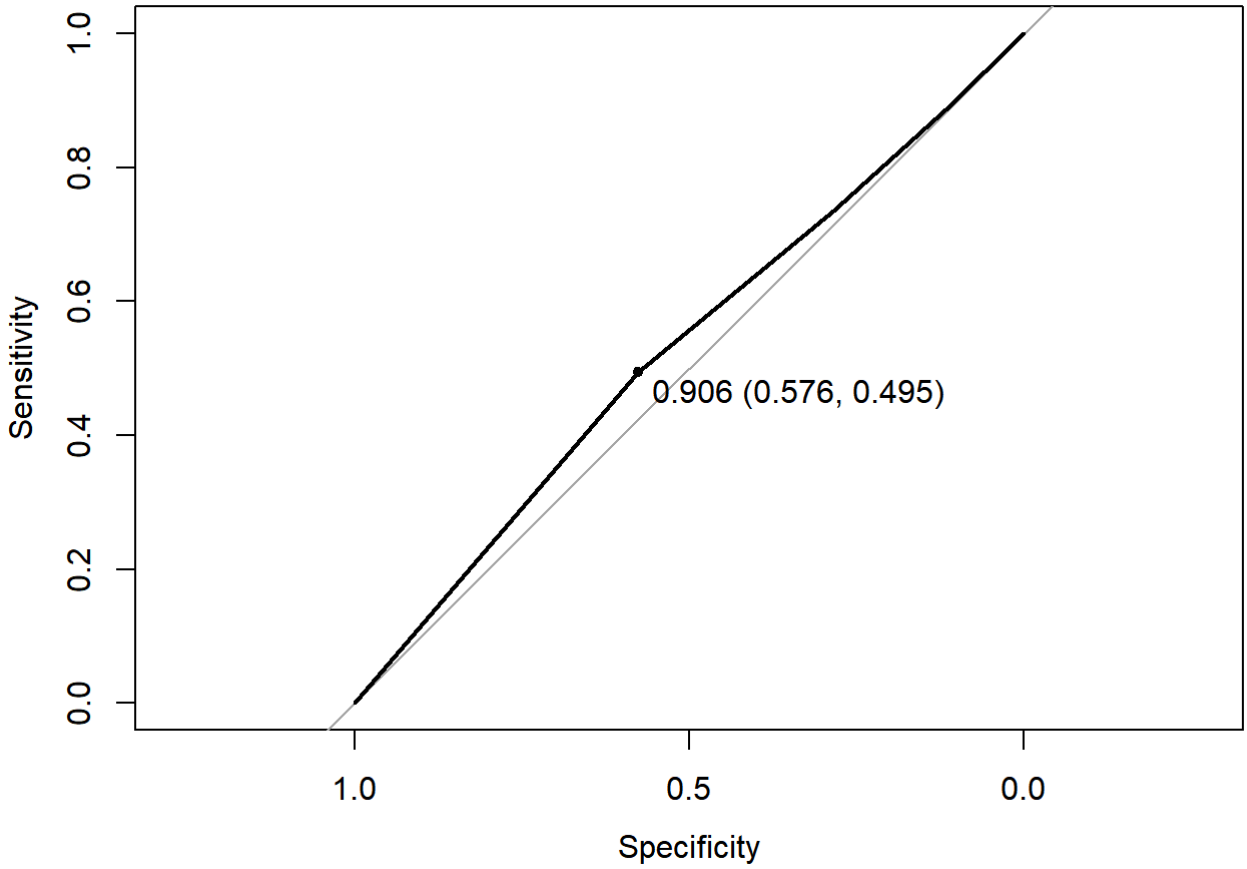
```
##
## Attaching package: 'pROC'
```

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

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

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

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


Sensitivity
0.0
0.2
0.4
0.6
0.8
1.0
0.906 (0.576, 0.495)
1.0
0.5
0.0
Specificity