FML_Assignment3

Thanasit C.

2024-02-27

1. Summary

1). All Answers are in the Section 5.
2). I found that the 'naiveBayes' function performed similar to the Bayes Probability Theory.According to the question, both of the probability calculated by the package and probability function is equal to 0.0914. However, it is different if we calculate from a trained dataset via pivot table, the probability is 0.0879, because of the effects of Bayes' assumption about the dependency of the variables. But for this particular 'set.seed', the accuracy of the 'naiveBayes' function is more closer to the prior probability, 0.096, than the probability from the table.

2. Libraries

```
library(dplyr)
library(caret)
require(e1071)
library(gmodels)
library(ggplot2)
library(ggpubr)
library(reshape2)
library(tables)
```

3. Import data

3.1. Set working directory
3.2. Import csv file as dataframe format
3.3. Check data structure

```
## set working directory
setwd("/Users/sieng/Documents/Study/MS.Business Analytics/SPRING
2024/Fundamental of Machine Learning/Assignment/Assignment 3")

## import data
maindf <- read.csv("UniversalBank.csv") %>% as.data.frame()

## check data structure
str(maindf)

## 'data.frame':    5000 obs. of  14 variables:
##  $ ID                 : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Age                : int  25 45 39 35 35 37 53 50 35 34 ...
##  $ Experience         : int  1 19 15 9 8 13 27 24 10 9 ...
```

```
##  $ Income           : int  49 34 11 100 45 29 72 22 81 180 ...
##  $ ZIP.Code         : int  91107 90089 94720 94112 91330 92121 91711
93943 90089 93023 ...
##  $ Family           : int  4 3 1 1 4 4 2 1 3 1 ...
##  $ CCAvg            : num  1.6 1.5 1 2.7 1 0.4 1.5 0.3 0.6 8.9 ...
##  $ Education        : int  1 1 1 2 2 2 2 3 2 3 ...
##  $ Mortgage         : int  0 0 0 0 0 155 0 0 104 0 ...
##  $ Personal.Loan    : int  0 0 0 0 0 0 0 0 0 1 ...
##  $ Securities.Account: int  1 1 0 0 0 0 0 0 0 0 ...
##  $ CD.Account       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Online           : int  0 0 0 0 0 1 1 0 1 0 ...
##  $ CreditCard       : int  0 0 0 0 1 0 0 1 0 0 ...
```

4. Data wrangling and prepairation

4.1. handle missing value

```
## 1) Find N/A value
sumna <- sum(is.na(maindf))
print("Number of N/A values in data set")
sumna

colsumna <- colSums(is.na(maindf))
print("Number of N/A by column")
colsumna

## [1] "Number of N/A values in data set"
## [1] 0
## [1] "Number of N/A by column"
##                 ID                Age         Experience
Income
##                  0                  0                  0
0
##           ZIP.Code             Family              CCAvg
Education
##                  0                  0                  0
0
##           Mortgage      Personal.Loan Securities.Account
CD.Account
##                  0                  0                  0
0
##             Online         CreditCard
##                  0                  0
```

4.2. Rearrange and remove irrelavent column
Select only 'Personal.Loan', 'CreditCard', and 'Online' for the model testing.

```
### 4.2 Rearrange and remove unrelated column ##############
maindf2 <- maindf %>% select(Personal.Loan, CreditCard, Online)
```

4.3. Reassign data attribute

```r
### 4.4 correcting data attributes ###############
# number()/integer()

# factor()
maindf2$Personal.Loan <- factor(maindf2$Personal.Loan)
maindf2$Online <- factor(maindf2$Online)
maindf2$CreditCard <- factor(maindf2$CreditCard)

str(maindf2)

## 'data.frame':    5000 obs. of  3 variables:
##  $ Personal.Loan: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 2 ...
##  $ CreditCard   : Factor w/ 2 levels "0","1": 1 1 1 1 2 1 1 2 1 1 ...
##  $ Online       : Factor w/ 2 levels "0","1": 1 1 1 1 1 2 2 1 2 1 ...
```

4.4. Split Data into 60% training and 40% validation

```r
set.seed(22)
### Split data into 60% train and 40% validation
q1_trainsplit = 0.6

q1_trainsplit_index <- caret::createDataPartition(y = maindf2$Personal.Loan,
p = q1_trainsplit, list = FALSE)
q1_train <- maindf2[q1_trainsplit_index,]
q1_validation <- maindf2[-q1_trainsplit_index,]
```

5. Modelling and problem solving

5.1 Question_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.

```r
### Create pivot table
ftable(xtabs(~ CreditCard + Personal.Loan + Online, data = q1_train))

##                          Online    0    1
## CreditCard Personal.Loan                 
## 0          0                     770 1150
##            1                      86  118
## 1          0                     304  488
##            1                      37   47
```

5.2 Question_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)].

```r
OnlineAndCC <- q1_train %>%
                  filter(CreditCard == 1, Online == 1)
```

```r
PersonalLoan <- OnlineAndCC %>%
                  filter(Personal.Loan == 1)

print("P(Loan = 1 | CC = 1 and Online = 1) is")
round(nrow(PersonalLoan)/nrow(OnlineAndCC), digits = 4)

## [1] "P(Loan = 1 | CC = 1 and Online = 1) is"
## [1] 0.0879
```

5.3 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.

```r
table(q1_train$Personal.Loan, q1_train$CreditCard)

##
##        0    1
##   0 1920  792
##   1  204   84

table(q1_train$Personal.Loan, q1_train$Online)

##
##        0    1
##   0 1074 1638
##   1  123  165
```

5.4. Question_D

Compute the following quantities [P(A | B) means "the probability of A given B"]:

```r
# Create Proportion table for Personal.Loan and Credit Card
propCCByLoan <- prop.table(table(q1_train$Personal.Loan,
q1_train$CreditCard), margin = 1)
print("Pivot table for Personal.Loan and Credit Card")
propCCByLoan

## [1] "Pivot table for Personal.Loan and Credit Card"
##
##             0         1
##   0 0.7079646 0.2920354
##   1 0.7083333 0.2916667

# Create Proportion table for Personal.Loan and Online
propOnlineByLoan <- prop.table(table(q1_train$Personal.Loan,
q1_train$Online), margin = 1)
print("Pivot table for Personal.Loan and Online")
propOnlineByLoan

## [1] "Pivot table for Personal.Loan and Online"
##
##             0         1
```

```
##   0 0.3960177 0.6039823
##   1 0.4270833 0.5729167
```

```r
### 5.4.1. P(CC = 1 | Loan = 1)
print(paste("P(CC = 1 | Loan = 1) =", round(propCCByLoan[2,2], digits = 4)))
```

```
## [1] "P(CC = 1 | Loan = 1) = 0.2917"
```

```r
### 5.4.2. P(Online = 1 | Loan = 1)
print(paste("P(Online = 1 | Loan = 1) =", round(propOnlineByLoan[2,2], digits
= 4)))
```

```
## [1] "P(Online = 1 | Loan = 1) = 0.5729"
```

```r
### 5.4.3. P(Loan = 1)
acceptloandf <- q1_train %>%
                  filter(Personal.Loan == 1)
probacceptloan <- round(nrow(acceptloandf)/nrow(q1_train), digits = 4)
print(paste("P(Loan = 1) =", probacceptloan))
```

```
## [1] "P(Loan = 1) = 0.096"
```

```r
### 5.4.4. P(CC = 1 | Loan = 0)
print(paste("P(CC = 1 | Loan = 0) =", round(propCCByLoan[1,2], digits = 4)))
```

```
## [1] "P(CC = 1 | Loan = 0) = 0.292"
```

```r
### 5.4.5. P(Online = 1 | Loan = 0)
print(paste("P(Online = 1 | Loan = 0) =", round(propOnlineByLoan[1,2], digits
= 4)))
```

```
## [1] "P(Online = 1 | Loan = 0) = 0.604"
```

```r
### 5.4.6. P(Loan = 0)
declineloandf <- q1_train %>%
                   filter(Personal.Loan == 0)
probdeclineloan <- round(nrow(declineloandf)/nrow(q1_train), digits = 4)
print(paste("P(Loan = 0) =", probdeclineloan))
```

```
## [1] "P(Loan = 0) = 0.904"
```

5.5. Question_E

Use the quantities computed above to compute the naive Bayes probability P(Loan = 1 | CC = 1, Online = 1).

```r
# P(Loan = 1 | CC = 1, Online = 1) = [P(CC = 1 | Loan = 1) * P(Online = 1 |
Loan = 1) * P(Loan = 1)] / [(P(CC = 1 | Loan = 1) * P(Online = 1 | Loan = 1)
* P(Loan = 1)) + (P(CC = 1 | Loan = 0) * P(Online = 1 | Loan = 0) * P(Loan =
0))]
q5_numerator <- propCCByLoan[2,2]*propOnlineByLoan[2,2]*probacceptloan
q5_denormurator <- (propCCByLoan[2,2]*propOnlineByLoan[2,2]*probacceptloan) +
(propCCByLoan[1,2]*propOnlineByLoan[1,2]*probdeclineloan)
```

```r
q5_nbProb <- round(q5_numerator/q5_denormurator, digits = 4)
```

```
## [1] "Answer_E;"

## [1] "P(Loan = 1 | CC = 1, Online = 1) = 0.0914"
```

5.6. Question_F
Compare this value with the one obtained from the pivot table in (B). Which is a more accurate estimate?

```
## [1] "Answer_F"

## [1] "The probability I got from the Question_E is equal to 0.0914 which is
a tiny bit difference from what is calculated from Question_B's pivot table
of 0.0879. In this case the calculation from Bayes probability is a bit
better because the result is closed to the prior probility of 0.096."
```

5.7. 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 Question_E.

```r
# Create test dataset
q7_test <- data.frame(Online = 1, CreditCard = 1)
q7_test$Online <- factor(q7_test$Online)
q7_test$CreditCard <- factor(q7_test$CreditCard)

# Create Model
q7_nb_model <- naiveBayes(Personal.Loan ~ Online + CreditCard, data =
q1_train)
#q7_nb_model

# training model
Predicted_train <- predict(q7_nb_model,newdata = q1_train, type = "raw")

# Validate Model
Predict_validation <- predict(q7_nb_model, newdata = q1_validation, type =
"raw")

#test Model
Predicted_test <- predict(q7_nb_model, newdata = q7_test, type = "raw")
Predicted_test

print("Answer_G")
print("By using 'naiveBayes' function the probability of customer accepting
the loan given they have Credit Card and have an online Banking is equal to
0.0914. This value is exactly the same as the value calculated by probability
function in Question_E.")
```

```
##              0          1
## [1,] 0.9085908 0.09140916
## [1] "Answer_G"
## [1] "By using 'naiveBayes' function the probability of customer accepting
the loan given they have Credit Card and have an online Banking is equal to
0.0914. This value is exactly the same as the value calculated by probability
function in Question_E."
```