Assignment_3

Tarun

2024-03-07

Description:“The purpose of this Assignment is to use Naive Bayes for classification”.

```
knitr::opts_chunk$set(echo = TRUE)
```

#Load the required libraries

```
library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

library(e1071)
library(class)
library(tidyverse)

## Warning: package 'tidyverse' was built under R version 4.3.3

## ── Attaching core tidyverse packages ──────────────────────── tidyverse
2.0.0 ──
## ✔ dplyr     1.1.4     ✔ readr     2.1.5
## ✔ forcats   1.0.0     ✔ stringr   1.5.1
## ✔ lubridate 1.9.3     ✔ tibble    3.2.1
## ✔ purrr     1.0.2     ✔ tidyr     1.3.1

## ── Conflicts ──────────────────────────────────────────
tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ✖ purrr::lift()   masks caret::lift()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all
conflicts to become errors
```

Get present working directory

```
getwd()

## [1] "C:/Users/tarun/OneDrive/Desktop"

setwd("C:\\Users\\tarun\\Downloads")
customer_data <- read.csv("UniversalBank.csv")
str(customer_data)
```

```
## '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 ...
```

Converting relevant columns to factor variables and checking their factor status.

Checking the structure of the dataset after conversion.

```
customer_data$Online <- as.factor(customer_data$Online)
is.factor(customer_data$Online)

## [1] TRUE

customer_data$CreditCard <- as.factor(customer_data$CreditCard)
is.factor(customer_data$CreditCard)

## [1] TRUE

customer_data$Personal.Loan <- as.factor(customer_data$Personal.Loan)
is.factor(customer_data$Personal.Loan)

## [1] TRUE

str(customer_data)

## '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     : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 2 ...
##  $ 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            : Factor w/ 2 levels "0","1": 1 1 1 1 1 2 2 1 2 1 ...
##  $ CreditCard        : Factor w/ 2 levels "0","1": 1 1 1 1 2 1 1 2 1 1 ...
```

#Data partition into Training and Validation.

```
set.seed(15)
customer_data1 <- createDataPartition(customer_data$Personal.Loan, p=0.60,
list = FALSE)
train_customer <- customer_data[customer_data1,]
validate_customer <- customer_data[-customer_data1,]
```

Data Normalization.

```
norm_data <- preProcess(train_customer[,-c(10,13,14)], method = c("center",
"scale"))
predict_tdata <- predict(norm_data, train_customer)
predict_vdata <- predict(norm_data, validate_customer)
summary(predict_tdata)
```

```
##        ID                 Age            Experience            Income
##  Min.   :-1.718966   Min.   :-1.9325   Min.   :-1.997167   Min.   :-1.4435
##  1st Qu.:-0.875360   1st Qu.:-0.8857   1st Qu.:-0.864443   1st Qu.:-0.7619
##  Median :-0.004208   Median :-0.0134   Median : 0.006883   Median :-0.2341
##  Mean   : 0.000000   Mean   : 0.0000   Mean   : 0.000000   Mean   : 0.0000
##  3rd Qu.: 0.862061   3rd Qu.: 0.8589   3rd Qu.: 0.878210   3rd Qu.: 0.5355
##  Max.   : 1.766336   Max.   : 1.9057   Max.   : 2.010934   Max.   : 3.3061
##     ZIP.Code             Family             CCAvg            Education
##  Min.   :-35.9506   Min.   :-1.2237   Min.   :-1.1014   Min.   :-1.0529
##  1st Qu.: -0.5244   1st Qu.:-1.2237   1st Qu.:-0.7024   1st Qu.:-1.0529
##  Median :  0.1846   Median :-0.3482   Median :-0.2465   Median : 0.1436
##  Mean   :  0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.:  0.6359   3rd Qu.: 0.5273   3rd Qu.: 0.3234   3rd Qu.: 1.3401
##  Max.   :  1.5125   Max.   : 1.4028   Max.   : 4.5978   Max.   : 1.3401
##     Mortgage        Personal.Loan Securities.Account   CD.Account
Online
##  Min.   :-0.5591   0:2712        Min.   :-0.3388    Min.   :-0.2404
0:1238
##  1st Qu.:-0.5591   1: 288        1st Qu.:-0.3388    1st Qu.:-0.2404
1:1762
##  Median :-0.5591                 Median :-0.3388    Median :-0.2404
##  Mean   : 0.0000                 Mean   : 0.0000    Mean   : 0.0000
##  3rd Qu.: 0.4322                 3rd Qu.:-0.3388    3rd Qu.:-0.2404
##  Max.   : 5.6581                 Max.   : 2.9506    Max.   : 4.1578
##  CreditCard
##  0:2128
##  1: 872
```

```
##
##
##
##
```

#A. Creating Pivot Table with Online as column variable and CC, Personal.Loan as row variables.

```
pivot_customer<- ftable(predict_tdata$Personal.Loan, predict_tdata$Online,
predict_tdata$CreditCard, dnn=c('Personal.loan','CreditCard', 'Online'))
pivot_customer

##                           Online    0    1
## Personal.loan CreditCard
## 0             0                   791  330
##               1                  1130  461
## 1             0                    82   35
##               1                   125   46
```

#B.Probability of Loan Acceptance (Loan=1) conditional on CC=1 and Online=1.

```
prob_customer<-pivot_customer[4,2]/(pivot_customer[2,2]+pivot_customer[4,2])
prob_customer

## [1] 0.09072978
```

#C. Probability for personal loan and Online.

```
pivot_customer1 <- ftable(predict_tdata$Personal.Loan, predict_tdata$Online,
dnn=c('PersonalLoan','Online'))
pivot_customer1

##              Online    0    1
## PersonalLoan
## 0                   1121 1591
## 1                    117  171
```

#C. Probability for personal loan and CreditCard.

```
pivot_customer2 <- ftable(predict_tdata$Personal.Loan,
predict_tdata$CreditCard, dnn=c('PersonalLoan','CreditCard'))
pivot_customer2

##              CreditCard    0    1
## PersonalLoan
## 0                       1921  791
## 1                        207   81
```

#D.Computation of Quantities(i).P(CC=1 | Loan= 1)

```
prob_customer1 <- pivot_customer2[2,2] / (pivot_customer2[2,2] +
pivot_customer2[2,1])
prob_customer1
```

```
## [1] 0.28125
```

#D.Computation of Quantities(ii).P(Online=1 | Loan=1)

```
prob_customer2 <- pivot_customer1[2,2] / (pivot_customer1[2,2] +
pivot_customer1[2,1])
prob_customer2

## [1] 0.59375
```

#D.Computation of Quantities(iii).P(Loan=1)

```
prob_customer3 <- ftable(predict_tdata$Personal.Loan)
prob_customer3

##     0    1
##
##  2712  288

prob_customer_3 <- prob_customer3[1,2] / (prob_customer3[1,2] +
prob_customer3[1,1])
prob_customer_3

## [1] 0.096
```

#D.Computation of Quantities(iv).P(CC=1 | Loan=0)

```
prob_customer4 <- pivot_customer2[1,2] / (pivot_customer2[1,2] +
pivot_customer2[1,1])
prob_customer4

## [1] 0.2916667
```

#D.Computation of Quantities(v).P(Online=1 | Loan=0)

```
prob_customer5 <- pivot_customer1[1,2] / (pivot_customer1[1,2] +
pivot_customer1[1,1])
prob_customer5

## [1] 0.5866519
```

#D.Computation of Quantities(vi).P(Loan=0)

```
prob_customer6 <- ftable(predict_tdata$Personal.Loan)
prob_customer6

##     0    1
##
##  2712  288

prob_customer_6 <- prob_customer6[1,1] / (prob_customer6[1,1] +
prob_customer6[1,2])
prob_customer_6
```

```
## [1] 0.904
```

#E. Computing Naive Bayes using conditional probabilities derived from D.

```
nb_customer <- (prob_customer1 * prob_customer2 * prob_customer_3) /
(prob_customer1 * prob_customer2 * prob_customer_3 + prob_customer4 *
prob_customer5 * prob_customer_6)
nb_customer

## [1] 0.09390827
```

#F. Compare the values of answers from B. and E. Compare this value to that got from the pivot table in (B). Which is the more accurate estimate? The probabilities calculated using the Bayes probability, i.e., B, is 0.09072978, while the probability obtained from Naive Bayes is 0.09390827. The comparison of Bayes with Naive Bayes shows that Naive Bayes is more probable.

#G. Using Naive Bayes directly applied to the data.

```
nb_model <- naiveBayes(Personal.Loan ~ Online + CreditCard, data =
predict_tdata)
nb_model

##
## 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 0.4133481 0.5866519
##   1 0.4062500 0.5937500
##
##    CreditCard
## Y           0         1
##   0 0.7083333 0.2916667
##   1 0.7187500 0.2812500
```

While utilizing the two tables generated in step C provides a clear and direct method for understanding how the Naive Bayes model computes P(LOAN=1|CC=1,Online=1), the pivot table in step B offers a quick approach to calculate P(LOAN=1|CC=1,Online=1) without relying on the Naive Bayes model. However, the prediction made by the model is less likely than the probability manually determined in step E. Despite this, the Naive Bayes model produces the same probability predictions as the earlier techniques. The estimated probability is more likely than the one obtained from step B. This discrepancy could be attributed to the manual calculation involved in step E, which introduces the possibility of errors when rounding fractions and provides only an approximation. # RD confusion matrix about Train_Data # Training

```
predicting.class <- predict(nb_model, newdata = train_customer)
confusion_matrix_train <- confusionMatrix(predicting.class,
train_customer$Personal.Loan)
confusion_matrix_train
```

```
## 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
##
```

RD confusion matrix about Validation_Data

Validation

```
predicting.class <- predict(nb_model, newdata = validate_customer)
confusion_matrix_validation <- confusionMatrix(predicting.class,
validate_customer$Personal.Loan)
confusion_matrix_validation
```

```
## 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
##
```

Validation set

```r
predicting.prob <- predict(nb_model, newdata = validate_customer, type =
"raw")
predicting.class <- predict(nb_model, newdata = validate_customer)
confusion_matrix_validation <- confusionMatrix(predicting.class,
validate_customer$Personal.Loan)
confusion_matrix_validation
```

```
## 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
```

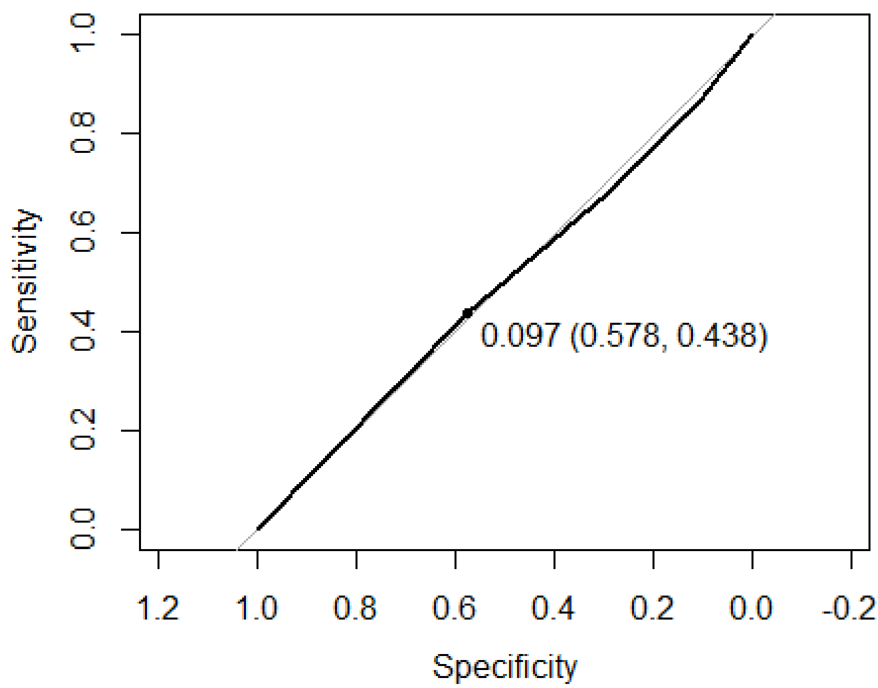
```
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 1.000
##             Specificity : 0.000
##          Pos Pred Value : 0.904
##          Neg Pred Value :   NaN
##              Prevalence : 0.904
##          Detection Rate : 0.904
##    Detection Prevalence : 1.000
##       Balanced Accuracy : 0.500
##
##        'Positive' Class : 0
##
```

Assessing the model's performance using Receiver Operating Characteristic (ROC) analysis.

Plotting the ROC curve with the optimal threshold marked.

```
library(pROC)

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

##
## Attaching package: 'pROC'

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

roc(validate_customer$Personal.Loan, predicting.prob[,2])

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

## Setting direction: controls < cases

##
## Call:
## roc.default(response = validate_customer$Personal.Loan, predictor =
predicting.prob[,     2])
##
## Data: predicting.prob[, 2] in 1808 controls
(validate_customer$Personal.Loan 0) < 192 cases
(validate_customer$Personal.Loan 1).
## Area under the curve: 0.4953

plot.roc(validate_customer$Personal.Loan, predicting.prob[,2], print.thres =
"best")
```

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



While the Naive Bayes model is good, at predicting probabilities it seems that setting a value of 0.906 could improve how sensitive and specific the predictions are. Changing this value to 0.906 would lower sensitivity to 0.495. Increase the accuracy to 0.576. This change might make the model work better and improve its accuracy when making loan approvals.