Assignment_3

Meenakshi Vaidhiyanathan

2024-03-08

Importing required packages including `naivebayes` package

```
#install.packages("reshape")
#install.packages("reshape2")
#install.packages("melt")
#install.packages("naivebayes")
#install.packages("pROC")
```

loading the necessary libraries

```
library(readr)
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)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```
library(class)
library(melt)
library(reshape)
```

```
##
## Attaching package: 'reshape'

## The following object is masked from 'package:class':
##
##     condense
```

```
## The following object is masked from 'package:dplyr':
##
##     rename
```

```r
library(reshape2)
```

```
##
## Attaching package: 'reshape2'

## The following objects are masked from 'package:reshape':
##
##     colsplit, melt, recast
```

```r
library(ggplot2)
library(ISLR)
library(naivebayes)
```

```
## naivebayes 0.9.7 loaded
```

```r
library(e1071)
library(pROC)
```

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

##
## Attaching package: 'pROC'

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

Importing the dataset and assigning it to the variable `bank`. Using the head function to display first six rows of the dataset `bank`.

```r
bank <- read_csv("~/Downloads/UniversalBank.csv")
```

```
## Rows: 5000 Columns: 14
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (14): ID, Age, Experience, Income, ZIP Code, Family, CCAvg, Education, M...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
head(bank)
```

```
## # A tibble: 6 x 14
##      ID   Age Experience Income `ZIP Code` Family CCAvg Education Mortgage
##   <dbl> <dbl>      <dbl>  <dbl>      <dbl>  <dbl> <dbl>     <dbl>    <dbl>
```

```
## 1      1    25           1     49      91107       4   1.6          1         0
## 2      2    45          19     34      90089       3   1.5          1         0
## 3      3    39          15     11      94720       1   1            1         0
## 4      4    35           9    100      94112       1   2.7          2         0
## 5      5    35           8     45      91330       4   1            2         0
## 6      6    37          13     29      92121       4   0.4          2       155
## # i 5 more variables: 'Personal Loan' <dbl>, 'Securities Account' <dbl>,
## #   'CD Account' <dbl>, Online <dbl>, CreditCard <dbl>
```

The is.na() function is used to check if there are any missing values in the dataset and is assigned to `bank_na` variable.

```
bank_na <- is.na.data.frame("universalbank")
```

The column Online, Credit card and personal loan have categorical variables which are converted to factors using the `as.factor()` function.

```
bank$Online<- as.factor(bank$Online)
bank$CreditCard<- as.factor(bank$CreditCard)
bank$`Personal Loan`<- as.factor(bank$`Personal Loan`)
```

#Data Partition and Normalization The process of Data partition is performed to create the training data that is asssigned to `Train_data` containing 60% of the dataset `bank` and the Validation data assigned to `Valid_data` containing the remaining 40% of the dataset `bank`.

```
set.seed(333)
Train_Index<- createDataPartition(bank$`Personal Loan`, p=0.6, list=FALSE)
Train_data <-bank[Train_Index,]
Valid_data <-bank[-Train_Index,]
```

Both Training and validation data is normalised using the fucntions `preProcess()` and `predict()` and assinged to `Train_normalised` and `Valid_normalised` respectively.

```
Model_normalised <- preProcess(Train_data[,-c(10,13:14)],method = c("center", "scale"))
Train_normalised <- predict(Model_normalised,Train_data)
Valid_normalised<- predict(Model_normalised,Valid_data)
```

The following code chink creates a pivot table for training data using the `table()` function.

```
Table.OCP <- table(Train_normalised$`Personal Loan`, Train_normalised$Online, Train_normalised$CreditCa
Table.OCP
```

```
## , , Credit Card = 0
##
##               Online
## Personal Loan    0    1
##             0  786 1120
##             1   73  135
##
## , , Credit Card = 1
##
##               Online
```

```
## Personal Loan    0    1
##              0  305  501
##              1   31   49
```

Part B: Computing P(Loan | Online & CC) As we look that the pivot table created in part A out of the total 550 records where of active online banking users with credit cards, 49 had accepted a personal loan, so

$$\mathbf{P}(\text{Loan} = 1 \mid \text{CC} = 1\text{and Online} = 1) = \frac{49}{550} = 0.089$$

.

Computing P(loan | Online & CC)

```r
Table.OCP[2,2,2] / (Table.OCP[2,2,2] + Table.OCP[1,2,2])
```

```
## [1] 0.08909091
```

Part C: Creating two separate pivot tables for Training data.One will have Loan as rows which is as a function of Online i.e columns and the other will have Loan (rows) as a function of CC. The 'table() function is used.

```r
online_table <- table(Train_normalised$`Personal Loan`, Train_normalised$Online, dnn=c("Personal Loan",
online_table
```

```
##              Online
## Personal Loan    0    1
##              0 1091 1621
##              1  104  184
```

```r
credit_table <- table(Train_normalised$`Personal Loan`, Train_data$CreditCard, dnn=c("Personal Loan", "
credit_table
```

```
##              Credit Card
## Personal Loan    0    1
##              0 1906  806
##              1  208   80
```

Part D : Computing the following quantities: i)P(CC = 1 | Loan = 1) (the proportion of credit card holders among the loan acceptors) i)

$\mathbf{P}$(CC = 1 | Loan = 1)= 80/80+208

```r
probability_ccl <- credit_table[2,2] / (credit_table[2,2] + credit_table[2,1])
probability_ccl
```

```
## [1] 0.2777778
```

ii) P(Online = 1 | Loan = 1) ii)

$\mathbf{P}$(Online = 1 | Loan = 1)= 184/184+104

```
probability_ol <- online_table[2,2] / (online_table[2,2] + online_table[2,1])
probability_ol
```

```
## [1] 0.6388889
```

iii)P(Loan = 1) (the proportion of loan acceptors) iii)

$$\mathbf{P}(\text{Loan} = 1) = 288/288{+}2712$$

```
probability_loan <- sum(Train_normalised$`Personal Loan`==1) / length(Train_normalised$`Personal Loan`)
probability_loan
```

```
## [1] 0.096
```

iv) P(CC = 1 | Loan = 0)
v)

$$\mathbf{P}(\text{CC} = 1 \mid \text{Loan} = 0) = 806/806{+}1906$$

```
probability_ccnl <-credit_table[1,2] / (credit_table[1,2] + credit_table[1,1])
probability_ccnl
```

```
## [1] 0.2971976
```

v)P(Online = 1 | Loan = 0) v)

$$\mathbf{P}(\text{Online} = 1 \mid \text{Loan} = 0) = 1621/1621{+}1091$$

```
probability_onl <- online_table[1,2] / (online_table[1,2] + online_table[1,1])
probability_onl
```

```
## [1] 0.5977139
```

vi)P(Loan = 0) vi)

$$\mathbf{P}(\text{Loan} = 0) = 2712/2712{+}288$$

```
probability_nl <- sum(Train_normalised$`Personal Loan`==0) / length(Train_normalised$`Personal Loan`)
probability_nl
```

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

Part E : The computed quantities from above were used for the Naive Bayes probability P(Loan = 1 | CC = 1, Online = 1).

$\mathbf{P}(\text{Loan} = 1 \mid \text{CC} = 1,\ \text{Online} = 1) = (0.6388 \text{ x } 0.2777 \text{ x } 0.096) \ / \ (0.6388 \text{ x } 0.2777 \text{ x } 0.096 + 0.5977 \text{ x } 0.2972 \text{ x } 0.904) = 0.095$

```
(probability_ol * probability_ccl * probability_loan) / (probability_ol * probability_ccl * probability_
```

```
## [1] 0.09591693
```

Part F: The pivot table values from (B) is compared with Naive bayes probability.

From the Naive Bayes classifier, a higher value for P(Loan = 1 | CC = 1, Online = 1) is obtained than with the direct computation obtained in part B. Interestingly, in part D we got the value of P(Loan = 1) as 0.096 same as value obtained in Naive Bayes classifier. So the concerned person being an online user with a bank-issued credit card is independent of the probability of the person accepting loan as suggested by Naive Bayes approach.

Part G: Performing Naive Bayes on the data using the `naiveBayes()` function and assigning to the variable `naive_data`. The value that is obtained for 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) from Naive bayes as seen is 0.09591693, which is equal to the value derived in part E.

```
naive_data <- naiveBayes(`Personal Loan`~Online+CreditCard,data=Train_normalised)
naive_data
```

```
##
## 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 0.4022861 0.5977139
##   1 0.3611111 0.6388889
##
##    CreditCard
## Y           0         1
##   0 0.7028024 0.2971976
##   1 0.7222222 0.2777778
```

AUC Value and ROC Curve

Plotting AUC (Area under curve) and ROC (Receiver Operating Characteristic) curve below using the `predict()` function by passing `naive_data` followed by using the `roc()` function and the `plot.roc()` (removing the second column).

```
label_predicted <-predict(naive_data,Valid_normalised, type = "raw")
head(label_predicted)
```

```
##              0          1
## [1,] 0.9107805 0.08921953
## [2,] 0.9107805 0.08921953
## [3,] 0.8955384 0.10446160
## [4,] 0.8955384 0.10446160
```

```
## [5,] 0.9181923 0.08180773
## [6,] 0.9107805 0.08921953
```

```r
roc(Valid_normalised$Online, label_predicted[,2])
```

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

## Setting direction: controls < cases

##
## Call:
## roc.default(response = Valid_normalised$Online, predictor = label_predicted[,     2])
##
## Data: label_predicted[, 2] in 821 controls (Valid_normalised$Online 0) < 1179 cases (Valid_normalise
## Area under the curve: 1
```

```r
plot.roc(Valid_normalised$Online,label_predicted[,2])
```

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

## Setting direction: controls < cases
```