Assignment 3

Kevin Gardner

2/28/2022

# ————————————————

# Following is the link to my GitHub account:

# <https://github.com/Kgardner22/64060_-kgardner>

# ————————————————

IMPORT AND PREPARE DATA:

Import the UniversalBank.csv file

UniversalBank <- read.table('C:/R/MyData/UniversalBank.csv', header = T, sep = ',')   
  
summary(UniversalBank)

## ID Age Experience Income ZIP.Code   
## Min. : 1 Min. :23.00 Min. :-3.0 Min. : 8.00 Min. : 9307   
## 1st Qu.:1251 1st Qu.:35.00 1st Qu.:10.0 1st Qu.: 39.00 1st Qu.:91911   
## Median :2500 Median :45.00 Median :20.0 Median : 64.00 Median :93437   
## Mean :2500 Mean :45.34 Mean :20.1 Mean : 73.77 Mean :93153   
## 3rd Qu.:3750 3rd Qu.:55.00 3rd Qu.:30.0 3rd Qu.: 98.00 3rd Qu.:94608   
## Max. :5000 Max. :67.00 Max. :43.0 Max. :224.00 Max. :96651   
## Family CCAvg Education Mortgage   
## Min. :1.000 Min. : 0.000 Min. :1.000 Min. : 0.0   
## 1st Qu.:1.000 1st Qu.: 0.700 1st Qu.:1.000 1st Qu.: 0.0   
## Median :2.000 Median : 1.500 Median :2.000 Median : 0.0   
## Mean :2.396 Mean : 1.938 Mean :1.881 Mean : 56.5   
## 3rd Qu.:3.000 3rd Qu.: 2.500 3rd Qu.:3.000 3rd Qu.:101.0   
## Max. :4.000 Max. :10.000 Max. :3.000 Max. :635.0   
## Personal.Loan Securities.Account CD.Account Online   
## Min. :0.000 Min. :0.0000 Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.000 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.0000   
## Median :0.000 Median :0.0000 Median :0.0000 Median :1.0000   
## Mean :0.096 Mean :0.1044 Mean :0.0604 Mean :0.5968   
## 3rd Qu.:0.000 3rd Qu.:0.0000 3rd Qu.:0.0000 3rd Qu.:1.0000   
## Max. :1.000 Max. :1.0000 Max. :1.0000 Max. :1.0000   
## CreditCard   
## Min. :0.000   
## 1st Qu.:0.000   
## Median :0.000   
## Mean :0.294   
## 3rd Qu.:1.000   
## Max. :1.000

Create a copy of the original data file to preserve

Original\_File <- UniversalBank

Load required libraries

library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

library(reshape2) #used for melt() and dcast();

## Warning: package 'reshape2' was built under R version 4.1.2

library(e1071) #used for naiveBayes();

Prepare the data by converting predictor and target variable to factors

UniversalBank$CreditCard=as.factor(UniversalBank$CreditCard)  
UniversalBank$Online=as.factor(UniversalBank$Online)  
UniversalBank$Personal.Loan=as.factor(UniversalBank$Personal.Loan)

We need to divide the data into training (60%) and validation (40%) sets

set.seed(64060)  
  
Train\_Index <- createDataPartition(UniversalBank$Personal.Loan, p=0.6, list = FALSE) #60% for train data  
Train.df <- UniversalBank[Train\_Index,]  
Validation.df <- UniversalBank[-Train\_Index,] #Remaining 40% for validation data

REQUIREMENT A:

Create a pivot table for the training data with Online as a column variable, CreditCard as a row variable, and Personal.Loan as a secondary row variable. The values inside the table should convey the count. Use functions melt() and cast(), or function table().

Pivot table created using ftable

Table1 <- xtabs(~ CreditCard + Online + Personal.Loan, data=Train.df)  
ftable(Table1)

## Personal.Loan 0 1  
## CreditCard Online   
## 0 0 772 75  
## 1 1152 120  
## 1 0 309 34  
## 1 479 59

Optional view of this same pivot table using melt();

Table1\_Long=melt(Table1, measure.vars=c("No", "Yes"), variable.name="Personal.Loan", value.name = "value")  
Table1\_Long

## CreditCard Online Personal.Loan value  
## 1 0 0 0 772  
## 2 1 0 0 309  
## 3 0 1 0 1152  
## 4 1 1 0 479  
## 5 0 0 1 75  
## 6 1 0 1 34  
## 7 0 1 1 120  
## 8 1 1 1 59

Optional view of this same pivot table using dcast();

Table1\_Wide = dcast(Table1\_Long, CreditCard + Online ~ Personal.Loan, value.var = "value" )  
Table1\_Wide

## CreditCard Online 0 1  
## 1 0 0 772 75  
## 2 0 1 1152 120  
## 3 1 0 309 34  
## 4 1 1 479 59

REQUIREMENT B:

Looking at the pivot tables created, what is the probability that this customer will accept the loan offer (Personal.Loan=1)?

ftable(Table1)

## Personal.Loan 0 1  
## CreditCard Online   
## 0 0 772 75  
## 1 1152 120  
## 1 0 309 34  
## 1 479 59

P(Personal.Loan=1 | CreditCard=1, Online=1)

((59/(479+59)) = (59/538) = 0.1096654

ANSWER: 0.1096654

REQUIREMENT C:

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

table(CreditCard=Train.df$CreditCard, Personal.Loan=Train.df$Personal.Loan)

## Personal.Loan  
## CreditCard 0 1  
## 0 1924 195  
## 1 788 93

table(Online=Train.df$Online, Personal.Loan=Train.df$Personal.Loan)

## Personal.Loan  
## Online 0 1  
## 0 1081 109  
## 1 1631 179

REQUIREMENT D:

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

1. P(CreditCard=1 | Personal.Loan=1) (93/(195+93)) = (93/288) = 0.3229 #Note: I’m using the CreditCard table above

* ANSWER = 0.3229

1. P(Online=1 | Personal.Loan=1) (179/(109+179)) = (179/288) = 0.6215 #Note: I’m using the Online table above

* ANSWER = 0.6215

1. P(Personal.Loan=1) ((195+93)/(1924+788+195+93)) = (288/3000) = 0.096 #Note: I’m using the CreditCard table above

ANSWER = 0.096

1. P(CreditCard=1 | Personal.Loan=0) (788/(1924+788)) = (788/2712) = 0.2906 #Note: I’m using the CreditCard table above

* ANSWER = 0.2906

1. P(Online=1 | Personal.Loan=0) (1631/(1081+1631)) = (1631/2712) = 0.6014 #Note: I’m using the Online table above

* ANSWER = 0.6014

1. P(Personal.Loan=0) ((1924+788)/(1924+788+195+93)) = (2712/3000) = 0.904 #Note: I’m using the CreditCard table above

* ANSWER = 0.904

REQUIREMENT E: Use the quantities computed above to compute the naive Bayes probability P(Personal.Loan=1 | CreditCard=1, Online=1)

Using the quantities from the tables generated in requirement C, we can compute the Naive Bayes Calculations as follows:

P = ((93/288)(179/288)(288/3000)) / (((93/288)(179/288)(288/3000))+((788/2712)(1631/2712)(2712/3000))) P = (((0.3229167)(0.6215278)(0.096)) / (((0.3229167)(0.6215278)(0.096)) / ((0.2905605)(0.6014012)(0.904))) P = 0.0192674 / (0.0192674 + 0.1579681) P = 0.0192674 / 0.1772355 P = 0.1087107

ANSWER = 0.1087107

REQUIREMENT F: Compare the value calculated in requirement E with the one obtained from the pivot table in requirement B.

In requirement B, we calculated this as: P(Personal.Loan=1 | CreditCard=1, Online=1) ((59/(479+59)) = (59/538) = 0.1096654 This is the Complete (Exact) Bayes Calculation

In requirement E, we calculated this as: P = (0.0192674 / 0.1772355) = 0.1087107 This is the Naive Bayes Calculation as described on page 194 of our textbook.

Which is a more accurate estimate?

ANSWER = The answer of 0.1096654 calculated in requirement B is more accurate. This is the Complete (Exact) Bayes Calculation that we calculated from the pivot tables. It does not make any assumptions as does the Naive Bayes Calculation in requirement E. Naive Bayes (E) assumes conditional independence while Bayes theorum (B) does not. This being said, Naive Bayes can provide a close estimate and typically, this has very little if any impact on the rank order of the output.

REQUIREMENT G: Which of the entries in this table are needed for computing P(Personal.Loan=1 | CreditCard=1, Online=1)?

ANSWER: The entries in the table needed to compute this are the results where CreditCard=1 and Online=1 showing the results of 479 observations for Personal.Loan=0 and 59 observations for Personal.Loan=1. We do not need the other data in the table. We then compute this by taking 59/(479+59) = 0.1096654.

Run naiveBayes on the data. Examine the model output on training data and find the entry that corresponds to P(Personal.Loan=1 | CreditCard=1, Online=1). Compare this to the number you obtained in requirement E.

nb.model<-naiveBayes(Personal.Loan~CreditCard+Online, data=Train.df)  
To\_Predict=data.frame(CreditCard="1", Online="1")  
predict(nb.model, To\_Predict, type='raw') #type set to raw to get probabilities;

## 0 1  
## [1,] 0.8912894 0.1087106

These results show, given CreditCard=1 and Online=1, the probability of the personal loan being accepted (Personal.Loan=1) is 0.1087106.

The number we calculated in requirement E was 0.1087107

There is a slight difference in these numbers due to rounding.

The niaveBayes model in requirement G computed the same value we manually calculated in requirement E. This Naive Bayes calculation assumes conditional independence while Bayes theorum, calculated in requirement B, does not. Therefore, the Bayes calculation in requirement B (0.1096654) is more accurate. This being said, Naive Bayes can provide a close estimate and typically, this has very little if any impact on the rank order of the output.