Assignment 3

Jared Bartee

2023-10-09

```
bank = read.csv("UniversalBank.csv")
summary(bank)
```

```
##        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
```

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

```
set.seed(1)
train.index <- sample(row.names(bank), 0.6*dim(bank)[1])
test.index <- setdiff(row.names(bank), train.index)
train.df <- bank[train.index, ]
test.df <- bank[test.index, ]
train <- bank[train.index, ]
test = bank[train.index,]
```

#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().

```
melted.bank = melt(train,id=c("CreditCard","Personal.Loan"),variable=
"Online")

## Warning: attributes are not identical across measure variables; they will
be
## dropped

recast.bank=dcast(melted.bank,CreditCard+Personal.Loan~Online)

## Aggregation function missing: defaulting to length

recast.bank[,c(1:2,14)]

##   CreditCard Personal.Loan Online
## 1          0             0   1924
## 2          0             1    198
## 3          1             0    801
## 4          1             1     77
```

#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)].

Answer: Probability of Loan acceptance give having a bank credit card and user of online services is 77/3000 = 2.6%

#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.

```
melted.bankc1 = melt(train,id=c("Personal.Loan"),variable = "Online")

## Warning: attributes are not identical across measure variables; they will
be
## dropped

melted.bankc2 = melt(train,id=c("CreditCard"),variable = "Online")
```

```
## Warning: attributes are not identical across measure variables; they will
be
## dropped

recast.bankc1=dcast(melted.bankc1,Personal.Loan~Online)

## Aggregation function missing: defaulting to length

recast.bankc2=dcast(melted.bankc2,CreditCard~Online)

## Aggregation function missing: defaulting to length

Loanline=recast.bankc1[,c(1,13)]
LoanCC=recast.bankc2[,c(1,14)]


Loanline

##   Personal.Loan Online
## 1             0   2725
## 2             1    275

LoanCC

##   CreditCard Online
## 1          0   2122
## 2          1    878
```

#D Compute the following quantities [P(A | B) means "the probability ofA given B"]: i. P(CC = 1 | Loan = 1) (the proportion of credit card holders among the loan acceptors);ii. P(Online = 1 | Loan = 1);iii. P(Loan = 1) (the proportion of loan acceptors);iv. P(CC = 1 | Loan = 0);v. P(Online = 1 | Loan = 0);vi. P(Loan = 0)

```
table(train[,c(14,10)])

##            Personal.Loan
## CreditCard    0    1
##          0 1924  198
##          1  801   77

table(train[,c(13,10)])

##       Personal.Loan
## Online    0    1
##      0 1137  109
##      1 1588  166

table(train[,c(10)])

##
##    0    1
## 2725  275
```

Answers:

i. 77/(77+198)= 28%

ii. 166/166+109)= 60.3%

iii. 275/(275+2725)= 9.2%

iv. 801/(801+1924)= 29.4%

v. 1588/(1588+1137)= 58.3%

vi. 2725/(2725+275)= 90.8%

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

```
((77/(77+198))*(166/(166+109))*(275/(275+2725)))/(((77/(77+198))*(166/(166+109))*(275/(275+2725)))+((801/(801+1924))*(1588/(1588+1137))*2725/(2725+275)))
```

```
## [1] 0.09055758
```

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

Answer: 9.05% is very similar to the 9.7% the difference between the exact method and the naive-baize method is the exact method would need the exact same independent variable classifications to predict, whereas the naive Bayes method does not.

#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).

```
naive.train = train.df[,c(10,13:14)]
naive.test = test.df[,c(10,13:14)]
naivebayes = naiveBayes(Personal.Loan~.,data=naive.train)
naivebayes
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##         0         1
## 0.90833333 0.09166667
##
## Conditional probabilities:
##    Online
## Y          0         1
##   0 0.4172477 0.5827523
```

```
##   1 0.3963636 0.6036364
##
##    CreditCard
## Y          0         1
##   0 0.706055 0.293945
##   1 0.720000 0.280000
```

Answer: The naive Bayes is the exact same output we received in the previous methods. (.280)(.603)(.09)/(.280.603.09+.29.58.908) = .09 which is the same response provided as above.