Assignment 3

Jacob Fabian

2022-10-09

```
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("tidyr")
library("ggplot2")
library("rpart")
library("caret")
```

```
## Loading required package: lattice
```

```
library('FNN')
library('melt')
library('MASS')
```

```
##
## Attaching package: 'MASS'

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

```
library('reshape2')
```

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

## The following object is masked from 'package:tidyr':
##
##     smiths
```

```r
library('naivebayes')
```

```
## naivebayes 0.9.7 loaded
```

```r
setwd("~/Downloads")
bank = read.csv("UniversalBank.csv")
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().

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

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

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

```
## Aggregation function missing: defaulting to length
```

```r
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
```

###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)].

###2.6%

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

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

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

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

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

```
## Aggregation function missing: defaulting to length
```

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

```
## Aggregation function missing: defaulting to length
```

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

Loanline
```

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

```r
LoanCC
```

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

###d. Compute the following quantities [P (A | B) means "the probability of A given B"]:P (CC = 1 | Loan = 1) (the proportion of credit card holders among the loan acceptors)P(Online=1|Loan=1)P (Loan = 1) (the proportion of loan acceptors)P(CC=1|Loan=0)P(Online=1|Loan=0)P(Loan=0)

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

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

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

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

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

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

I.28%

II. 60.3%

III. .2%

IV. 29.4%

V. 58.3%

VI. 90.8%

###Use the quantities computed above to compute the naive Ba1 probability P(Loan = 1 | CC = 1, Online = 1).

```r
((77/(77+198))*(166/(166+109))*(275/(275+2725)))/(((77/(77+198))*(166/(166+109))*(275/(275+2725)))+((80
```

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

f. Compare this value with the one obtained from the pivot table in (b). Which is a more accurate estimate? 9.05% are very similar to the 9.7% the difference between the exact method and the naive-baise method is the exact method would need the the exact same independent variable classifications to predict, where 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)? In R, 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).

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

```
##
## ================================== Naive Bayes ==================================
##
##  Call:
## naive_bayes.formula(formula = Personal.Loan ~ ., data = naive.train)
##
## ---------------------------------------------------------------------------------
##
## Laplace smoothing: 0
```

```
##
## ----------------------------------------------------------------------------
##
##  A priori probabilities:
##
##          0          1
## 0.90833333 0.09166667
##
## ----------------------------------------------------------------------------
##
##  Tables:
##
## ----------------------------------------------------------------------------
##  ::: Online (Bernoulli)
## ----------------------------------------------------------------------------
##
## Online         0         1
##      0 0.4172477 0.3963636
##      1 0.5827523 0.6036364
##
## ----------------------------------------------------------------------------
##  ::: CreditCard (Bernoulli)
## ----------------------------------------------------------------------------
##
## CreditCard        0        1
##          0 0.706055 0.720000
##          1 0.293945 0.280000
##
## ----------------------------------------------------------------------------
```

###(.280)(.603)(.09)/(.280.603.09+.29.58.908) = .09 which is the same response provided in the previous methods.