Machine Learning Assignment-3 Pradeep reddy kethu

```
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("softImpute")
```

```
## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

## Loaded softImpute 1.4-1

##
## Attaching package: 'softImpute'

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

```
library("ggplot2")
library("ROCR")
library("caret")
```

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

```r
library("tidyverse")
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --

## v tibble  3.1.8     v stringr 1.4.1
## v readr   2.1.2     v forcats 0.5.2
## v purrr   0.3.4
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x softImpute::complete() masks tidyr::complete()
## x Matrix::expand()       masks tidyr::expand()
## x dplyr::filter()        masks stats::filter()
## x dplyr::lag()           masks stats::lag()
## x purrr::lift()          masks caret::lift()
## x Matrix::pack()         masks tidyr::pack()
## x Matrix::unpack()       masks tidyr::unpack()
```

```r
library("SnowballC")
library("Hmisc")
```

```
## Loading required package: survival
##
## Attaching package: 'survival'
##
## The following object is masked from 'package:caret':
##
##     cluster
##
## Loading required package: Formula
##
## Attaching package: 'Hmisc'
##
## The following object is masked from 'package:softImpute':
##
##     impute
##
## The following objects are masked from 'package:dplyr':
##
##     src, summarize
##
## The following objects are masked from 'package:base':
##
##     format.pval, units
```

```r
library('tinytex')
library('FNN')
library("dplyr")
library("tidyr")
library("caTools")
library("reshape2")
```

```
##
```

```
## Attaching package: 'reshape2'
##
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```
library("rattle")
```

```
## Loading required package: bitops
##
## Attaching package: 'bitops'
##
## The following object is masked from 'package:Matrix':
##
##     %&%
##
## Rattle: A free graphical interface for data science with R.
## Version 5.5.1 Copyright (c) 2006-2021 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library("rpart")
library("e1071")
```

```
##
## Attaching package: 'e1071'
##
## The following object is masked from 'package:Hmisc':
##
##     impute
##
## The following object is masked from 'package:softImpute':
##
##     impute
```

```
rm(list=ls())
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().

```r
melted.bank = melt(train,id=c("CreditCard","Personal.Loan"),variable= "Online")
recast.bank=dcast(melted.bank,CreditCard+Personal.Loan~Online)
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)].

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

#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

```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"]: 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)

```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
```

```r
probability1<-77/(77+198)
probability1
```

```
## [1] 0.28
```

```r
probability2<-166/(166+109)
probability2
```

```
## [1] 0.6036364
```

```r
probability3<-275/(275+2725)
probability3
```

```
## [1] 0.09166667
```

```r
probability4<-801/(801+1924)
probability4
```

```
## [1] 0.293945
```

```r
probability5<-1588/(1588+1137)
probability5
```

```
## [1] 0.5827523
```

```
probability6<-2725/(2725+275)
probability6
```

```
## [1] 0.9083333
```

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

```
(probability1*probability2*probability3)/((probability1*probability2*probability3)+(probability4*probabi
```

```
## [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).

```
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
```

the naive bayes is the exact same output we recieved in the previous methods. (.280)*(.603)*(.09)/(.280.*603*.09+.29.*58*.908) = .09 which is the same response provided as above.