Assignmrnt 3

Nourah

3/1/2021

```r
###reading the file
df <-read.csv(file="~/Desktop/MSBA-spring 2021/ML/ML3/xid-170057171_1.csv")
```

```r
###loading packages
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
```

```r
library(class)
library(caret)
```

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

```
## Loading required package: ggplot2
```

```r
library(FNN)
```

```
##
## Attaching package: 'FNN'
```

```
## The following objects are masked from 'package:class':
##
##     knn, knn.cv
```

```r
library(e1071)
library(reshape2)
```

```r
### First I would like to change numerical data to categorical data
df$Personal.Loan<-factor(df$Personal.Loan)
df$Online<-factor(df$Online)
df$CreditCard<-factor(df$CreditCard)
```

```r
### Second I will partition the data into training (60%) and validation (40%) sets.
set.seed(123)
df1 <- sample(2,nrow(df), replace=TRUE, prob=c(0.6,0.4))
tdf <- df[df1==1,]
vdf <- df[df1==2,]
```

```r
### 1-Create a pivot table for the training data with Online as a column variable, CC as a row v
ariable, and Loan as a secondary row variable. The values inside the table should convey the cou
nt. In R use functions melt() and cast(), or function table(). In Python, use panda dataframe me
thods melt() and pivot().
Pivot <- table(df$Personal,df$CreditCard)
pivot_df <- as.data.frame(Pivot)
colnames(pivot_df) <- c("PersonalLoan", "CreditCard","Online")
pivot_df
```

```
##   PersonalLoan CreditCard Online
## 1            0          0   3193
## 2            1          0    337
## 3            0          1   1327
## 4            1          1    143
```

```r
### Creating pivot table by the function table()
```

```r
### 1-Create a pivot table for the training data with Online as a column variable, CC as a row v
ariable, and Loan as a secondary row variable. The values inside the table should convey the cou
nt. In R use functions melt() and cast(), or function table(). In Python, use panda dataframe me
thods melt() and pivot().

melt1<- melt(tdf,id=c("CreditCard","Personal.Loan"),variable="Online")
```

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

```r
cast1<-dcast(melt1,CreditCard+Personal.Loan~Online)
```

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

```r
cast1[,c(1:2,14)]
```

```
##   CreditCard Personal.Loan Online
## 1          0             0   1925
## 2          0             1    191
## 3          1             0    813
## 4          1             1     82
```

```r
###creating pivot table using melt() and cast() functions
```

```r
###B. Consider the task of classifying a customer who owns a bank credit card and is actively us
ing online banking services. Looking at the pivot table, what is the probability that this custo
mer will accept the loan offer? [This is the probability of loan acceptance (Loan = 1) condition
al on having a bank credit card (CC = 1) and being an active user of online banking services (On
line = 1)].

P1 = table(tdf[,c(10,13,14)])
P2<- as.data.frame(P1)
P2
```

```
##   Personal.Loan Online CreditCard Freq
## 1             0      0          0  776
## 2             1      0          0   69
## 3             0      1          0 1149
## 4             1      1          0  122
## 5             0      0          1  317
## 6             1      0          1   36
## 7             0      1          1  496
## 8             1      1          1   46
```

```r
###C. Create two separate pivot tables for the training data. One will have Loan (rows) as a fun
ction of Online (columns) and the other will have Loan (rows) as a function of CC.
table(tdf[,c(10,13)])
```

```
##              Online
## Personal.Loan    0    1
##             0 1093 1645
##             1  105  168
```

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

```
##              CreditCard
## Personal.Loan    0    1
##             0 1925  813
##             1  191   82
```

```r
###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)
Prop_1<-table(tdf[,c(14,10)])
F2<-Prop_1[2,2]/(Prop_1[2,2]+Prop_1[1,2])
F2
```

```
## [1] 0.3003663
```

```r
###ii. P(Online = 1 | Loan = 1)
Prop_2<-table(tdf[,c(13,10)])
F2<-Prop_2[2,2]/(Prop_2[2,2]+Prop_2[1,2])
F2
```

```
## [1] 0.6153846
```

```r
###iii. P(Loan = 1) (the proportion of loan acceptors)
Prop_3<-table(df[,10])
F3<-Prop_3[2]/(Prop_3[2]+Prop_3[1])
F3
```

```
##     1
## 0.096
```

```r
###iv. P(CC=1|Loan=0)
Prop_4<-table(tdf[,c(14,10)])
F4<-Prop_4[2,1]/(Prop_4[2,1]+Prop_4[1,1])
F4
```

```
## [1] 0.2969321
```

```r
###v. P(Online = 1 | Loan = 0)
Prop_5<-table(tdf[,c(13,10)])
F5<-Prop_5[2,1]/(Prop_5[2,1]+Prop_5[1,1])
F5
```

```
## [1] 0.6008035
```

```r
###vi. P(Loan = 0)

Prop_6<-table(tdf[,10])
F6<-Prop_6[1]/(Prop_6[1]+Prop_6[2])
F6
```

```
##         0
## 0.9093324
```

```r
###E. Use the quantities computed above to compute the naive Bayes probability P(Loan = 1 | CC =
1, Online = 1).
## NBP=(S1 * S2 * S3)/ [(S1 * S2 * S3)) + (S4 * S5 * S6)]
## NBP=(0.3003663*0.6153846*0.096)/[(0.3003663*0.3003663*0.6153846)+(0.2969321*0.6008035*0.90933
24)]
## NBP= 0.081
```

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

## The value from the pivot table = 0.3003663 and the value from the NBP = 0.081. In my opinion
the value from the pivot table is more accurate because naive Bayes is based on assumptions.
```

```r
###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 t
hat corresponds to P(Loan = 1 | CC = 1, Online = 1). Compare this to the number you obtained in
(E).

NB<-tdf[,c(10,13,14)]
NB1 <- naiveBayes(Personal.Loan~.,data=NB)
NB1
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##          0          1
## 0.90933245 0.09066755
##
## Conditional probabilities:
##    Online
## Y           0         1
##   0 0.3991965 0.6008035
##   1 0.3846154 0.6153846
##
##    CreditCard
## Y           0         1
##   0 0.7030679 0.2969321
##   1 0.6996337 0.3003663
```

```r
## In q E, NBP= 0.081 and we got 0.915
```