Assignment_3_FML

Aditya Ashish Kulkarni

2024-03-09

Instruction : The file UniversalBank.csv contains data on 5000 customers of Universal Bank. The data include customer demographic information (age, income, etc.), the customer's relationship with the bank (mortgage, securities account, etc.), and the customer response to the last personal loan campaign (Personal Loan). Among these 5000 customers, only 480 (= 9.6%) accepted the personal loan that was offered to them in the earlier campaign. In this exercise, we focus on two predictors: Online (whether or not the customer is an active user of online banking services) and Credit Card (abbreviated CC below) (does the customer hold a credit card issued by the bank), and the outcome Personal Loan (abbreviated Loan below).Partition the data into training (60%) and validation (40%) sets.

```
library(caret)
```

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

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

Reading CVS file

```
#importing CVS file

df <- read.csv("UniversalBank.csv")
```

```
#Using dim
dim(df)
```

```
## [1] 5000   14
```

For partitioning the data into training 60% and validation 40%

```
set.seed(69)

train.index <- sample(row.names(df), 0.6*dim(df)[1])
valid.index <- setdiff(row.names(df), train.index)

names(df)[10] = "Personal_Loan"

train.df <- df[train.index,]
valid.df <- df[valid.index,]

cat("size of Training Set:", nrow(train.df))
```

```
## size of Training Set: 3000
```

```
cat("Size of Validation Set:", nrow(valid.df))
```

```
## Size of Validation Set: 2000
```

Question 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().

Creating Pivot table

```
library(e1071)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v lubridate 1.9.3     v tibble    3.2.1
## v purrr     1.0.2     v tidyr     1.3.0
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x purrr::lift()   masks caret::lift()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
#Making a pivot table of all Online, CC and Loan
df_2= train.df %>%
  select(CreditCard,Personal_Loan,Online)

#Making Pivot table
p.df_2= ftable(df_2)
print(p.df_2)
```

```
##                          Online    0    1
## CreditCard Personal_Loan
## 0          0                     746 1165
##            1                      78  131
## 1          0                     309  481
##            1                      37   53
```

Question 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)].

Calculating Probability

```
#Probability

#P(Loan=1/CC=1,Online=1)= Total number of customers with CC = 1 and Online = 1 / Number of customers wi

prob_loan_accept <- (p.df_2[4,2]/(sum(p.df_2[3,2],p.df_2[4,2])))



cat("Probability that Loan acceptance:", (prob_loan_accept))
```

```
## Probability that Loan acceptance: 0.09925094
```

Question 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.

Creating First Pivot Table

```
#Creating First Pivot table - Personal Loan as Row and Online as Column

PT_1= train.df %>%
  select(Personal_Loan,Online)

#Making Pivot Table
p.PT_1= ftable(PT_1)
print(p.PT_1)
```

```
##               Online    0    1
## Personal_Loan
## 0                     1055 1646
## 1                      115  184
```

Creating Second Pivot Table

```
#Creating Second Pivot table - Personal Loan as Row and CreditCard  as Column
PT_2= train.df %>%
  select(Personal_Loan,CreditCard)

#Making Pivot Table
p.PT_2= ftable(PT_2)
print(p.PT_2)
```

```
##               CreditCard    0    1
## Personal_Loan
## 0                       1911  790
## 1                        209   90
```

Question 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)

Computing Quantities

```r
# I. P(CC = 1 / Loan = 1)

Prob_1 <- (p.PT_2[2,2]/(sum(p.PT_2[2,])))
cat("Probability of P(CC = 1 / Loan = 1):", Prob_1)
```

```
## Probability of P(CC = 1 / Loan = 1): 0.3010033
```

```r
# II. P(Online = 1 / Loan = 1)

Prob_2 <- (p.PT_1[2,2]/(sum(p.PT_1[2,])))
cat("Probability of P(Online = 1 / Loan = 1):", Prob_2)
```

```
## Probability of P(Online = 1 / Loan = 1): 0.6153846
```

```r
# III. P(Loan = 1) (the proportion of loan acceptors)

Prob_3 <- (sum(p.PT_2[2,])/sum(p.PT_2))
cat("Probability of P(Loan = 1) (the proportion of loan acceptors):",Prob_3 )
```

```
## Probability of P(Loan = 1) (the proportion of loan acceptors): 0.09966667
```

```r
# IV. P(CC = 1 / Loan = 0)

Prob_4 <- (p.PT_2[1,2]/sum(p.PT_2[1,]))
cat("Probability of P(CC = 1 / Loan = 0):",Prob_4)
```

```
## Probability of P(CC = 1 / Loan = 0): 0.2924843
```

```r
# V. P(Online = 1 / Loan = 0)

Prob_5 <- (p.PT_1[1,2]/sum(p.PT_1[1,]))
cat("Probability of P(Online = 1 / Loan = 0):",Prob_5)
```

```
## Probability of P(Online = 1 / Loan = 0): 0.6094039
```

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

Prob_6 <- (sum(p.PT_2[1,])/sum(p.PT_2))
cat("Probability of P(Loan = 0):",Prob_6)
```

```
## Probability of P(Loan = 0): 0.9003333
```

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

P(Loan = 1 / CC = 1, Online = 1) = P(CC=1 Loan=1) * P(Online=1 Loan=1)*P(Loan=1) / P(CC=1,Online=1)

```r
#Naive Bayes probability formula for P(Loan = 1 / CC = 1, Online = 1) = P(CC=1/Loan=1) * P(Online=1/Loa

# P(cc=1, Online=1)
prob_7 = sum(p.PT_2[,2])/sum(p.PT_2)
prob_8 = sum(p.PT_1[,1])/sum(p.PT_1)

#Applying Naive bayes formula

naive_bayes_prob <- ((Prob_1*Prob_2*Prob_3)/(prob_7*prob_8))

naive_bayes_prob
```

```
## [1] 0.1613771
```

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

Answer) Here, Loan Acceptance probability using bayes method is : 0.09925094 and Naive Bayes is : 0.1037165

According to this bayes : 0.09925094 is more accurat, Naive Bayes is not accurate and has small deviation from actual value because it assumes all the events are independent.

Question 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).

We would need all the entries in our training data set where Online = 1 and Credit Card = 1.

```r
#Using naive Bayes on training dataset
Naive <- naiveBayes(Personal_Loan ~., data = train.df)
nt <- predict(Naive, newdata = train.df)
nt.2 = predict(Naive, newdata = train.df,type = "raw")
nt=data.frame(nt)
nt.2 = round(nt.2,2)
nt.2 = data.frame(nt.2)

# Comparing predicton and actual values in training data set:
compare = data.frame(ID = rep(NA,3000),
  NaiveBayes_prediction = rep(NA,3000),
                      Actual = rep(NA,3000),
  Naive_Probablity = rep(NA,3000))

compare$ID = train.df$ID
compare$NaiveBayes_prediction = nt$nt
compare$Actual = train.df$Personal_Loan
compare$Naive_Probablity = nt.2[,2]
head(compare)
```

```
##     ID NaiveBayes_prediction Actual Naive_Probablity
## 1 2016                     0      0             0.01
## 2 1233                     0      0             0.06
## 3 1371                     0      0             0.00
```

```
## 4 3021                    1      0             0.94
## 5 2471                    1      0             1.00
## 6 3183                    0      0             0.00
```

Finding entries corresponding P(Loan = 1/ Credit = 1 , Online = 1):

```
special = train.df %>%
  filter( Personal_Loan==1 ,Online ==1 , CreditCard== 1)

head(special[,c('ID','Personal_Loan','Online','CreditCard')])
```

```
##     ID Personal_Loan Online CreditCard
## 1  982             1      1          1
## 2  972             1      1          1
## 3 4148             1      1          1
## 4 4963             1      1          1
## 5 3055             1      1          1
## 6  210             1      1          1
```

So we know that for above IDs personal_loan = 1, Credit Card = 1 and Online = 1. Hence, we can check value of probability for any of these case.

```
# Checking for ID=982 and 972
compare %>%
  filter(ID==c(982,972))
```

```
##    ID NaiveBayes_prediction Actual Naive_Probablity
## 1 982                    1      1                1
## 2 972                    1      1                1
```

Hence, the probability as per the naive bayes model is 1.00 for asked entries. However, Naive Bayes probability that we calculated while considering only online = 1 and Credit Card = 1 in part E had a value of 0.16. **Therefore, we can conclude that it's important to take all columns into consideration while making a model since it can lead to huge differences in final results.**

Calculating accuracy for our model :

```
CM = confusionMatrix(as.factor(train.df$Personal_Loan), as.factor(compare$NaiveBayes_prediction))

CM$overall[1]*100
```

```
## Accuracy
## 88.53333
```

Note: Hence, we observe that our model is only 88%. This means that Naive Bayes is not the right classifier for this data. This is expected since Naive Bayes is supposed to go well with categorical data but our data has many numeric variables. Due to this, Naive Bayes classifier needs to use gaussian method which does not always lead to satisfying results.