Assignment 3

Rob DiVincenzo 10-15-2023 BA 64060

This is the submission for Assignment 3.

#Read data  
DF=read.csv("./UniversalBank.csv") # Read the Online Retail csv file  
#Load libraries  
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(caret)

## Loading required package: ggplot2

## Loading required package: lattice

library(ISLR)  
library(FNN)  
library(gmodels)  
library(reshape)

##   
## Attaching package: 'reshape'

## The following object is masked from 'package:dplyr':  
##   
## rename

library(reshape2)

##   
## Attaching package: 'reshape2'

## The following objects are masked from 'package:reshape':  
##   
## colsplit, melt, recast

library(e1071)   
  
set.seed(351)  
  
#Select variables in need  
#DF <- select(DF, Age, Experience, Income, Family, CCAvg, Education, Mortgage, Personal.Loan, Securities.Account, CD.Account, Online, CreditCard) # Select a subset of variables  
  
#Create dummy variables for Education (categorical variable with 2+ levels)  
#DF$Education\_1 <- as.integer(DF$Education == 1)  
#DF$Education\_2 <- as.integer(DF$Education == 2)  
#DF$Education\_3 <- as.integer(DF$Education == 3)  
#Drop no longer needed Education field  
#DF<-DF[,-6]  
  
#Partition Data into 60% Train, 40% val  
Train\_Index=createDataPartition(DF$Age,p=0.60, list=FALSE)  
Train\_Data = DF[Train\_Index,] # create the training data; we include all columns; note the index is row, column  
Val\_Data = DF[-Train\_Index,] # create the val set  
  
#Create copy of data for normalization  
#train.norm.df <- Train\_Data  
#val.norm.df <- Val\_Data  
  
# use preProcess() from the caret package to normalize data, ignore target variable personal loan  
#norm.values <- preProcess(Train\_Data[,], method=c("center", "scale"))  
  
#Replace columns with normalized values  
#train.norm.df[,] <- predict(norm.values, Train\_Data[,])  
#val.norm.df[,] <- predict(norm.values, Val\_Data[,])  
  
  
#A  
  
#use melt() to melt data  
mlt <- melt(Train\_Data)

## No id variables; using all as measure variables

#use cast() to create pivot table  
pivot\_table <- dcast(mlt, Train\_Data$CreditCard + Train\_Data$Personal.Loan ~ Train\_Data$Online)

## Aggregation function missing: defaulting to length

#rename columns  
colnames(pivot\_table) <- c("Credit Card", "Personal Loan", "Online = 0", "Online = 1")  
  
  
#B  
# Looking at the pivot table, the count of Loan = 1, CC = 1, and Online = 1 is 49.  
# The count of CC = 1 and Online = 1 is 542 (49 where loan = 1 + 493 where loan = 0)  
# Therefore the probability is 49 / 542, or 0.09040590406 -> 9.04%  
  
#C  
  
#use cast() to create pivot table with Loans (rows) as a function of Online (columns)   
pivot\_table\_online <- dcast(mlt, Train\_Data$Personal.Loan ~ Train\_Data$Online)

## Aggregation function missing: defaulting to length

#rename columns  
colnames(pivot\_table\_online) <- c("Personal Loan", "Online = 0", "Online = 1")  
  
#use cast() to create pivot table with Loans (rows) as a function of CC (columns)   
pivot\_table\_cc <- dcast(mlt, Train\_Data$Personal.Loan ~ Train\_Data$CreditCard)

## Aggregation function missing: defaulting to length

#rename columns  
colnames(pivot\_table\_cc) <- c("Personal Loan", "Credit Card = 0", "Credit Card = 1")  
  
#D  
#i. P(CC = 1 | Loan = 1)  
#count of customers with CC = 1 and Loan = 1 = 84  
#count of customers with Loan = 1 = (200 + 82) = 282  
# 84 / 282 = 0.29787 = 29.787 -> 29.79%  
  
#ii. P(Online = 1 | Loan = 1)   
#count of customers with Online = 1 and Loan = 1 = 166  
#count of customers with Loan = 1 = 282  
# 166 / 282 = 0.58865 = 58.865% => 58.87%  
  
#iii. P(Loan = 1) (the proportion of loan acceptors)  
#count of customers with Loan = 1 = 282  
#total count of customers (1928 + 790) + (201 + 82) = 2718 + 283 = 3001  
# 282 / 3001 = 0.09397 = 9.397% -> 9.40%  
  
#vi. P(CC = 1 | Loan = 0)  
#count of customers with CC = 1 and Loan = 0 = 787  
#count of customers with Loan = 0 = (1930 + 787) = 2717  
# 787 / 2717 = 0.28966 = 28.966% -> 28.97%  
  
#v. P(Online = 1 | Loan = 0)  
#count of customers with Online =1 and Loan = 0 = 1644  
#count of customers with Loan = 0 = 2717  
# 1644/2717 = 0.60508 = 60.508% -> 60.51%  
  
#vi. P(Loan = 0)  
#count of customers with Loan = 0 = 2717  
#total count of customers = 3001  
# 2717 / 3001 = 0.90537 = 90.537% -> 90.54%   
  
#E  
#translate into naive bayes probability formula  
#P(Loan=1∣CC=1,Online=1)= [ P(CC=1,Online=1∣Loan=1) \* P(Loan=1) ] / [ P(CC=1, Online=1) ]  
  
#expand numerator conditionals  
#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) ]  
  
#expand denominator conditionals  
#P(Loan=1∣CC=1,Online=1)= [ P(CC=1|Loan=1) \* P(Online=1|Loan=1) \* P(Loan=1) ] / [ ( P(CC=1|Loan=1) \* P(Online=1|Loan=1) \* P(Loan=1) ) + ( P(CC=1|Loan=0) \* P(Online=1|Loan=0)\*P(Loan=0) )]  
  
#plug in numbers  
#P(Loan=1,CC=1,Online=1)= [0.29787 \* 0.58865 \* 0.09397 ] / [ ( 0.29787 \* 0.58865 \* 0.09397 ) + (0.28966 \* 0.60508 \* 0.90537) ]  
  
#solve numerator  
#P(Loan=1,CC=1,Online=1)= 0.01648 / [ ( 0.29787 \* 0.58865 \* 0.09397 ) + (0.28966 \* 0.60508 \* 0.90537) ]  
  
#solve denominator  
#P(Loan=1,CC=1,Online=1)= 0.01648 / [ 0.01648 + 0.15868 ]  
#P(Loan=1,CC=1,Online=1)= 0.01648 / 0.17516  
  
#solve formula  
#P(Loan=1,CC=1,Online=1)= 0.09409 = 9.409% -> 9.41%  
  
#F  
#The calculation in E (9.41%) is similar to B (9.04%). While it's possible that B is more accurate since it is based on so few predictors it will quickly become more impractical and complex as the number of predictors and new / unique records increase, which is when the Naive Bayes classification would become more efficient and valuable.  
  
  
#G  
  
#create naive bayes model  
nb\_model <-naiveBayes(Personal.Loan~CreditCard+Online,data = Train\_Data)  
nb\_model

##   
## Naive Bayes Classifier for Discrete Predictors  
##   
## Call:  
## naiveBayes.default(x = X, y = Y, laplace = laplace)  
##   
## A-priori probabilities:  
## Y  
## 0 1   
## 0.90536488 0.09463512   
##   
## Conditional probabilities:  
## CreditCard  
## Y [,1] [,2]  
## 0 0.2896577 0.4536870  
## 1 0.2957746 0.4571958  
##   
## Online  
## Y [,1] [,2]  
## 0 0.6050791 0.4889237  
## 1 0.5845070 0.4936767

#create prediction data  
prediction\_data <- data.frame(CC = 1, Online = 1)  
#plug in the data to predict the model outcome  
predict\_prob <- predict(nb\_model, newdata = prediction\_data, type = "raw")

## Warning in predict.naiveBayes(nb\_model, newdata = prediction\_data, type =  
## "raw"): Type mismatch between training and new data for variable 'CreditCard'.  
## Did you use factors with numeric labels for training, and numeric values for  
## new data?

#print the model outcome for Personal Loan = 1  
print(predict\_prob[1, "1"])

## 1   
## 0.0914601

#As shown in the work above, all of the entries in the table were needed for computing P(Loan = 1 | CC = 1, Online = 1).  
#Running naive Bayes on the data results in a very similar number .90146 -> 9.15% to the number (9.41%) obtained in E.