Assignment 3 - Kerrie Mars

Kerrie Mars

2022-10-15

setwd("/Users/Kyle Mars/Desktop/Kerrie/Machine Learning")  
#clear existing data in Environment  
rm(list=ls())  
#load data  
bank <- read.csv("UniversalBank.csv", header = TRUE)  
#find dimension of data frame  
dim(bank)

## [1] 5000 14

#open libraries  
library(ISLR)

## Warning: package 'ISLR' was built under R version 4.1.3

library(forecast) #for evaluating performance

## Warning: package 'forecast' was built under R version 4.1.3

## Registered S3 method overwritten by 'quantmod':  
## method from  
## as.zoo.data.frame zoo

library(class) #for allowing a numerical output variable

## Warning: package 'class' was built under R version 4.1.3

library(psych) #for creating dummies

## Warning: package 'psych' was built under R version 4.1.3

library(caret) #for data partition, normalize data

## Warning: package 'caret' was built under R version 4.1.3

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 4.1.3

##   
## Attaching package: 'ggplot2'

## The following objects are masked from 'package:psych':  
##   
## %+%, alpha

## Loading required package: lattice

library(FNN) #for Performing knn classification

## Warning: package 'FNN' was built under R version 4.1.3

##   
## Attaching package: 'FNN'

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

library(e1071) #for performing Naive Bayes

## Warning: package 'e1071' was built under R version 4.1.3

library(reshape) #for creating pivot tables

## Warning: package 'reshape' was built under R version 4.1.3

##   
## Attaching package: 'reshape'

## The following object is masked from 'package:class':  
##   
## condense

library(reshape2) #for cast function

## Warning: package 'reshape2' was built under R version 4.1.3

##   
## Attaching package: 'reshape2'

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

library(MASS)

## Warning: package 'MASS' was built under R version 4.1.3

library(gmodels)

## Warning: package 'gmodels' was built under R version 4.1.3

summary(bank)

## ID Age Experience Income ZIP.Code   
## Min. : 1 Min. :23.00 Min. :-3.0 Min. : 8.00 Min. : 9307   
## 1st Qu.:1251 1st Qu.:35.00 1st Qu.:10.0 1st Qu.: 39.00 1st Qu.:91911   
## Median :2500 Median :45.00 Median :20.0 Median : 64.00 Median :93437   
## Mean :2500 Mean :45.34 Mean :20.1 Mean : 73.77 Mean :93153   
## 3rd Qu.:3750 3rd Qu.:55.00 3rd Qu.:30.0 3rd Qu.: 98.00 3rd Qu.:94608   
## Max. :5000 Max. :67.00 Max. :43.0 Max. :224.00 Max. :96651   
## Family CCAvg Education Mortgage   
## Min. :1.000 Min. : 0.000 Min. :1.000 Min. : 0.0   
## 1st Qu.:1.000 1st Qu.: 0.700 1st Qu.:1.000 1st Qu.: 0.0   
## Median :2.000 Median : 1.500 Median :2.000 Median : 0.0   
## Mean :2.396 Mean : 1.938 Mean :1.881 Mean : 56.5   
## 3rd Qu.:3.000 3rd Qu.: 2.500 3rd Qu.:3.000 3rd Qu.:101.0   
## Max. :4.000 Max. :10.000 Max. :3.000 Max. :635.0   
## Personal.Loan Securities.Account CD.Account Online   
## Min. :0.000 Min. :0.0000 Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.000 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.0000   
## Median :0.000 Median :0.0000 Median :0.0000 Median :1.0000   
## Mean :0.096 Mean :0.1044 Mean :0.0604 Mean :0.5968   
## 3rd Qu.:0.000 3rd Qu.:0.0000 3rd Qu.:0.0000 3rd Qu.:1.0000   
## Max. :1.000 Max. :1.0000 Max. :1.0000 Max. :1.0000   
## CreditCard   
## Min. :0.000   
## 1st Qu.:0.000   
## Median :0.000   
## Mean :0.294   
## 3rd Qu.:1.000   
## Max. :1.000

#change numerical variables to categorical first  
bank$Personal.Loan = as.factor(bank$Personal.Loan)  
bank$Online = as.factor(bank$Online)  
bank$CreditCard = as.factor(bank$CreditCard)  
  
#Create training and validation sets  
#Partitioning the data into Training(60%) and Validation(40%)  
#set seed for reproducting the partition  
set.seed(1)   
  
train.index <- sample(rownames(bank), dim(bank)[1]\*0.6)  
train.df <- bank[train.index, ]  
valid.index <- setdiff(rownames(bank), train.index)  
valid.df <- bank[valid.index, ]  
train <- bank[train.index, ]  
valid <- 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.   
  
#use melt() to stack a set of columns into a single column of data  
melted.bank <- melt(train.df, id=c("CreditCard", "Personal.Loan"), variable = "Online")

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

#use cast() to reshape data and generate pivot table  
cast.bank <- dcast(melted.bank, CreditCard + Personal.Loan ~ Online)

## Aggregation function missing: defaulting to length

cast.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?  
#NOTE: [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)].  
x = (77/3000)  
Answer = x\*100 #express as a percentage  
Answer

## [1] 2.566667

#ANSWER: The probability of Personal.Loan = 1 and CreditCard = 1 and an active user of Online = 77/3000  
#in the training data which is 2.6%.  
  
#C - Create two separate pivot tables for the training data.   
#One will have Loan (rows) as a function of Online (columns)  
#The other will have Loan (rows) as a function of CC.  
  
melted.bank2 <- melt(train.df, id = c("Personal.Loan"), variable = "Online")

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

melted.bank3 <- melt(train.df, id = c("CreditCard"), variable = "Online")

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

cast.bank2 <- cast(melted.bank2, Personal.Loan ~ Online)

## Aggregation requires fun.aggregate: length used as default

cast.bank3 <- cast(melted.bank3, CreditCard ~ Online)

## Aggregation requires fun.aggregate: length used as default

LoanOnline <- cast.bank2[, c(1, 13)]  
LoanCC <- cast.bank3[, c(1, 14)]  
  
#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)  
table(train.df[,c(14,10)])

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

77/(198+77)

## [1] 0.28

#ii. P(Online = 1 | Loan = 1)  
table(train.df[,c(13,10)])

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

166/(166+109)

## [1] 0.6036364

#iii. P(Loan = 1)(the proportion of loan acceptors)  
table(train.df[,c(10)])

##   
## 0 1   
## 2725 275

275/(2725+275)

## [1] 0.09166667

#iv. P(CC = 1 | Loan = 0)  
table(train.df[,c(14,10)])

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

801/(1924+801)

## [1] 0.293945

#v. P(Online = 1 | Loan = 0)  
table(train.df[,c(13,10)])

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

1588/(1137+1588)

## [1] 0.5827523

#vi. P(Loan = 0)  
table(train.df[,c(10)])

##   
## 0 1   
## 2725 275

2725/(2725+275)

## [1] 0.9083333

#E - Use the quantities computed above to compute the naive Bayes probability  
#P(Loan = 1 | CC = 1, Online = 1)  
(.28\*.60363\*.09166)/((.28\*.60363\*.09166)+(.2939\*.5827\*.9083))

## [1] 0.09057373

#F - Compare this value with the one obtained from the pivot table in (B).  
#Which is a more accurate estimate?  
  
loan.nb <- naiveBayes(Personal.Loan ~ ., data = train.df)  
loan.nb

##   
## 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:  
## ID  
## Y [,1] [,2]  
## 0 2490.874 1459.635  
## 1 2386.156 1419.473  
##   
## Age  
## Y [,1] [,2]  
## 0 45.46972 11.35683  
## 1 45.01818 11.51370  
##   
## Experience  
## Y [,1] [,2]  
## 0 20.22899 11.37218  
## 1 19.83636 11.52459  
##   
## Income  
## Y [,1] [,2]  
## 0 65.85028 40.41044  
## 1 144.73455 30.88285  
##   
## ZIP.Code  
## Y [,1] [,2]  
## 0 93157.71 1769.086  
## 1 93107.51 1717.120  
##   
## Family  
## Y [,1] [,2]  
## 0 2.364404 1.152018  
## 1 2.618182 1.115434  
##   
## CCAvg  
## Y [,1] [,2]  
## 0 1.714073 1.556814  
## 1 3.903236 2.084294  
##   
## Education  
## Y [,1] [,2]  
## 0 1.848073 0.8380416  
## 1 2.200000 0.7593697  
##   
## Mortgage  
## Y [,1] [,2]  
## 0 52.09358 90.54271  
## 1 109.33455 164.55801  
##   
## Securities.Account  
## Y [,1] [,2]  
## 0 0.09761468 0.2968474  
## 1 0.12727273 0.3338859  
##   
## CD.Account  
## Y [,1] [,2]  
## 0 0.03192661 0.1758370  
## 1 0.26909091 0.4442959  
##   
## 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

#training  
  
pred.class <- predict(loan.nb, newdata = train.df)  
confusionMatrix(pred.class, train.df$Personal.Loan)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 2498 116  
## 1 227 159  
##   
## Accuracy : 0.8857   
## 95% CI : (0.8737, 0.8968)  
## No Information Rate : 0.9083   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.4189   
##   
## Mcnemar's Test P-Value : 2.86e-09   
##   
## Sensitivity : 0.9167   
## Specificity : 0.5782   
## Pos Pred Value : 0.9556   
## Neg Pred Value : 0.4119   
## Prevalence : 0.9083   
## Detection Rate : 0.8327   
## Detection Prevalence : 0.8713   
## Balanced Accuracy : 0.7474   
##   
## 'Positive' Class : 0   
##

#validation  
  
pred.class <- predict(loan.nb, newdata = valid.df)  
confusionMatrix(pred.class, valid.df$Personal.Loan)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 1628 85  
## 1 167 120  
##   
## Accuracy : 0.874   
## 95% CI : (0.8587, 0.8882)  
## No Information Rate : 0.8975   
## P-Value [Acc > NIR] : 0.9997   
##   
## Kappa : 0.4182   
##   
## Mcnemar's Test P-Value : 3.352e-07   
##   
## Sensitivity : 0.9070   
## Specificity : 0.5854   
## Pos Pred Value : 0.9504   
## Neg Pred Value : 0.4181   
## Prevalence : 0.8975   
## Detection Rate : 0.8140   
## Detection Prevalence : 0.8565   
## Balanced Accuracy : 0.7462   
##   
## 'Positive' Class : 0   
##

#The Naive Bayes is a more accurate estimate  
  
#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).  
  
naive.train <- train.df[,c(10, 13:14)]  
naive.test <- valid.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 giving us a similar number (rounding) to our method above which is .09.