Machine Learning 3

Yash Bhanusali

03/05/2022

library("reshape2")  
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")

##   
## Attaching package: 'tidyr'

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

library("ggplot2")  
library("ROCR")  
library("rpart")  
library("rpart.plot")  
library("caret")

## Loading required package: lattice

## Warning: package 'lattice' was built under R version 4.1.2

library("randomForest")

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':  
##   
## margin

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

library("tidyverse")

## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v tibble 3.1.4 v stringr 1.4.0  
## v readr 2.0.2 v forcats 0.5.1  
## v purrr 0.3.4

## -- Conflicts ------------------------------------------ tidyverse\_conflicts() --  
## x randomForest::combine() masks dplyr::combine()  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()  
## x purrr::lift() masks caret::lift()  
## x randomForest::margin() masks ggplot2::margin()

library("tm")

## Loading required package: NLP

##   
## Attaching package: 'NLP'

## The following object is masked from 'package:ggplot2':  
##   
## annotate

library("SnowballC")  
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("glmnet")

## Loaded glmnet 4.1-2

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

library("dummies")

## dummies-1.5.6 provided by Decision Patterns

library('tinytex')  
library('GGally')

## Registered S3 method overwritten by 'GGally':  
## method from   
## +.gg ggplot2

library('gplots')

##   
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':  
##   
## lowess

library('FNN')  
library("dplyr")  
library("tidyr")  
library("caTools")  
library("ggpubr")  
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())  
U\_bank <- read\_csv("UniversalBank (1).csv")

## Rows: 5000 Columns: 14

## -- Column specification --------------------------------------------------------  
## Delimiter: ","  
## dbl (14): ID, Age, Experience, Income, ZIP Code, Family, CCAvg, Education, M...

##   
## i Use `spec()` to retrieve the full column specification for this data.  
## i Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

View(U\_bank)

U\_bank <- read.csv("UniversalBank (1).csv")  
U\_bank$Personal.Loan = as.factor(U\_bank$Personal.Loan)  
U\_bank$Online = as.factor(U\_bank$Online)  
U\_bank$CreditCard = as.factor(U\_bank$CreditCard)  
set.seed(1)  
train.index <- sample(row.names(U\_bank), 0.6\*dim(U\_bank)[1])   
test.index <- setdiff(row.names(U\_bank), train.index)   
train.df <- U\_bank[train.index, ]  
test.df <- U\_bank[test.index, ]  
train <- U\_bank[train.index, ]  
test = U\_bank[train.index,]

##1]. A pivot table is being created for the training data with online as a column variable, with two row variables CC as a row variable and Loan as a secondary row variable. Inside the table values should convey the count. In R use functions melt() and cast() or function table ()

melted.U\_bank = melt(train,id=c("CreditCard","Personal.Loan"),variable= "Online")

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

recast.U\_bank=dcast(melted.U\_bank,CreditCard+Personal.Loan~Online)

## Aggregation function missing: defaulting to length

recast.U\_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

##2]The task here is to classify a customer who owns a bank credit card and actively uses the online banking services. After looking at the pivot table we can find out 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)].

##The probability of the loan being accepted by the prospective customer given they have a bank credit card and they use the online services is 77/3000 = 2.6%

##3]. Creating two separate pivot tables for the training data. One is having Loan (rows) as a function of Online (columns) and the other is having Loan (rows) as a function of CC.

melted.U\_bankc1 = melt(train,id=c("Personal.Loan"),variable = ("Online"))

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

recast.U\_bankc1=dcast(melted.U\_bank,Personal.Loan~Online)

## Aggregation function missing: defaulting to length

recast.U\_bankc1[,c(1:2,13)]

## Personal.Loan ID Online  
## 1 0 2725 2725  
## 2 1 275 275

melted.U\_bankc2 = melt(train,id=c("CreditCard"),variable = "Online")

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

recast.U\_bankc2=dcast(melted.U\_bank,CreditCard~Online)

## Aggregation function missing: defaulting to length

recast.U\_bankc2[,c(1:2,13)]

## CreditCard ID Online  
## 1 0 2122 2122  
## 2 1 878 878

recast.U\_bankc1=dcast(melted.U\_bankc1,Personal.Loan~Online)

## Aggregation function missing: defaulting to length

recast.U\_bankc2=dcast(melted.U\_bankc2,CreditCard~Online)

## Aggregation function missing: defaulting to length

RelLoanline=recast.U\_bankc1[,c(1,13)]  
RelLoanCC = recast.U\_bankc2[,c(1,14)]  
RelLoanline

## Personal.Loan Online  
## 1 0 2725  
## 2 1 275

RelLoanCC

## CreditCard Online  
## 1 0 2122  
## 2 1 878

##4]. Computing 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)

table(train[,c(14,10)])

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

table(train[,c(13,10)])

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

table(train[,c(10)])

##   
## 0 1   
## 2725 275

1. 77/(77+198)=28%
2. 166/(166+109)= 60.3% iii.275/(275+2725)=9.2%
3. 801/(801+1924)=29.4%
4. 1588/(1588+1137) = 58.3%
5. 2725/(2725+275) = 90.8% ##5]. Using the quantities computed above to compute the naive Ba1 probability P(Loan = 1 | CC = 1, Online = 1).

((77/(77+198))\*(166/(166+109))\*(275/(275+2725)))/(((77/(77+198))\*(166/(166+109))\*(275/(275+2725)))+((801/(801+1924))\*(1588/(1588+1137))\*2725/(2725+275)))

## [1] 0.09055758

##6]. Comparing 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.

##7]. 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. ##The same response provided as above (.280)(.603)(.09)/(.280.603.09+.29.58.908) = .09