Assignment 3 Machine Learning

Sumit Dutt Mishra

06/03/2021

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

## Loading required package: lattice

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 1.3.0 ──

## ✓ tibble 3.0.6 ✓ stringr 1.4.0  
## ✓ readr 1.4.0 ✓ forcats 0.5.1  
## ✓ 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

##   
## Attaching package: 'softImpute'

## The following object is masked from 'package:tidyr':  
##   
## complete

library("glmnet")

## Loaded glmnet 4.1-1

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("reshape2")

##   
## Attaching package: 'reshape2'

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

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())  
setwd("~/Downloads")

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

<0>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().

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

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

recast.bankdata=dcast(melted.bankdata,CreditCard+Personal.Loan~Online)

## Aggregation function missing: defaulting to length

recast.bankdata[,c(1:2,14)]

## CreditCard Personal.Loan Online  
## 1 0 0 1924  
## 2 0 1 198  
## 3 1 0 801  
## 4 1 1 77

1. 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)].

Probability of Loan acceptance given having a bank credit card and user of online services is 77/3000 = 2.6%

1. 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.

melted.bankdatac1 = melt(train,id=c("Personal.Loan"),variable = "Online")

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

melted.bankdatac2 = melt(train,id=c("CreditCard"),variable = "Online")

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

recast.bankdatac1=dcast(melted.bankdatac1,Personal.Loan~Online)

## Aggregation function missing: defaulting to length

recast.bankdatac2=dcast(melted.bankdatac2,CreditCard~Online)

## Aggregation function missing: defaulting to length

RelLoanline=recast.bankdatac1[,c(1,13)]  
RelLoanCC = recast.bankdatac2[,c(1,14)]  
  
RelLoanline

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

RelLoanCC

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

1. Compute the following quantities [P (A | B) means “the probability of A given B”]:
2. P (CC = 1 | Loan = 1) (the proportion of credit card holders among the loan acceptors)
3. P(Online=1|Loan=1)
4. P (Loan = 1) (the proportion of loan acceptors)
5. P(CC=1|Loan=0)
6. P(Online=1|Loan=0)
7. 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%
6. Use 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

1. Compare 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.
2. Which of 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. (.280)(.603)(.09)/(.280.603.09+.29.58.908) = .09 which is the same response provided as above.