Untitled

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
chooseCRANmirror(graphics = getOption("menu.graphics"), ind = 79,
                 local.only = FALSE)
install.packages("class")
```

```
## Installing package into 'C:/Users/ibeme/Documents/R/win-library/4.1'
## (as 'lib' is unspecified)

## package 'class' successfully unpacked and MD5 sums checked

## Warning: cannot remove prior installation of package 'class'

## Warning in file.copy(savedcopy, lib, recursive = TRUE): problem copying C:
## \Users\ibeme\Documents\R\win-library\4.1\00LOCK\class\libs\x64\class.dll to C:
## \Users\ibeme\Documents\R\win-library\4.1\class\libs\x64\class.dll: Permission
## denied

## Warning: restored 'class'

##
## The downloaded binary packages are in
##  C:\Users\ibeme\AppData\Local\Temp\Rtmp6DpOXN\downloaded_packages
```

```
install.packages("caret")
```

```
## Installing package into 'C:/Users/ibeme/Documents/R/win-library/4.1'
## (as 'lib' is unspecified)

## package 'caret' successfully unpacked and MD5 sums checked

## Warning: cannot remove prior installation of package 'caret'

## Warning in file.copy(savedcopy, lib, recursive = TRUE): problem copying C:
## \Users\ibeme\Documents\R\win-library\4.1\00LOCK\caret\libs\x64\caret.dll to C:
## \Users\ibeme\Documents\R\win-library\4.1\caret\libs\x64\caret.dll: Permission
## denied
```

```
## Warning: restored 'caret'

##
## The downloaded binary packages are in
##  C:\Users\ibeme\AppData\Local\Temp\Rtmp6DpOXN\downloaded_packages
```

```
install.packages("ISLR")
```

```
## Installing package into 'C:/Users/ibeme/Documents/R/win-library/4.1'
## (as 'lib' is unspecified)

## package 'ISLR' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\ibeme\AppData\Local\Temp\Rtmp6DpOXN\downloaded_packages
```

```
require(e1071)
```

```
## Loading required package: e1071
```

```
library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```
library(class)
library(ISLR)
```

Naive Bayes Algorithm

A.Create a pivot table for the training data with Online(col variable), CC (row variable), and Loan (secondary row variable).

- Imported data from Universal Bank CSV file.
- Two Predictors from the dataset : Online, CreditCard and Outcome: Personal_Loan
- Factorized the categorical variables
- Divide the data into 60% training and 40% validation
- Created a pivot table using ftable which conveys the count with Online as a col variable, CC as a row variable, and Loan as a secondary row variable.

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

#Selected 2 predictor variables(Online and CreditCard) and an outcome(Personal.Loan) variable

Universal_data <- Universal_data[,c(10,13,14)]
```

```r
#Factorized the categorical variables

Universal_data$Personal.Loan<-factor(Universal_data$Personal.Loan)
Universal_data$Online <- factor(Universal_data$Online)
Universal_data$CreditCard <- factor(Universal_data$CreditCard)
str(Universal_data)
```

```
## 'data.frame':    5000 obs. of  3 variables:
##  $ Personal.Loan: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 2 ...
##  $ Online       : Factor w/ 2 levels "0","1": 1 1 1 1 1 2 2 1 2 1 ...
##  $ CreditCard   : Factor w/ 2 levels "0","1": 1 1 1 1 2 1 1 2 1 1 ...
```

```r
set.seed(123)

# Data Partitioning - (60% for Training data and 40% validation data)

Personal_Loan.tr.in <- createDataPartition(Universal_data$Personal.Loan,p=0.6, list=FALSE) # 60% reserv
Personal_Loan.tr <-  Universal_data[Personal_Loan.tr.in,]
Personal_Loan.va <- Universal_data[-Personal_Loan.tr.in,] # Validation  data is rest

#str(Personal_Loan.tr)
#str(Personal_Loan.va)
#summary(Personal_Loan.tr)
#summary(Personal_Loan.va)

# Created a pivot table with Online as a col variable, CC as a row variable, and Loan as a secondary ro

ftable(Personal_Loan.tr,row.vars = c(3,1),col.vars = "Online")
```

```
##                          Online    0    1
## CreditCard Personal.Loan
## 0          0                     791 1144
##            1                      79  125
## 1          0                     310  467
##            1                      33   51
```

B.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)].
- Looking at the pivot table, Actual Probability p(Loan=1|CC=1,Online=1) = 51/(467+51) = 0.098456
  p(CC=1,Online=1|Loan=1).p(loan=1)/(p(cc=1,online=1|loan=1).p(loan=1))+(p(cc=1,online=1|loan=0).p(loan=0)) =(51/288)*(288/3000)/((51/3000)+((467/2712)*(2712/3000))) = 0.098456

```r
p<-((51/288)*(288/3000))/((51/3000)+((467/2712)*(2712/3000)))
print(paste("The actual Probability of P(Loan=1| CC=1, Online=1) from pivot table: ", p))
```

```
## [1] "The actual Probability of P(Loan=1| CC=1, Online=1) from pivot table:  0.0984555984555985"
```

C.Create two separate pivot tables for the training data.

- One Pivot table will have Loan (rows) as a function of Online (columns) and the other table will have Loan (rows) as a function of CC.
- Created two pivot tables using ftables

```
ftable(Personal_Loan.tr,row.vars ="Personal.Loan",col.vars = "Online")
```

```
##               Online    0    1
## Personal.Loan
## 0                    1101 1611
## 1                     112  176
```

```
ftable(Personal_Loan.tr,row.vars ="Personal.Loan",col.vars = "CreditCard")
```

```
##               CreditCard    0    1
## Personal.Loan
## 0                        1935  777
## 1                         204   84
```

D.Compute the following quantities [P(A | B) means "the probability ofA given B"]:

- Used ftable to create pivot tables and prop.table to calculate all the below proportions.

i. P(CC = 1 | Loan = 1) is : 0.291666666666667
ii. P(Online = 1 | Loan = 1) is : 0.611111111111111
iii. P(Loan = 1) is : 0.096
iv. P(CC = 1 | Loan = 0) is : 0.286504424778761
v. P(Online = 1 | Loan = 0) is : 0.594026548672566
vi. P(Loan = 0) is : 0.904

```
Loan_Online_Table <- prop.table(ftable(Personal_Loan.tr,row.vars ="Personal.Loan",col.vars = "Online"),m
Loan_CreditCard_Table <- prop.table(ftable(Personal_Loan.tr,row.vars ="Personal.Loan",col.vars = "Credit
Loan_Table <- prop.table(table(Personal_Loan.tr$Personal.Loan),margin = NULL)

#i.P(CC = 1 | Loan = 1) (the proportion of credit card holders among the loan acceptors)
a<-Loan_CreditCard_Table[2,2]
print(paste("The proportion of credit card holders among the loan acceptors P(CC = 1 | Loan = 1) is :",a
```

```
## [1] "The proportion of credit card holders among the loan acceptors P(CC = 1 | Loan = 1) is : 0.2916
```

```
#ii.P(Online = 1 | Loan = 1)
b<-Loan_Online_Table[2,2]
print(paste("The proportion of active online users among the loan acceptors P(Online = 1 | Loan = 1) is
```

```
## [1] "The proportion of active online users among the loan acceptors P(Online = 1 | Loan = 1) is : 0.6
```

```r
#iii.P(Loan = 1) (the proportion of loan acceptors)
c<-Loan_Table[2]
print(paste("The proportion of loan acceptorsP(Loan = 1) is :",c))
```

```
## [1] "The proportion of loan acceptorsP(Loan = 1) is : 0.096"
```

```r
#iv.P(CC = 1 | Loan = 0)
d<-Loan_CreditCard_Table[1,2]
print(paste("(the proportion of credit card holders among the loan rejectors P(CC = 1 | Loan = 0) is :"
```

```
## [1] "(the proportion of credit card holders among the loan rejectors P(CC = 1 | Loan = 0) is : 0.2865
```

```r
#v.P(Online = 1 | Loan = 0)
e<-Loan_Online_Table[1,2]
print(paste("The proportion of active online users among the loan rejectors (Online = 1 | Loan = 0) is
```

```
## [1] "The proportion of active online users among the loan rejectors (Online = 1 | Loan = 0) is : 0.59
```

```r
#vi.P(Loan = 0)
f<-Loan_Table[1]
print(paste("The proportion of loan rejectors P(Loan = 0) is :",f))
```

```
## [1] "The proportion of loan rejectors P(Loan = 0) is : 0.904"
```

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

- Using all the conditional probabilties from D to compute Naive Bayes Probability Naive_Bayes_Prob <- 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

```r
Naive_Bayes_Prob <- (Loan_CreditCard_Table[2,2]*Loan_Online_Table[2,2]*Loan_Table[2])/((Loan_CreditCard
Naive_Bayes_Prob
```

```
##         1
## 0.1000861
```

```r
print(paste("The Naive Probability of P(Loan=1| CC=1, Online=1): ", Naive_Bayes_Prob))
```

```
## [1] "The Naive Probability of P(Loan=1| CC=1, Online=1):  0.100086055488176"
```

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

- Looking at the pivot table, The probability of that the customer will accept the loan offer from (B) P(Loan=1|CC=1,Online=1) = 0.098455598
- The probability from naive bayes (E) :P(Loan=1|CC=1,Online=1) = 0.1000861
- There is a minimal difference between (B) and (E) which is 0.001630502
- When it comes to comparison, (0.098 is similar to 0.100)
- Pivot table(Actual) probability(B) is more accurate than Naive Bayes probability(E).Since pivot table does not make the assumption of the probabilities (taking a loan if you are a cc holder and if you are an online customer) being independent. And also, there are few variables and categories to consider. So the Pivot table probability is feasible in this case.

```
print(paste("Looking at the pivot table, the Probability of P(Loan=1| CC=1, Online=1): ", p))
```

```
## [1] "Looking at the pivot table, the Probability of P(Loan=1| CC=1, Online=1):  0.0984555984555985"
```

```
print(paste("The Naive Probability of P(Loan=1| CC=1, Online=1): ", Naive_Bayes_Prob))
```

```
## [1] "The Naive Probability of P(Loan=1| CC=1, Online=1):  0.100086055488176"
```

```
print(paste("The Minimal Difference of the both probabilities from (B) and (E): ", Naive_Bayes_Prob-p))
```

```
## [1] "The Minimal Difference of the both probabilities from (B) and (E):  0.00163045703257775"
```

G. Run Naive Bayes Model on the data. Compare this to the number you obtained in (E).

- 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 will need Personal Loan (Which is dependent variable) and Online, Credit card columns (Predictor variables)
- Performed Naive Bayes model on the training data.

OBSERVATIONS FROM (E):

The Naive Bayes probability of P(Loan=1| CC=1, Online=1) from (E) is 0.1000860

OBSERVATIONS FROM THE NAIVE BAYES MODEL:

Prior Probabilities: P(Loan=0) = 0.904 P(Loan=1) = 0.096

Conditional Probabilities:

P(CC = 1 | Loan = 0) = 0.2865044 P(CC = 1 | Loan = 1) = 0.2916667 P(Online = 1 | Loan = 0) = 0.5940265 P(Online = 1 | Loan = 1) = 0.6111111

CALCULATION OF NAIVE BAYES PROBABILITY WITH THE VALUES OBTAINED FROM NAIVE BAYES MODEL :

(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))

=((0.291667)*(0.611111)*(0.096))/(((0.291667)*(0.611111)*(0.096))+((0.286504)*(0.594026)*(0.904))= 0.1000864

- The Prior, conditional and the Naive Bayes Probabilities from the Naive Bayes model is similar to (E)

```
set.seed(123)

loan.nb<-naiveBayes(Personal.Loan ~ CreditCard+Online, data = Personal_Loan.tr)

loan.nb
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##     0     1
## 0.904 0.096
##
## Conditional probabilities:
##    CreditCard
## Y           0         1
##   0 0.7134956 0.2865044
##   1 0.7083333 0.2916667
##
##    Online
## Y           0         1
##   0 0.4059735 0.5940265
##   1 0.3888889 0.6111111
```

```
l<-((0.291667)*(0.611111)*(0.096))/(((0.291667)*(0.611111)*(0.096))+((0.286504)*(0.594026)*(0.904)))


print(paste("the Naive Bayes Probability from the Model is :",l))
```

```
## [1] "the Naive Bayes Probability from the Model is : 0.100086358778749"
```

```
#Confusion Matrix for the training data


#Training set
set.seed(123)

pred.prob_train <- predict(loan.nb,newdata = Personal_Loan.tr,type="raw")

#Table with Personal training data and their predicted probabilities using raw arguments
predict_table_train<-cbind(Personal_Loan.tr,pred.prob_train)
head(predict_table_train)
```

```
##   Personal.Loan Online CreditCard         0          1
## 1             0      0          0 0.9082737 0.09172629
## 2             0      0          0 0.9082737 0.09172629
## 4             0      0          0 0.9082737 0.09172629
## 5             0      0          1 0.9061594 0.09384060
## 7             0      1          0 0.9021538 0.09784623
## 9             0      1          0 0.9021538 0.09784623
```

```
pred.train <- predict(loan.nb,newdata = Personal_Loan.tr)
confusionMatrix(pred.train,Personal_Loan.tr$Personal.Loan)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2712  288
##          1    0    0
##
##                Accuracy : 0.904
##                  95% CI : (0.8929, 0.9143)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : 0.5157
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 1.000
##             Specificity : 0.000
##          Pos Pred Value : 0.904
##          Neg Pred Value :   NaN
##              Prevalence : 0.904
##          Detection Rate : 0.904
##    Detection Prevalence : 1.000
##       Balanced Accuracy : 0.500
##
##        'Positive' Class : 0
##
```

```
head(pred.train)
```

```
## [1] 0 0 0 0 0 0
## Levels: 0 1
```

```
#Validation set

pred.prob_valid <- predict(loan.nb,newdata = Personal_Loan.va,type="raw")
predict_table_valid <- cbind(Personal_Loan.va,pred.prob_valid)
head(predict_table_valid)
```

```
##    Personal.Loan Online CreditCard         0          1
## 3              0      0          0 0.9082737 0.09172629
```

```
## 6               0      1          0 0.9021538 0.09784623
## 8               0      0          1 0.9061594 0.09384060
## 11              0      0          0 0.9082737 0.09172629
## 15              0      0          0 0.9082737 0.09172629
## 16              0      1          1 0.8999139 0.10008606
```

```r
pred.valid <- predict(loan.nb,newdata = Personal_Loan.va)
summary(pred.valid)
```

```
##    0    1
## 2000    0
```

```r
confusionMatrix(pred.valid,Personal_Loan.va$Personal.Loan)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1808  192
##          1    0    0
##
##                Accuracy : 0.904
##                  95% CI : (0.8902, 0.9166)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : 0.5192
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 1.000
##             Specificity : 0.000
##          Pos Pred Value : 0.904
##          Neg Pred Value :   NaN
##              Prevalence : 0.904
##          Detection Rate : 0.904
##    Detection Prevalence : 1.000
##       Balanced Accuracy : 0.500
##
##        'Positive' Class : 0
##
```

Including ROC Plots

The output shows that using a cutoff of 0.901 produces the maximum value for Sensitivity (of 0.839) + Specificity (of 0.184).

```r
require(pROC)
```

```
## Loading required package: pROC
```

```
## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
# Note the delayed probabilities are in column 1
roc(Personal_Loan.va$Personal.Loan,pred.prob_valid[,1])
```

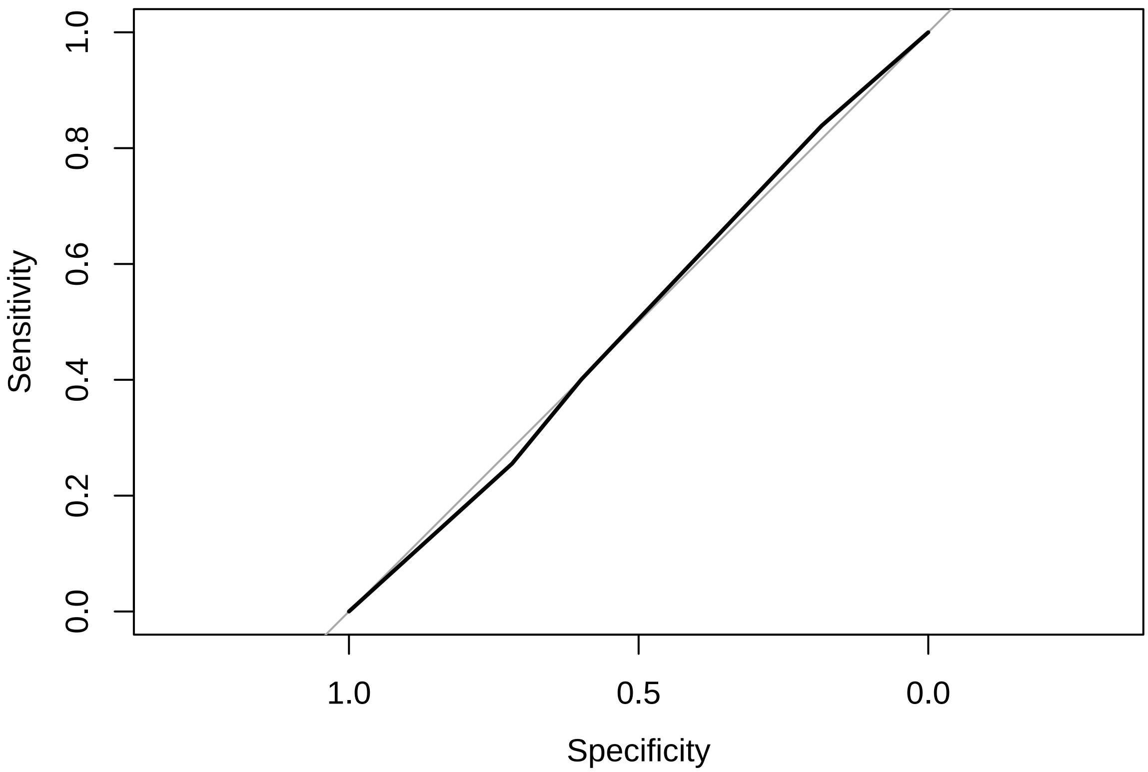
```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

##
## Call:
## roc.default(response = Personal_Loan.va$Personal.Loan, predictor = pred.prob_valid[,     1])
##
## Data: pred.prob_valid[, 1] in 1808 controls (Personal_Loan.va$Personal.Loan 0) < 192 cases (Personal_
## Area under the curve: 0.5014
```

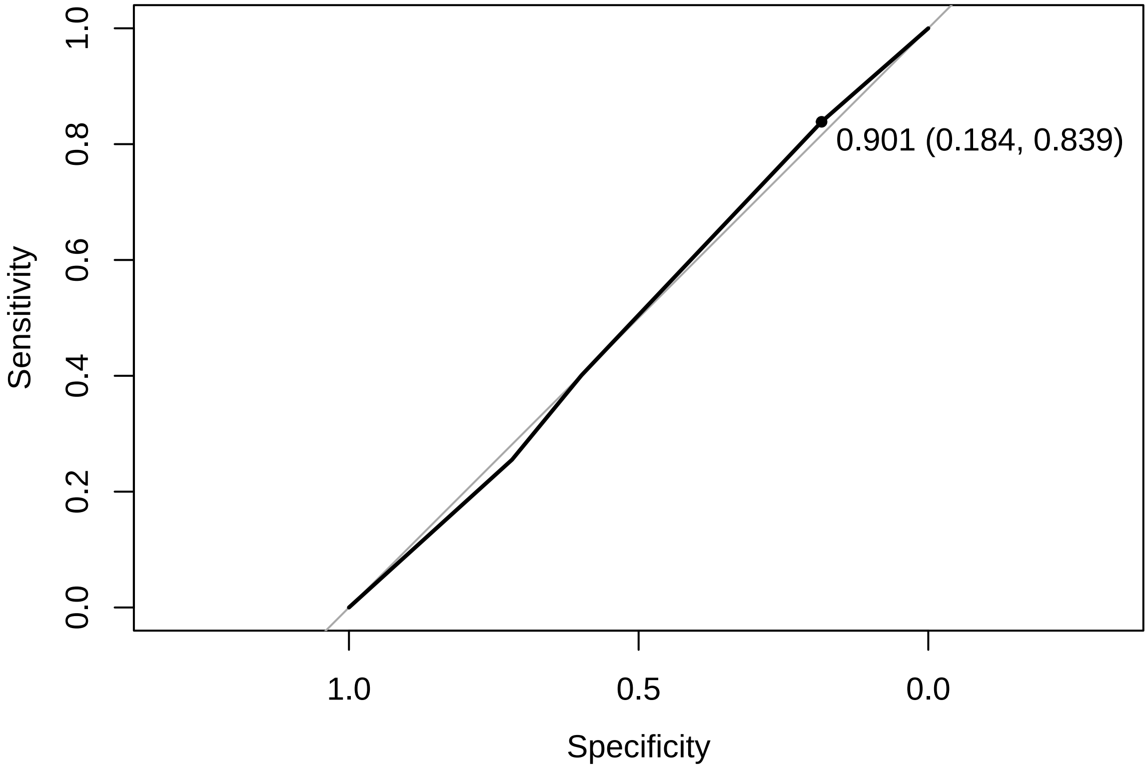
```
plot.roc(Personal_Loan.va$Personal.Loan,pred.prob_valid[,1])
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```




```
plot.roc(Personal_Loan.va$Personal.Loan,pred.prob_valid[,1],print.thres="best")
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.