Naive Bayes for classification

Shashidhar Reddy Boreddy

2022-10-18

```
#calling required libraries
library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```
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(ggplot2)
library(lattice)
library(knitr)
library(rmarkdown)
library(e1071)
```

```
#importing The Data Set
library(readr)
sb_data <- read.csv("C:/Users/shash/Dropbox/PC/Downloads/UniversalBank.csv")
View(sb_data)
```

```
####The consecutive section simply extracts the csv file, removes the ID and zip code (like last time,
bsr1 <- sb_data %>% select(Age, Experience, Income, Family, CCAvg, Education, Mortgage, Personal.Loan ,
bsr1$CreditCard <- as.factor(bsr1$CreditCard)
bsr1$Personal.Loan <- as.factor((bsr1$Personal.Loan))
bsr1$Online <- as.factor(bsr1$Online)
```

```
##This creates the data separation, as well as the train and validation data.
select.var <- c(8,11,12)
set.seed(23)
Train.Index = createDataPartition(bsr1$Personal.Loan, p=0.60, list=FALSE)
Train_Data = bsr1[Train.Index,select.var]
Validation.Data = bsr1[-Train.Index,select.var]
```

```
##A. Create a pivot table for the training data with Online as a column variable, CC as a row variable,
#In the resulting pivot table CC and LOAN are both rows, and online is a column.
attach(Train_Data)
##ftable is defined as "function table".
ftable(CreditCard,Personal.Loan,Online)
```

```
##                          Online    0    1
## CreditCard Personal.Loan
## 0          0                     773 1127
##            1                      82  114
## 1          0                     315  497
##            1                      39   53
```

```
detach(Train_Data)
```

###Given Online=1 and CC=1, we add 53 (Loan=1 from ftable) to 497 (Loan=0 from ftable), which equals 550, to get the conditional probability that Loan=1. 53/550 = 0.096363 or 9.64% of the time.

```
##B. Consider the task of classifying a customer who owns a bank credit card and is actively using onli

prop.table(ftable(Train_Data$CreditCard,Train_Data$Online,Train_Data$Personal.Loan),margin=1)
```

```
##                0          1
##
## 0 0  0.90409357 0.09590643
##   1  0.90813860 0.09186140
## 1 0  0.88983051 0.11016949
##   1  0.90363636 0.09636364
```

###The above code generates a Percentage pivot table that depicts the loan probability depending on CC and online.

```
##C. Create two separate pivot tables for the training data. One will have Loan (rows) as a function of

attach(Train_Data)
ftable(Personal.Loan,Online)
```

```
##               Online    0    1
## Personal.Loan
## 0                    1088 1624
## 1                     121  167
```

```
ftable(Personal.Loan,CreditCard)
```

```
##                 CreditCard    0    1
## Personal.Loan
## 0                          1900  812
## 1                           196   92
```

```
detach(Train_Data)
```

###In the Above first , "Online" compensates a column, "Loans" compensates a row, and "Credit Card" compensates a column.

```
##D. Compute the following quantities [P(A | B) means "the probability ofA given B"]:
prop.table(ftable(Train_Data$Personal.Loan,Train_Data$CreditCard),margin=)
```

```
##               0          1
##
## 0  0.63333333 0.27066667
## 1  0.06533333 0.03066667
```

```
prop.table(ftable(Train_Data$Personal.Loan,Train_Data$Online),margin=1)
```

```
##              0         1
##
## 0  0.4011799 0.5988201
## 1  0.4201389 0.5798611
```

sbi) 92/288 = 0.3194 or 31.94%

sbii) 167/288 = 0.5798 or 57.986%

sbiii) total loans= 1 from table (288) divide by total from table (3000) = 0.096 or 9.6%

sbiV) 812/2712 = 0.2994 or 29.94%

sbV) 1624/2712 = 0.5988 or 59.88%

sbVi) total loans=0 from table(2712) divided by total from table (3000) = 0.904 or 90.4%

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

(0.3194 * 0.5798 * 0.096)/[(0.3194 * 0.5798 * 0.096)+(0.2994 * 0.5988 * 0.904)] = 0.0988505642823701 or 9.885%

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

###There is no significant difference between 0.096363, or 9.64%, and 0.0988505642823701, or 9.885%. The pivot table value is the more accurate estimated value since it does not rely on the probabilities being independent. While E analyzes probability of each of those counts, B uses a direct computation from a count. As a result, B is more specific, whereas E is more generic.

```
##G. Which of the entries in this table are needed for computing P(Loan = 1 | CC = 1, Online = 1)? Run

## Displaying TRAINING dataset
sb_data.sb <- naiveBayes(Personal.Loan ~ ., data = Train_Data)
sb_data.sb
```

```
##
## 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:
##    Online
## Y           0         1
##   0 0.4011799 0.5988201
##   1 0.4201389 0.5798611
##
##    CreditCard
## Y           0         1
##   0 0.7005900 0.2994100
##   1 0.6805556 0.3194444
```

##The pivot table in step B may be used to quickly compute P(LOAN=1|CC=1,Online=1) without using the Naive Bayes model, whereas using the two tables produced in step C makes it straightforward and obvious HOW you are computing P(LOAN=1|CC=1,Online=1) using the Naive Bayes model.

###The model forecast, however, is lower than the probability estimated manually in step E. The Naive Bayes model predicts the same probability as the preceding techniques. The predicted probability is closer to the one from step B. This is possible because step E needs manual computation, which raises the chance of error when rounding fractions and resulting in a rough estimate.

```
## sb confusion matrix for Train_Data
##TRAINING
prediction_class <- predict(sb_data.sb, newdata = Train_Data)
confusionMatrix(prediction_class, Train_Data$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
##
```

###Despite being very sensitive, this model had a low specificity. The model projected that all values would be 0 in the absence of all real values from the reference. Because of the enormous number of 0, even if the model missed all values of 1, it still yields 90.4% accuracy.

```
prediction.probab <- predict(sb_data.sb, newdata=Validation.Data, type="raw")
prediction_class <- predict(sb_data.sb, newdata = Validation.Data)
confusionMatrix(prediction_class, Validation.Data$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
##
```

#Let's look at the model graphically and choose the best threshold.

```r
library(pROC)
```

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

```
##
## Attaching package: 'pROC'
```

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

```r
roc(Validation.Data$Personal.Loan,prediction.probab[,1])
```

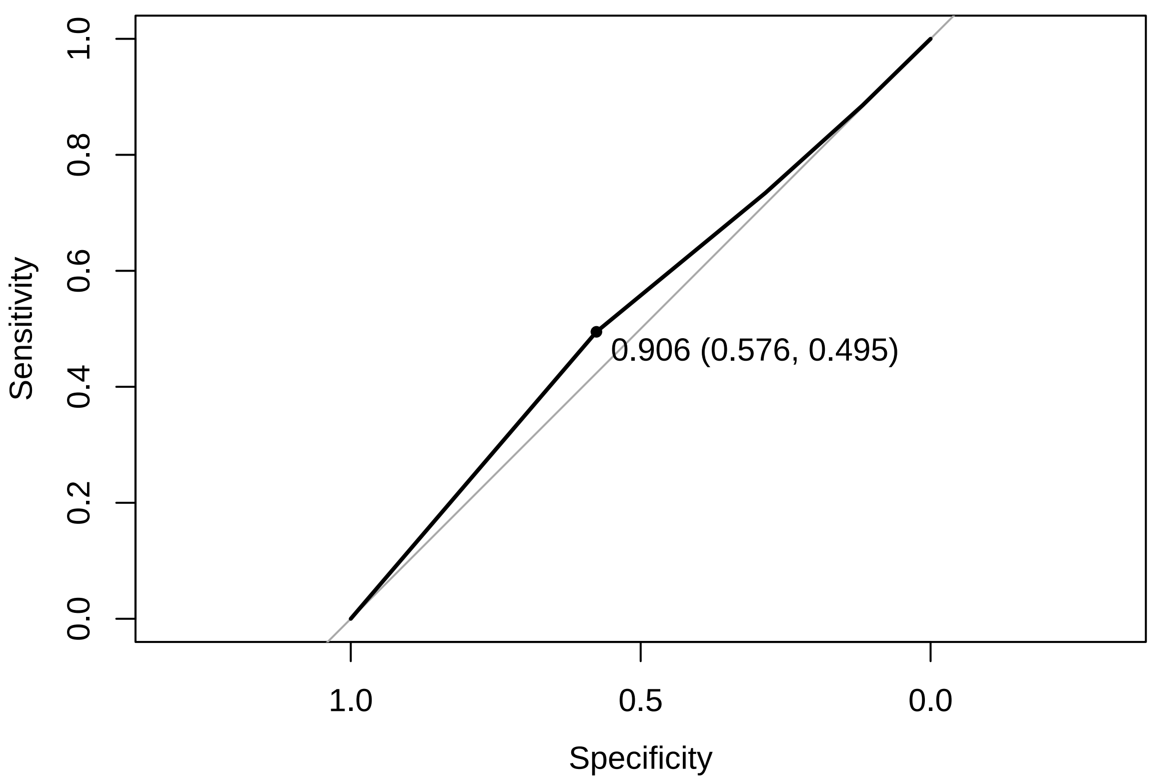
```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
##
## Call:
## roc.default(response = Validation.Data$Personal.Loan, predictor = prediction.probab[,     1])
##
## Data: prediction.probab[, 1] in 1808 controls (Validation.Data$Personal.Loan 0) < 192 cases (Validat
## Area under the curve: 0.5302
```

```r
plot.roc(Validation.Data$Personal.Loan,prediction.probab[,1],print.thres="best")
```

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



##As a consequence, it is possible to demonstrate that the model might be improved by adopting a cutoff of 0.906, which would lower sensitivity to 0.495 and increasing specificity to 0.576.