Naive Bayes for classification

Nikhil Reddy Addula

2022-10-16

```
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)
```

```
NR_bank <- read.csv("~/Documents/assignments/FUNDAMENTALS ML/UniversalBank.csv")
```

```
##The following portion simply extracts the csv file, eliminates ID and zip code (like last time, but p
nv2cf <- NR_bank %>% select(Age, Experience, Income, Family, CCAvg, Education, Mortgage, Personal.Loan,
nv2cf$CreditCard <- as.factor(nv2cf$CreditCard)
nv2cf$Personal.Loan <- as.factor((nv2cf$Personal.Loan))
nv2cf$Online <- as.factor(nv2cf$Online)
```

```
#This creates the data partition, train data and validation data
selected.var <- c(8,11,12)
set.seed(23)
Train_Index = createDataPartition(nv2cf$Personal.Loan, p=0.60, list=FALSE)
Train_Data = nv2cf[Train_Index,selected.var]
Validation_Data = nv2cf[-Train_Index,selected.var]
```

```
##A. Create a pivot table for the training data with Online as a column variable, CC as a row variable,
#CC and LOAN are both rows and online is a column in the generated pivot table.

attach(Train_Data)
##ftable "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 that Online=1 and CC=1, we add 53 (Loan=1 from ftable) to 497 (Loan=0 from ftable), which equals 550, to obtain 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 code above displays a percentage pivot table, which shows the probabilities of a loan based 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)
```

##Above in the first, "Online" compensates a column, "Loans" puts up 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
```

NRi) 92/288 = 0.3194 or 31.94%

NRii) 167/288 = 0.5798 or 57.986%

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

NRiV) 812/2712 = 0.2994 or 29.94%

NRV) 1624/2712 = 0.5988 or 59.88%

NRVi) 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?

Among both 0.096363, or 9.64%, and 0.0988505642823701, or 9.885%, there is no significant difference. Since it does not depend on the probabilities being independent, the pivot table value is the estimated value that is more accurate. While E analyzes probability of each of those counts, B employs a straight computation from a count. As a result, B is more precise whereas E is ideal for generality.

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

##TRAINING dataset
NR_bank.nb <- naiveBayes(Personal.Loan ~ ., data = Train_Data)
NR_bank.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:
##    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 rapidly compute P(LOAN=1|CC=1,Online=1) without relying on the Naive Bayes model, while utilizing the two tables established in step C makes it simple and apparent HOW you are computing P(LOAN=1|CC=1,Online=1)using the Naive Bayes model.

However, the model prediction is lower than the probability calculated manually in step E. The Naive Bayes model predicts the same probability as the methods employed previously. The probability that was estimated is closer to the one from step B. This could be the case since step E requires manual calculation, which introduces the possibility of inaccuracy when rounding fractions and results in simply an approximation.

```
## NB confusion matrix for Train_Data
##TRAINING
pred.class <- predict(NR_bank.nb, newdata = Train_Data)
confusionMatrix(pred.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
##
```

This model exhibited a relatively poor specificity despite being very sensitive. All values were predicted by the model to be 0, lacking all actual values from the reference. Even if the model missed all values of 1, it still provides a 90.4% accuracy because of the large amount of 0.

```
pred.prob <- predict(NR_bank.nb, newdata=Validation_Data, type="raw")
pred.class <- predict(NR_bank.nb, newdata = Validation_Data)
confusionMatrix(pred.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 now examine the model graphically and select the ideal threshold.

```
library(pROC)
```

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

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

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

```
roc(Validation_Data$Personal.Loan,pred.prob[,1])
```

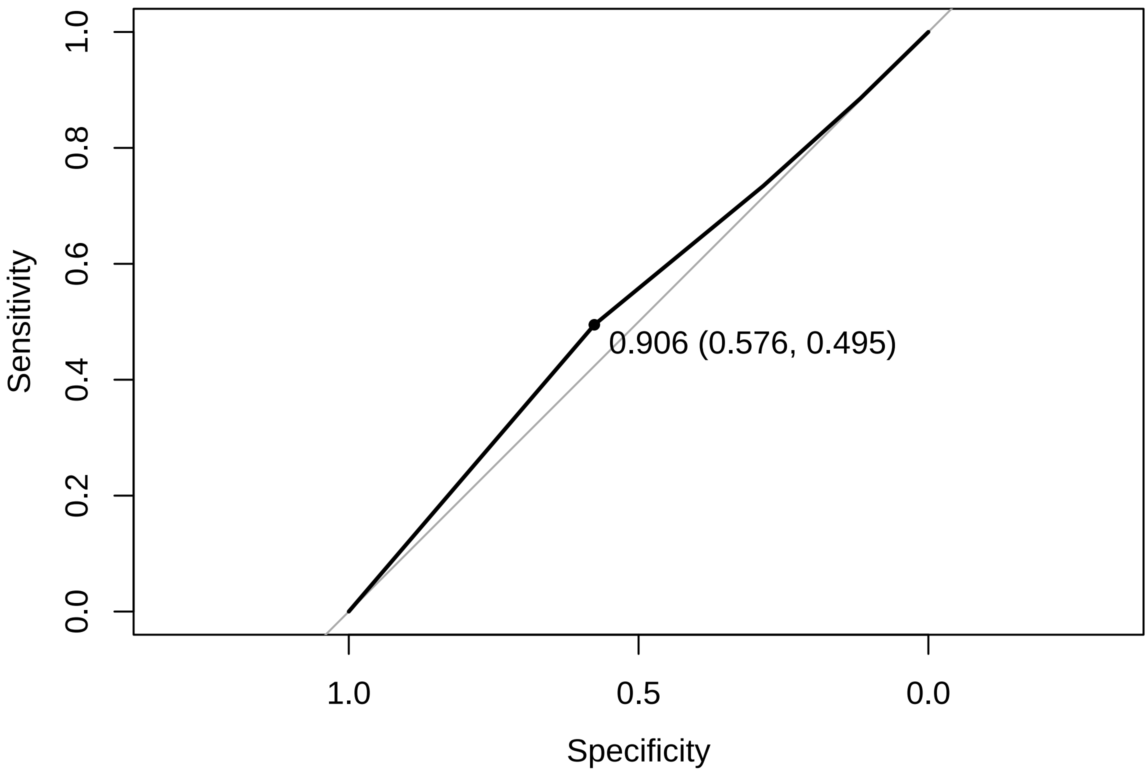
```
## Setting levels: control = 0, case = 1
```

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

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

```
plot.roc(Validation_Data$Personal.Loan,pred.prob[,1],print.thres="best")
```

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



As a result, it can be shown that the model might be improved by using a cutoff of 0.906, which would reduce sensitivity to 0.495 and raise specificity to 0.576.