FML3

Vijay

2022-03-04

```
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(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

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

```
Original <- read.csv("UniversalBank.csv")
UniBank_df <- Original %>% select(Age, Experience, Income, Family, CCAvg, Education, Mortgage, Personal
UniBank_df$CreditCard <- as.factor(UniBank_df$CreditCard)
UniBank_df$Personal.Loan <- as.factor((UniBank_df$Personal.Loan))
UniBank_df$Online <- as.factor(UniBank_df$Online)
```

```
selected.var <- c(8,11,12)
set.seed(23)
train.index= createDataPartition(UniBank_df$Personal.Loan, p=0.60, list=FALSE)
traindata = UniBank_df[train.index,selected.var]
validationdata = UniBank_df[-train.index,selected.var]
```

```r
attach(traindata)
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
```

```r
detach(traindata)
```

probability is 53/(53+497) =53/550 = 0.096363

```r
prop.table(ftable(traindata$CreditCard,traindata$Online,traindata$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
```

```r
attach(traindata)
ftable(Personal.Loan,Online)
```

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

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

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

```r
detach(traindata)
```

```r
prop.table(ftable(traindata$Personal.Loan,traindata$CreditCard),margin=1)
```

```
##             0         1
##
## 0  0.7005900 0.2994100
## 1  0.6805556 0.3194444
```

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

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

Di) 92/288 = 0.3194 or 31.94%

Dii) 167/288 = 0.5798 or 57.986%

Diii) total loans= 1 from table (288) divided by total count from table (3000) = 0.096 or 9.6%

DiV) 812/2712 = 0.2994 or 29.94%

DV) 1624/2712 = 0.5988 or 59.88%

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

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

F)B is more accurate.

```
‘‘‘r
Universalbank.nb <- naiveBayes(Personal.Loan ~ ., data = traindata)
Universalbank.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
```

```r
pred.class <- predict(Universalbank.nb, newdata = traindata)
confusionMatrix(pred.class, traindata$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 extremely sensitive, this model had a low specificity. All values were expected to be zero in the model, however the reference had all true values. Due to the large amount of 0 values, the model still gives a 90.4 percent accuracy despite missing all 1 data. ## Validation set

```r
#confusionMatrix
pred.prob <- predict(Universalbank.nb, newdata=validationdata, type="raw")
pred.class <- predict(Universalbank.nb, newdata = validationdata)
confusionMatrix(pred.class, validationdata$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
##
```

```
library(pROC)
```

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

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

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

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

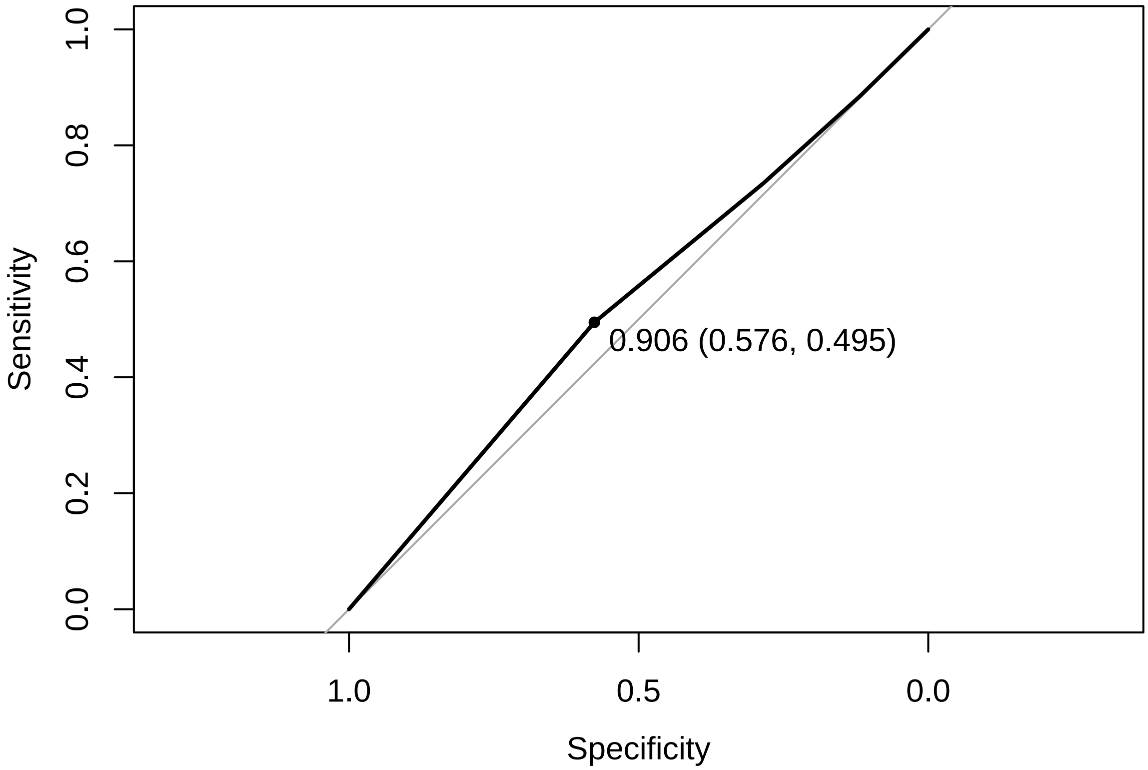
```
## Setting levels: control = 0, case = 1
```

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

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

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

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



This suggests that lowering the sensitivity to 0.495 and boosting the specificity to 0.576 by setting a threshold of 0.906 could enhance the model.