FML_Assignment_3

2024-02-28

```r
#capabilities("X11")
```

```r
library(caret)
```

Importing necessary libraries

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```r
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
```

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

```r
#importing The Data Set
library(readr)
Universal_Bank_data <- read.csv("/Users/meghana/Downloads/UniversalBank.csv")
#View(Universal_Bank_data)
```

```
Universal_Bank_data_1 <- Universal_Bank_data %>% select(Age, Experience, Income, Family, CCAvg, Educati
Universal_Bank_data_1$CreditCard <- as.factor(Universal_Bank_data_1$CreditCard)
Universal_Bank_data_1$Personal.Loan <- as.factor((Universal_Bank_data_1$Personal.Loan))
Universal_Bank_data_1$Online <- as.factor(Universal_Bank_data_1$Online)
```

The consecutive section simply extracts the csv file, removes the ID and zip code (like last time), and then creates the appropriate variables factors, changing numerical variables to categorical first.

```
select.var = c(8,11,12)
set.seed(23)
Train_Index = createDataPartition(Universal_Bank_data_1$Personal.Loan, p=0.60, list=FALSE)
Train_Data = Universal_Bank_data_1[Train_Index,select.var]
Validation_Data = Universal_Bank_data_1[-Train_Index,select.var]
```

This creates the data separation, as well as the train and validation data.

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(). In Python, use panda dataframe methods melt() and pivot().

```
attach(Train_Data)
##ftable is defined as "function table".
ftable(CreditCard,Personal.Loan,Online)
```

In the resulting pivot table CC and LOAN are both rows, and online is a column.

```
##                          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.

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

B. 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)].

```
##                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.

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

C. 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.

```
##               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(Train_Data)
```

"Online" compensates a column, "Loans" compensates a row, and "Credit Card" compensates a column in the first example above.

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

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

```
##              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
```

```
#### i) P(CC = 1 | Loan = 1) (the proportion of credit card holders among the loan acceptors)
92/288
```

```
## [1] 0.3194444
```

```
#### ii) P(Online=1|Loan=1)
167/288
```

```
## [1] 0.5798611
```

```
#### iii) P(Loan = 1) (the proportion of loan acceptors)
288/3000
```

```
## [1] 0.096
```

```


#### iv) P(CC=1|Loan=0)
812/2712
```

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

```
## [1] 0.29941
```

```
#### v) P(Online=1|Loan=0)
1624/2712
```

```
## [1] 0.5988201
```

```
#### vi) P(Loan = 0)
2712/3000
```

```
## [1] 0.904
```

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?

The difference between 0.096363, or 9.64%, and 0.0988505642823701, or 9.885%, is not statistically significant. Since the pivot table value does not depend on the probabilities being independent, it is the more accurate estimated value. B uses a direct computation from a count, while E analyses the probability of each of those counts. Because of this, B is more particular while E is more general.

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

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

While P(LOAN=1|CC=1,Online=1) can be found using the pivot table in step B without the Naive Bayes model, it is clear and simple to use the two tables produced in step C. How to use the Naive Bayes model to find P(LOAN=1|CC=1,Online=1).

That being said, the probability that was manually estimated in step E is higher than the model forecast. The probability predicted by the Naive Bayes model is the same as that of the earlier methods. The estimated probability is more in line with the step B estimate. This is made feasible by the fact that step E requires manual computation, which increases the possibility of inaccuracy when rounding fractions and producing an approximate result.

Confusion matrix for Train_Data

```r
prediction_class <- predict(sb_data.sb, newdata = Train_Data)
confusionMatrix(prediction_class, Train_Data$Personal.Loan)
```

Training

```
## 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
##
```

```r
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)
```

Although this model had a low specificity, it was very sensitive. The model predicted that all values would be 0 in the case that all real values from the reference were missing. Because

there are so many zeros, the model would still have a 90.4% accuracy rate even if it missed every value of 1.

```
## 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
##
```

```r
library(pROC)
```

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

```
## 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])
```

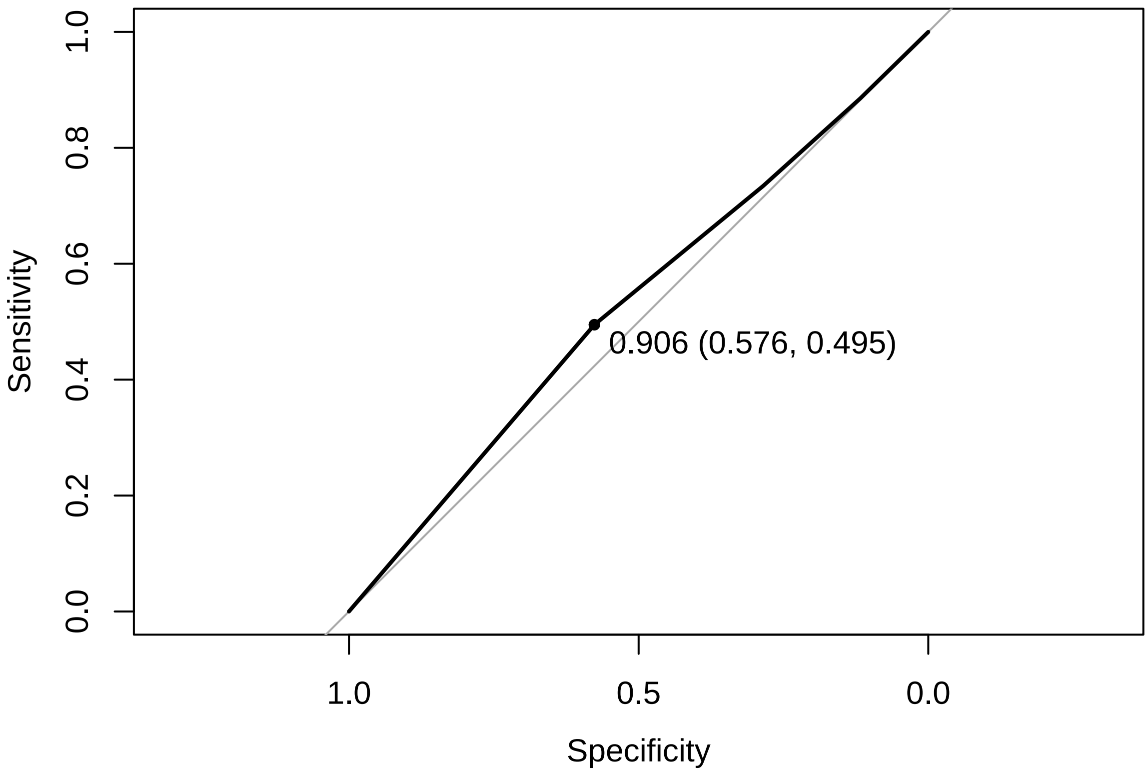
```
## Setting levels: control = 0, case = 1
```

```
## 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
```



Therefore, it is possible to show that a cutoff of 0.906, which would increase specificity to 0.576 and decrease sensitivity to 0.495, could improve the model.