Naive Bayes Classification

64060_cosadebe

2025-10-08

```r
## Load needed libraries
suppressPackageStartupMessages({library(plyr)
  library(dplyr)
  library(reshape2)      # for pivot table melt() & cast() functions
  library(caret)
  library(e1071)           # for naveBayes() classification
  })
```

```r
## Read UniversalBank dataset
UniversalBank <- read.csv("UniversalBank.csv")

# Select relevant variables for modeling
UniversalBank <- UniversalBank %>%
  select(Personal.Loan, CreditCard, Online)

set.seed(123)  # set seed for reproducibility

## Partition dataset into training (60%) and validation (40%) sets
Train_Index <- createDataPartition(UniversalBank$Personal.Loan,
                                   p = 0.6, list = FALSE)

Train <- UniversalBank[Train_Index, ]
validation <- UniversalBank[-Train_Index, ]
```

A. Create a pivot table for the training data with Online varriable as a column variable

```r
# create column to count occurrences
Train$count <- 1

# reshape Train data into long format:
Train_melt <- melt(Train, id.vars = c("CreditCard", "Personal.Loan", "Online"),
                   measure.vars = "count")

# reshape into wide format
Train_pivot_table <- dcast(Train_melt, CreditCard + Personal.Loan ~ Online,
                           fun.aggregate = sum)
```

```
# update col names of Online classes
names(Train_pivot_table)[3:4] <- c("Online_0", "Online_1")

print(Train_pivot_table)
```

```
##   CreditCard Personal.Loan Online_0 Online_1
## 1          0             0      785     1145
## 2          0             1       65      122
## 3          1             0      317      475
## 4          1             1       34       57
```

B. Calculate the probability that a customer will accept a loan offer, given they own a credit card (CreditCard = 1) and use online banking (Online = 1):

- P(Loan = 1 | CC = 1, Online = 1)

```
prob_Loan1_given_CC1_OnL1 <- Train_pivot_table %>%
  summarise(prob = Online_1[CreditCard == 1 & Personal.Loan == 1] /
            sum(Online_1[CreditCard == 1]))

print(prob_Loan1_given_CC1_OnL1)
```

```
##        prob
## 1 0.1071429
```

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

- Loan (rows) as a function of Online (columns)
- Loan (rows) as a function of CC (columns)

```
# Loan (rows) as a function of Online (columns) - pivot table
Train_pivot_1 <- dcast(Train_melt, Personal.Loan ~ Online,
                       fun.aggregate = sum)
# update col names of Online classes
names(Train_pivot_1)[2:3] <- c("Online_0", "Online_1")
print(Train_pivot_1)
```

```
##   Personal.Loan Online_0 Online_1
## 1             0     1102     1620
## 2             1       99      179
```

```
# Loan (rows) as a function of CC  (columns) - pivot table
Train_pivot_2 <- dcast(Train_melt, Personal.Loan ~ CreditCard,
                       fun.aggregate = sum)
# update col names of Online classes
names(Train_pivot_2)[2:3] <- c("CreditCard_0", "CreditCard_1")
print(Train_pivot_2)
```

```
##   Personal.Loan CreditCard_0 CreditCard_1
## 1             0         1930          792
## 2             1          187           91
```

D. Compute the following quantities:

i. P(CC = 1 | Loan = 1)

```r
prob_CC1_given_loan1 <- Train_pivot_2 %>%
  filter(Personal.Loan == 1) %>%
  summarise(prob = CreditCard_1/(CreditCard_1 + CreditCard_0))

print(prob_CC1_given_loan1)
```

```
##        prob
## 1 0.3273381
```

ii. P(Online = 1 | Loan = 1)

```r
prob_Online1_given_loan1 <- Train_pivot_1 %>%
  filter(Personal.Loan == 1) %>%
  summarise(prob = Online_1/(Online_1 + Online_0))

print(prob_Online1_given_loan1)
```

```
##        prob
## 1 0.6438849
```

iii. P(Loan = 1) (the proportion of loan acceptors)

```r
prob_loan1 <- mean(Train$Personal.Loan == 1)

print(prob_loan1)
```

```
## [1] 0.09266667
```

iv. P(CC = 1 | Loan = 0)

```r
prob_CC1_given_loan0 <- Train_pivot_2 %>%
  filter(Personal.Loan == 0) %>%
  summarise(prob = CreditCard_1/(CreditCard_1 + CreditCard_0))

print(prob_CC1_given_loan0)
```

```
##        prob
## 1 0.2909625
```

v. P(Online = 1 | Loan = 0)

```r
prob_Online1_given_loan0 <- Train_pivot_1 %>%
  filter(Personal.Loan == 0) %>%
  summarise(prob = Online_1/(Online_1 + Online_0))

print(prob_Online1_given_loan0)
```

```
##        prob
## 1 0.5951506
```

vi. P(Loan = 0)

```
prob_loan0 <- Train %>%
  count(Personal.Loan) %>%
  summarise(prob = n[Personal.Loan == 0]/sum(n))

print(prob_loan0)
```

```
##        prob
## 1 0.9073333
```

E. Use the quantities computed above to compute the naive Bayes probability

```
# for Loan = 1
# P(Loan = 1 | CC = 1, Online = 1) = P(Loan = 1) * P(CC = 1 | Loan = 1) *
# P(Online = 1 | Loan = 1)
NB_prob_Loan1_given_CC1_OnL1 <- prob_loan1 * prob_CC1_given_loan1 *
                                prob_Online1_given_loan1

# for Loan = 0
# P(Loan = 0 | CC = 1, Online = 1) = P(Loan = 0) * P(CC = 1 | Loan = 0) *
# P(Online = 1 | Loan = 0)
NB_prob_Loan0_given_CC1_OnL1 <- prob_loan0 * prob_CC1_given_loan0 *
                                prob_Online1_given_loan0

# Normalizing
norm_NB_p_Loan1_given_CC1_OnL1 <- NB_prob_Loan1_given_CC1_OnL1 /
                                  (NB_prob_Loan1_given_CC1_OnL1 +
                                   NB_prob_Loan0_given_CC1_OnL1 )

print(norm_NB_p_Loan1_given_CC1_OnL1)
```

```
##        prob
## 1 0.1105637
```

F. Compare the probability obtained by the Naive Bayes Classifier in part E to the probability calculated using Naive Bayes' Theorem in part B

- Accuracy is compared using the Brier Score metric - (probability MSE)

```
#Calculate Brier score for probability obtained from part B - Bayes Theorem
Brier_Score_B <- data.frame(Personal.Loan = Train$Personal.Loan,
                            prob = rep(prob_Loan1_given_CC1_OnL1)) %>%
  summarise(MSE = mean((as.numeric(Personal.Loan) - prob)^2))
```

```
#Calculate Brier score for probability obtained from part E - Naive Bayes Probability
Brier_Score_E <- data.frame(Personal.Loan = Train$Personal.Loan,
                           prob = rep(norm_NB_p_Loan1_given_CC1_OnL1)) %>%
  summarise(MSE = mean((as.numeric(Personal.Loan) - prob)^2))

print(paste("Brier_Score_B:", round(Brier_Score_B,4)))
```

```
## [1] "Brier_Score_B: 0.0843"
```

```
print(paste("Brier_Score_E:", round(Brier_Score_E,4)))
```

```
## [1] "Brier_Score_E: 0.0844"
```

On calibrated scale of 0 (perfect) to 1 (worst) for Brier Score, Brier_Score_B: 0.0843 is closer to 0 than Brier_Score_E: 0.0844 - that is, the probability obtained from part B: 0.1071429 is more accurate than probability obtained from part E: 0.1105637.

G. Naive Bayes Classification

- The Personal.Loan, CreditCard, and Online variables from the UniversalBank
- table are needed for computing P(Loan = 1 | CC = 1, Online = 1)

```
# drop count column from Train data initialized in part A.
Train <- Train[, -4]

# build a naive Bayes classifier
nb_model <- naiveBayes(Personal.Loan~CreditCard+Online, data = Train)

# returning probability of both classifcation for output "Y"
Predicted_Train_label_ <- predict(nb_model, Train, type = "raw")

# Create dataframe for returned probablitites
df <- data.frame(Predicted_Train_label_)

# combine Train data and df, subset for entry that corresponds to
# P(Loan = 1 | CC = 1, Online = 1
df_sub <- subset(cbind(Train, df), Personal.Loan == 1 & CreditCard == 1 &
                   Online == 1)

head(df_sub,1)
```

```
##    Personal.Loan CreditCard Online        X0        X1
## 30             1          1      1 0.8843065 0.1156935
```

The probability value from the model output for P(Loan = 1 | CC = 1, Online = 1): 0.1156935, is slightly greater than the value obtained using the Naive Bayes probability formula in part E: 0.1105637.