```r
library(class)
library(caret)
```

```
## 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(e1071)
```

```r
data = read.csv("/Users/srikanthgembali/Downloads/UniversalBank.csv")
head(data)
```

```
##   ID Age Experience Income ZIP.Code Family CCAvg Education Mortgage
## 1  1  25          1     49    91107      4   1.6         1        0
## 2  2  45         19     34    90089      3   1.5         1        0
## 3  3  39         15     11    94720      1   1.0         1        0
## 4  4  35          9    100    94112      1   2.7         2        0
## 5  5  35          8     45    91330      4   1.0         2        0
## 6  6  37         13     29    92121      4   0.4         2      155
##   Personal.Loan Securities.Account CD.Account Online CreditCard
## 1             0                  1          0      0          0
## 2             0                  1          0      0          0
## 3             0                  0          0      0          0
## 4             0                  0          0      0          0
## 5             0                  0          0      0          1
## 6             0                  0          0      1          0
```

```r
dim(data)
```

```
## [1] 5000   14
```

```r
# Partition the data into training (60%) and validation (40%) sets
set.seed(123)
trainIndex <- createDataPartition(data$Personal.Loan, p = 0.6, list = FALSE)
trainData <- data[trainIndex, ]
validationData <- data[-trainIndex, ]
```

```
dim(trainData)
```

```
## [1] 3000   14
```

```
dim(validationData)
```

```
## [1] 2000   14
```

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.

```
library(pander)
subset = trainData[c("CreditCard","Personal.Loan","Online")]
pivot_table = ftable(subset)
pandoc.table(pivot_table,style = "grid", split.tables = Inf)
```

```
##
##
## +------------+---------------+--------+-----+------+
## |            |               | Online |  0  |  1   |
## +------------+---------------+--------+-----+------+
## | CreditCard | Personal.Loan |        |     |      |
## +------------+---------------+--------+-----+------+
## |     0      |       0       |        | 785 | 1145 |
## +------------+---------------+--------+-----+------+
## |            |       1       |        | 65  | 122  |
## +------------+---------------+--------+-----+------+
## |     1      |       0       |        | 317 | 475  |
## +------------+---------------+--------+-----+------+
## |            |       1       |        | 34  |  57  |
## +------------+---------------+--------+-----+------+
```

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)]

```
probability = 57/532
cat("Prob (Loan = 1 | CC = 1, Online = 1):", round(probability*100,2),"%")
```

```
## Prob (Loan = 1 | CC = 1, Online = 1): 10.71 %
```

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_subset = trainData[c("Personal.Loan","Online")]
pivot_online = ftable(online_subset)
pandoc.table(pivot_online,style = "grid", split.tables = Inf)
```

```
##
##
## +---------------+--------+------+------+
## |               | Online |  0   |  1   |
## +---------------+--------+------+------+
## | Personal.Loan |        |      |      |
## +---------------+--------+------+------+
## |       0       |        | 1102 | 1620 |
## +---------------+--------+------+------+
## |       1       |        |  99  | 179  |
## +---------------+--------+------+------+
```

```
cc_subset = trainData[c("Personal.Loan","CreditCard")]
pivot_cc = ftable(cc_subset)
pandoc.table(pivot_cc,style = "grid", split.tables = Inf)
```

```
##
##
## +---------------+------------+------+-----+
## |               | CreditCard |  0   |  1  |
## +---------------+------------+------+-----+
## | Personal.Loan |            |      |     |
## +---------------+------------+------+-----+
## |       0       |            | 1930 | 792 |
## +---------------+------------+------+-----+
## |       1       |            | 187  | 91  |
## +---------------+------------+------+-----+
```

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

i. P(CC = 1 | Loan = 1) (the proportion of credit card holders among the loan acceptors)

```
cc_loan1 <- pivot_cc[2,2]  # Number of credit card holders among loan acceptors
loan1 <- sum(pivot_cc[2,])  # Total number of loan acceptors
CC_given_loan1 <- cc_loan1 / loan1
cat("Prob (CC = 1 | Loan = 1):", round(CC_given_loan1*100,2),"%")
```

```
## Prob (CC = 1 | Loan = 1): 32.73 %
```

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

```
online_loan1 <- pivot_online[2,2]  # Number of online banking users among loan acceptors
loan2 <- sum(pivot_online[2,])  # Total number of loan acceptors
online_given_loan1 <- online_loan1 / loan2
cat("Prob (Online = 1 | Loan = 1):", round(online_given_loan1*100,2),"%")
```

```
## Prob (Online = 1 | Loan = 1): 64.39 %
```

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

```r
loan_acceptors <- sum(trainData$Personal.Loan == 1)
total <- nrow(trainData)
loan_acceptors1 <- loan_acceptors/total
cat("Prob (Loan = 1):", round(loan_acceptors1*100,2),"%")
```

```
## Prob (Loan = 1): 9.27 %
```

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

```r
cc_loan2 <- pivot_cc[1,2] # Number of credit card users among non-loan acceptors
loan3 <- sum(pivot_cc[1,]) # Total number of non-loan acceptors
CC_notgiven_loan <- cc_loan2 / loan3
cat("Prob (CC = 1 | Loan = 0):", round(CC_notgiven_loan*100,2),"%")
```

```
## Prob (CC = 1 | Loan = 0): 29.1 %
```

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

```r
online_loan2 <- pivot_online[1,2]  # Number of online banking users among non-loan acceptors
loan4 <- sum(pivot_online[1,])  # Total number of non-loan acceptors
online_notgiven_loan <- online_loan2 / loan4
cat("Prob (Online = 1 | Loan = 0):", round(online_notgiven_loan*100,2),"%")
```

```
## Prob (Online = 1 | Loan = 0): 59.52 %
```

vi. P(Loan = 0)

```r
nonloan_acceptors <- 1 - loan_acceptors1
cat("Prob (Loan = 0):", round(nonloan_acceptors*100,2),"%")
```

```
## Prob (Loan = 0): 90.73 %
```

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

```r
nb_prob <- ((CC_given_loan1*online_given_loan1*loan_acceptors1)/((CC_given_loan1*online_given_loan1*loa
cat("Prob (Loan = 1 | CC = 1, Online = 1):", round(nb_prob*100,2),"%")
```

```
## Prob (Loan = 1 | CC = 1, Online = 1): 11.06 %
```

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

10.71% (in B) is almost similar to the Naive Bayes probability 11.06%. This method requires same independent variables to make predictions and also limited by the exact classification of the independent variables, where as the Naive Bayes does not require as it can be more flexible with its predictions, but also may be less precise due to simplifying assumptions of independence among features.

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).

```r
# 3 entries are required to compute P(Loan = 1 | CC = 1, Online = 1) which are included in subset funct
model <- naiveBayes(Personal.Loan ~ ., data = subset)
model
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##          0          1
## 0.90733333 0.09266667
##
## Conditional probabilities:
##    CreditCard
## Y        [,1]      [,2]
##   0 0.2909625 0.4542897
##   1 0.3273381 0.4700881
##
##    Online
## Y        [,1]      [,2]
##   0 0.5951506 0.4909531
##   1 0.6438849 0.4797134
```

```r
result <- (0.327 * 0.643 * 0.092) / ((0.327 * 0.643 * 0.092) + (0.290 * 0.595 * 0.907))
cat("Prob (Loan = 1 | CC = 1, Online = 1):", round(result*100),"%")
```

```
## Prob (Loan = 1 | CC = 1, Online = 1): 11 %
```

The output calculated from the model is 11%, which is similar to the output obtained in E i.e 11.6%