7.2 개인 대출 수락

a. Consider the following customer:

Age = 40, Experience = 10, Income = 84, Family = 2, CCAvg = 2, Education_1 = 0, Education_2 = 1, Education_3 = 0, Mortgage = 0, Securities Account = 0, CD Account = 0, Online = 1, and Credit Card = 1. Perform a k-NN classification with all predictors except ID and ZIP code using $k = 1$. Remember to transform categorical predictors with more than two categories into dummy variables first.

1) Education 변수를 dummy variable로 변환

```
setwd("C:/rdata")
bank.df <- read.csv("UniversalBank.csv")
str(bank.df)

###############create dummy variable
bank.df$Education <- factor(bank.df$Education)
edu.df <- as.data.frame(model.matrix(~0 + Education, data=bank.df))
edu.df
bank.df <- cbind(bank.df[, -c(1, 5, 8)], edu.df[,])
```

2) 데이터 분할

```
set.seed(111)
train.index <- sample(row.names(bank.df), 0.6*dim(bank.df)[1])
valid.index <- setdiff(row.names(bank.df), train.index)
train.df <- bank.df[train.index, ]
valid.df <- bank.df[valid.index, ]
```

3) new data 생성

```
###############creat new data
new.df <- data.frame(Age = 40, Experience = 10, Income = 84,
                     Family = 2, CCAvg = 2, Mortgage = 0, Securities.Account = 0,
                     CD.Account = 0, Online = 1, CreditCard = 1,
                     Education1 = 0, Education2 = 1, Education3 = 0)
```

4) 정규화

```
###############normination
library(caret)
train.norm.df <- train.df
valid.norm.df <- valid.df
bank.norm.df <- bank.df
new.norm.df <- new.df

norm.values <- preProcess(train.df[,-7], method = c("center", "scale"))
train.norm.df[, -7] <- predict(norm.values, train.df[, -7])
valid.norm.df[, -7] <- predict(norm.values, valid.df[, -7])
bank.norm.df[, -7] <- predict(norm.values, bank.df[, -7])
new.norm.df <- predict(norm.values, new.df)
```

5) k-NN 분류

```
###############knn(k=1)
library(FNN)
nn <- knn(train=train.norm.df[ , -7], test=new.norm.df, cl=train.norm.df[ , 7], k=1, prob=TRUE)
nn
```

```
> nn
[1] 0
attr(,"prob")
[1] 1
attr(,"nn.index")
      [,1]
[1,] 2899
attr(,"nn.dist")
          [,1]
[1,] 0.4796033
Levels: 0
```

=>New고객의 Personal.loan 값을 0으로 분류한다.

c. 최적의 k를 사용하여 검증 세트에 대한 정오행렬표를 만드시오.

(1) 최적의 k 찾기

```
###############find optimal k
library(caret)
accuracy.df <- data.frame(k=seq(1,5000,1), accuracy=rep(0,5000))
dim(accuracy.df)
accuracy.df

valid.norm.df[, 7] <- as.factor(valid.norm.df[, 7])
class(knn.pred)
class(valid.norm.df[, 7])

for(i in 1:5000){
  knn.pred <- knn(train=train.norm.df[ , -7], test=valid.norm.df[ , -7],
                  cl=train.norm.df[, 7], k=i)
  accuracy.df[i,2]<- confusionMatrix(knn.pred, valid.norm.df[, 7])$overall[1]
}

accuracy.df
```

```
> accuracy.df
    k accuracy
1   1   0.9580
2   2   0.9540
3   3   0.9630
4   4   0.9515
5   5   0.9570
6   6   0.9525
7   7   0.9545
8   8   0.9485
9   9   0.9530
10 10   0.9470
11 11   0.9465
12 12   0.9450
13 13   0.9455
14 14   0.9425
15 15   0.9445
16 16   0.9430
17 17   0.9440
18 18   0.9415
19 19   0.9435
20 20   0.9410
21 21   0.9410
```

=> k값을 3으로 선택

(2) 정오행렬표 만들기

```
knn.pred <- knn(train = train.norm.df[, -7], test=valid.norm.df[, -7],
                cl = train.norm.df[, 7], k=3)
conf <- confusionMatrix(knn.pred, as.factor(valid.norm.df[, 7]), positive = '1')
conf
```

```
> conf
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1790   65
         1    9  136

               Accuracy : 0.963
                 95% CI : (0.9538, 0.9708)
    No Information Rate : 0.8995
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7665

 Mcnemar's Test P-Value : 1.62e-10

            Sensitivity : 0.6766
            Specificity : 0.9950
         Pos Pred Value : 0.9379
         Neg Pred Value : 0.9650
             Prevalence : 0.1005
         Detection Rate : 0.0680
   Detection Prevalence : 0.0725
      Balanced Accuracy : 0.8358

       'Positive' Class : 1
```

d. Consider the following customer: Age = 40, Experience = 10, Income = 84, Family = 2, CCAvg = 2, Education_1 = 0, Education_2 = 1, Education_3 = 0, Mortgage = 0, Securities Account = 0, CD Account = 0, Online = 1 and Credit Card = 1. Classify the customer using the best k.

```
###############knn(k=3, train=bank.norm.df)
library(FNN)
knn.pred.new <- knn(train=bank.norm.df[ , -7], test=new.norm.df, cl=bank.norm.df[ , 7], k=3, prob=TRUE)
knn.pred.new
row.names(bank.df)[attr(knn.pred.new, "nn.index")]
```

```
> knn.pred.new
[1] 0
attr(,"prob")
[1] 1
attr(,"nn.index")
     [,1] [,2] [,3]
[1,] 4035 4408 3399
attr(,"nn.dist")
          [,1]     [,2]      [,3]
[1,] 0.4796033 0.496999 0.6359329
Levels: 0
> row.names(bank.df)[attr(knn.pred.new, "nn.index")]
[1] "4035" "4408" "3399"
```

=>New고객의 Personal.loan값을 0으로 분류한다.

e. 이번에는 데이터를 학습용, 검증용, 그리고 평가용 세트로 다시 분할하시오.(50%:30%:20%). 위에서 선택된 k를 사용하여 k-최근접이웃을 적용하시오. 평가세트에 대한 분류행렬을 학습 세트 및 검증 세트의 정오행렬표와 비교하시오. 차이점을 찾아내고 그 이유에 대하여 설명하시오.

```
###############data partition & norm
set.seed(111)
train.index <- sample(row.names(bank.df), 0.5*dim(bank.df)[1])
valid.index <- sample(setdiff(row.names(bank.df), train.index), 0.3*dim(bank.df)[1])
test.index <- sample(setdiff(row.names(bank.df), c(train.index,valid.index)))

train.df <- bank.df[train.index, ]
valid.df <- bank.df[valid.index, ]
test.df <- bank.df[test.index, ]

train.norm.df <- train.df
valid.norm.df <- valid.df
test.norm.df <- test.df
new.norm.df <- new.df

train.norm.df[, -7] <- predict(norm.values, train.df[, -7])
valid.norm.df[, -7] <- predict(norm.values, valid.df[, -7])
test.norm.df[, -7] <- predict(norm.values, test.df[, -7])
new.norm.df <- predict(norm.values, new.df)

#train confusion matrix
knn.pred <- knn(train=train.norm.df[ , -7], test=train.norm.df[ , -7],
                cl=train.norm.df[, 7], k=3)

conf.train <-confusionMatrix(knn.pred, as.factor(train.norm.df[, 7]), positive = '1')
conf.train

# valid confusion matrix
knn.pred <- knn(train=train.norm.df[ , -7], test=valid.norm.df[ , -7],
                cl=train.norm.df[, 7], k=3)

conf.valid <-confusionMatrix(knn.pred, as.factor(valid.norm.df[, 7]), positive = '1')
conf.valid

# test confusion matirx
knn.pred <- knn(train=train.norm.df[ , -7], test=test.norm.df[ , -7],
                cl=train.norm.df[, 7], k=3)

conf.test <- confusionMatrix(knn.pred, as.factor(test.norm.df[, 7]), positive = '1')
conf.test
```

train. confusion matrix

```
> conf.train
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 2717   63
         1    4  216

               Accuracy : 0.9777
                 95% CI : (0.9717, 0.9827)
    No Information Rate : 0.907
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8537

 Mcnemar's Test P-Value : 1.382e-12

            Sensitivity : 0.77419
            Specificity : 0.99853
         Pos Pred Value : 0.98182
         Neg Pred Value : 0.97734
             Prevalence : 0.09300
         Detection Rate : 0.07200
   Detection Prevalence : 0.07333
      Balanced Accuracy : 0.88636

       'Positive' Class : 1
```

valid. confusion matrix

```
> conf.valid
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1790   65
         1    9  136

               Accuracy : 0.963
                 95% CI : (0.9538, 0.9708)
    No Information Rate : 0.8995
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7665

 Mcnemar's Test P-Value : 1.62e-10

            Sensitivity : 0.6766
            Specificity : 0.9950
         Pos Pred Value : 0.9379
         Neg Pred Value : 0.9650
             Prevalence : 0.1005
         Detection Rate : 0.0680
   Detection Prevalence : 0.0725
      Balanced Accuracy : 0.8358

       'Positive' Class : 1
```

test. confusion matrix

```
> conf.test
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 906  27
         1   3  64

               Accuracy : 0.97
                 95% CI : (0.9574, 0.9797)
    No Information Rate : 0.909
    P-Value [Acc > NIR] : 1.322e-14

                  Kappa : 0.7942

 Mcnemar's Test P-Value : 2.679e-05

            Sensitivity : 0.7033
            Specificity : 0.9967
         Pos Pred Value : 0.9552
         Neg Pred Value : 0.9711
             Prevalence : 0.0910
         Detection Rate : 0.0640
   Detection Prevalence : 0.0670
      Balanced Accuracy : 0.8500

       'Positive' Class : 1
```

>>>accuracy가 유의미한 차이를 보이지 않는다. 즉, 새로운 데이터가 와도 과적합이 아니다.